FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Intelligent Ticket Management Assistant for Helpdesk Operations

**Leonardo Silva Ferreira**

PHD THESIS


Programa Doutoral em Engenharia Informática

Supervisor: Professor Daniel Castro Silva

Co-Supervisor: Mikel Uriarte Itzazelaia

June 13, 2025

# Intelligent Ticket Management Assistant for Helpdesk Operations

**Leonardo Silva Ferreira**

Programa Doutoral em Engenharia Informática

Approved in oral examination by the committee:

Chair: Doutor Pedro Nuno Ferreira da Rosa da Cruz Diniz
External Examiner: Doutor Pedro Manuel Henriques da Cunha Abreu
External Examiner: Doutor Paulo Jorge Freitas de Oliveira Novais
Internal Examiner: Doutor Carlos Manuel Milheiro de Oliveira Pinto Soares
Internal Examiner: Doutora Ana Paula Cunha da Rocha
Supervisor: Doutor Daniel Augusto Gama de Castro Silva

June 13, 2025

# Abstract

With the dynamic evolution of the internet, particularly in domains such as multimedia services, cloud computing, internet of things, virtualization, and artificial intelligence, companies have seen significant expansion in their market and services. However, this growth has also exposed numerous vulnerabilities that threaten the confidentiality, integrity, and availability of organizational and personal data. As information technology analysts work to address security system alerts, artificial intelligence has introduced new avenues for breaching security, ranging from simple, low-cost methods to highly sophisticated attacks. Low-cost approaches, such as phishing and password spraying take advantage of human error and weak password practices. In contrast, challenging threats include advanced persistent attacks and zero-day exploits, which require significant expertise and resources, often disrupting critical systems. Many organizations rely on cybersecurity helpdesk centers, internal or outsourced, to manage incidents. However, these centers often struggle to respond effectively due to data overload and a lack of qualified operators.

This dissertation addresses the shortage of skilled operators and the high volume of incidents in helpdesk operations by developing a ticket management assistant to support human operators in resolving incidents. The platform integrates a context-aware recommender system that identifies the fastest analyst-procedure pair for each incident and continually improves with each treatment followed. To ensure data privacy, this recommender system is trained using artificial data generated by a custom synthetic data generator. Furthermore, this thesis explores the possibility of enhancing this assistant with automated machine learning functionalities to predict incoming tickets. This feature could help managers anticipate workloads and proactively adjust the composition of the security teams.

The development of this tool is supported by the collaboration with a cybersecurity company, S21sec, which provides anonymized historical incident treatment data structures and taxonomies. However, synthetic data generation techniques are essential due to the absence of granular information on incident resolution and related parameters in the shared data set, which also requires privacy. The implemented generator builds artificial datasets that can mimic distributions similar to those observed in the real dataset while emulating real-world behaviours, including ticket prioritization, scheduling, and treatment.

The artificial data generator is evaluated regarding its efficiency in replicating real-world datasets using similarity measures such as Hellinger distance and Kullback-Leibler divergence. Furthermore, several ticket scheduling scenarios are explored, with different number of operators and distributions across three work shifts. The results demonstrate that this module can replicate ticket distributions and treatment durations observed in real datasets. Additionally, it allows for the simulation of real-world helpdesk operations, providing a solid foundation for exploring diverse operational contexts without compromising privacy. The analysis of the ticket scheduling consistently shows that scenarios characterized by a high shift imbalance and fewer operators lead to longer wait times and more tickets scheduled for later treatment.

The recommender system is assessed from two perspectives: scalability and impact on ticket

treatment. The first phase uses various test datasets with different sizes and numbers of operators, analyzed with metrics such as the average recommendation time and memory usage. In contrast, the impact on ticket treatment is examined by considering improvements in ticket waiting times before being allocated to an operator and the response time required for their resolution, using different recommendation acceptance degrees. The results indicate that the number of operators the recommender system utilizes has a slightly larger impact on its scalability than the number of test tickets. Both features show a similar linear growth pattern regarding the referred metrics, but the number of operators has a larger slope. The integration of this recommender system into the ticket treatment for the testing synthetic dataset reduced the average response time by 37.9% to 45.1% and the average wait time by 62.2% to 63.2%, assuming operators always accept the recommendations. With varying recommendation acceptance rates, the average wait time remains constant, while the response time improvement ranges from 0.4% to 11.7%.

The potential application of automated machine learning for predictive analysis is explored through a case study, comparing the system's recommended team dimensionality decisions with expected outcomes. The case study evaluates the system in terms of how accurately it makes predictions about future incident workloads and its effectiveness in suggesting appropriate team size adjustments based on past historical operator performance and the forecasts obtained. Among the tested dataset distributions, models trained in three years of data outperformed those trained on four years, showing a better mean average error using real data on ticket frequency throughout the year. Regarding team dimensionality recommendations, including hiring or dismissing operators, the tool based on automated machine learning frequently proposed decisions closely aligned with those that could have been proposed in the same period.

Collectively, these results show that the proposed system can optimize ticket treatment workflows in real-world applications, leading to more efficient use of resources and reduced operational delays. Furthermore, its ability to simulate real-world operations without compromising privacy allows security operations centers to test several scenarios and refine their strategies.

# Resumo

A evolução dinâmica da internet, particularmente em setores como serviços multimédia, computação na *cloud*, *internet of things*, simulação, e inteligência artificial, levou a que as empresas tenham assistido a uma expansão significativa dos seus serviços e mercados. No entanto, este crescimento também expôs diversas vulnerabilidades que ameaçam a confidencialidade, integridade e disponibilidade dos dados organizacionais e pessoais. Enquanto os especialistas trabalham para responder aos alertas gerados por sistemas de segurança, a inteligência artificial tem introduzido novas formas de comprometer a segurança, variando desde métodos simples e de baixo custo até ataques altamente sofisticados. Abordagens de baixo custo como por exemplo, *phishing* e *password spraying*, exploram falhas humanas e senhas frágeis. Por outro lado, ameaças mais complexas, como os *advanced persistent attacks* e *zero-day exploits*, exigem conhecimento e recursos, sendo frequentemente direcionados a sistemas críticos. Várias organizações dependem de centros de *helpdesk* de cibersegurança, sejam internos ou subcontratados, para gerir incidentes. Contudo, estes centros enfrentam dificuldades em responder de forma eficaz devido à sobrecarga de dados e escassez de operadores qualificados.

Esta dissertação aborda a carência de operadores especializados e o elevado volume de incidentes enfrentados pelas operações de *helpdesk*, propondo o desenvolvimento de um assistente de gestão de *tickets* para apoiar operadores humanos na resolução destes incidentes. A plataforma incorpora um sistema de recomendação que, dependendo do contexto, identifica o par operador-procedimento mais rápido para cada incidente, melhorando continuamente com cada tratamento realizado. Para garantir a privacidade dos dados, o sistema de recomendação é treinado com dados artificiais gerados por um gerador de dados personalizado. Além disso, a tese explora a possibilidade de melhorar este assistente com funcionalidades de *automated machine learning* para prever *tickets* futuros. Esta funcionalidade pode auxiliar gestores na antecipação da carga de trabalho e na adaptação proativa das suas equipas de segurança.

O desenvolvimento desta ferramenta é realizada em colaboração com a empresa de cibersegurança S21sec, que disponibilizou estruturas e taxonomias de dados históricos anonimizados relacionados com o tratamento de incidentes. No entanto, devido à ausência de informação granular sobre a resolução dos incidentes e informação relacionada com o conjunto de dados partilhados e à necessidade de preservar a privacidade, as técnicas de geração de dados sintéticos tornam-se essenciais. O gerador implementado cria dados artificiais que replicam distribuições semelhantes às dos dados reais, ao mesmo tempo que simula processos do mundo real, incluindo priorização de *tickets*, agendamento e tratamento.

O gerador de dados artificiais é avaliado pela sua eficiência em replicar as características de conjuntos de dados do mundo real, através de métricas de similaridade como *Hellinger distance* e *Kullback-Leibler divergence*. Além disso, são explorados vários cenários de agendamento de *tickets*, variando o número de operadores e a sua distribuição em três turnos de trabalho. Os resultados demonstram que esta ferramenta consegue replicar *tickets* com diferentes distribuições e durações de tratamento, derivadas dos dados reais. Adicionalmente, este gerador permite a simulação de

operações reais de *helpdesk*, proporcionando uma base sólida para explorar diversos contextos operacionais sem comprometer a privacidade dos mesmos. A análise do agendamento de *tickets* mostra consistentemente que cenários caracterizados por um grande desequilíbrio nos turnos e com menos operadores levam a tempos de espera mais longos e a mais *tickets* agendados para tratamento posterior.

O sistema de recomendação é testado em duas vertentes: escalabilidade e impacto no tratamento de *tickets*. A primeira fase utiliza diversos conjuntos de dados de teste com tamanhos e números de operadores diferentes, analisados com métricas como o tempo médio de recomendação e memória consumida. Por outro lado, o impacto no tratamento de *tickets* é examinado considerando as melhorias nos tempos de espera dos *tickets* antes de serem atribuídos a um operador e o tempo de resposta necessário para a sua resolução, utilizando diferentes graus de aceitação das recomendações. Os resultados indicam que o número de operadores que o sistema de recomendação utiliza tem um impacto ligeiramente superior na sua escalabilidade em comparação com o número de *tickets* de teste. Ambas as características mostram um padrão de crescimento linear semelhante em relação às métricas referidas, sendo que o número de operadores apresenta um maior declive. A integração deste sistema de recomendação no tratamento de *tickets* sintéticos de teste reduziu o tempo médio de resposta entre 37.9% e 45.1% e o tempo médio de espera entre 62.2% e 63.2%, assumindo que os operadores aceitam sempre as recomendações. Com taxas de aceitação de recomendações variáveis, o tempo médio de espera mantém-se constante, enquanto a melhoria no tempo de resposta varia entre 0.4% e 11.7%.

A potencial aplicação de *automated machine learning* para a análise preditiva é explorada através de um estudo onde as decisões relativamente à dimensionalidade das equipas recomendadas pelo sistema são comparadas com os resultados esperados. Este estudo avalia o sistema com base na precisão das previsões sobre sobre o número de incidentes futuros e na na eficácia com que sugere ajustes no tamanho das equipas, com base no desempenho histórico dos operators e nas previsões recolhidas. Entre as distribuições de conjuntos de dados testadas, os modelos treinados com dados de três anos superaram os treinados com dados de quatro anos, apresentando um erro médio mais baixo ao utilizar dados reais sobre a frequência de *tickets* ao longo do ano. Relativamente às recomendações de dimensionalidade da equipa, incluindo a contratação ou despedimento de operadores, a ferramenta baseada em *automated machine learning* propôs frequentemente decisões estreitamente alinhadas com as esperadas no mesmo periodo.

Coletivamente, estes resultados mostram que as ferramentas propostas podem otimizar os fluxos de trabalho de tratamento de *tickets* em aplicações do mundo real, levando a um uso mais eficiente dos recursos e a uma redução dos atrasos operacionais. Além disso, a sua capacidade de simular operações do mundo real sem comprometer a privacidade permite que os centros de operações de segurança possam testar vários cenários e aperfeiçar as suas estratégias.

# Acknowledgements

Along with the completion of this thesis, I have been fortunate to receive the guidance and support of many colleagues, friends, and family, without whom this accomplishment would not have been possible.

First, I express my gratitude to my advisors, Daniel Castro Silva and Mikel Uriate, for their constant support during this work. Your expertise, mentorship, and friendship were crucial to the success of this work. Daniel Castro Silva, your continuous and detailed feedback and time dedicated to this project helped me push forward and believe in my capabilities. Mikel Uriarte, your encouragement and dedication have also been invaluable to this research and me. I am confident that this thesis couldn't have reached this stage without both of you.

I am also grateful to all my friends for their support over the last few years, offering strength and encouragement during challenging times. In particular, I would like to thank André Lago, Carlos Granja, Luís Duarte, and Edgar Ramos for always being there when I needed it most.

To my incredible family, your love and words have always been the source to improve myself. To my parents and sister, thank you for always believing in me and all the support throughout this journey. To my wife, your love and endless patience have been essential in helping me close this chapter of my life. Thank you for always standing at my side and for reminding me what truly matters in life.

Lastly, I dedicate this thesis to Lila, Luna, and Manuel Nogueira, whose memory inspires and guides me. Your presence in my life was a gift, and I will never forget you.

This work has been partially supported by the SONAE IM LAB@FEUP, under a research Ph.D. project funded by S21sec.

Leonardo Ferreira

*"When you look at yourself from a universal standpoint, something inside always reminds or informs you that there are bigger and better things to worry about"*

Albert Einstein

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| ABM | Agent-Based Modeling |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| ASD | Average Selected with Distance |
| AUC | Area Under the Curve |
| AutoML | Automated Machine Learning |
| BN | Bayesian Network |
| BPR | Bayesian Personalized Ranking |
| BS | Bayesian Similarity |
| BSSB-RS | Behavioural Social Stream-Based Recommender System |
| CARS | Context-Aware Recommender System |
| CB | Content-Based |
| CF | Collaborative Filtering |
| CGAN | Conditional Generative Adversarial Network |
| CI | Computational Intelligence |
| CLCREC | Contrastive Learning-based Cold-start Recommender System |
| CME | Cluster with Minimum Error |
| CNN | Convolutional Neural Network |
| CoNCARS | Convolutional Context-Aware Recommender System |
| CPE | Common Productor Enumerator |
| CPU | Central Processing Unit |
| CSSR | Cloud Services Security Recommender system |
| CSV | Comma Separeted Value |
| DABEL | Data Generation Based on Complexity per Classes |
| DARPA | Defense Advanced Research Projects Agency |
| DART | Data Analytics Research Team |
| DCARS | Deep Context-Aware Recommendation System |
| DCG | Discounted Cumulative Gain |
| DDOS | Distributed Denial-Of-Service |
| DL | Deep Learning |
| DoD | Department of Defense |
| DPGAN | Differentially Private Generative Adversarial Network |

| | |
|---|---|
| DSRS | Decision Support and Recommender Systems |
| DSS | Decision Support System |
| DT | Decision Tree |
| DTW | Dynamic Time Wrapping |
| ECAM | Explicit Context-Aware Model |
| ENCM | Explicit Neural Context Model |
| FGCR | Fused graph Context-Aware Recommendation System |
| FM | Fuzzy Matching |
| FSG | Fully Synthetic Generation |
| GA | Genetic Algorithm |
| GAN | Generative Adversarial Network |
| GGLR | Graph based Geographical Latent Representation |
| GNN | Graph Neural Network |
| HCAM | Hierarchical Context-Aware Model |
| HR | Hit Rate |
| IDS | Intrusion Detection System |
| IDSG | Information discovery and analysis systems Data and Scenario Generator |
| IDSS | Intelligent Decision Support System |
| IoT | Internet of Things |
| IP | Internet Protocol |
| IT | Information Technology |
| IUI | Intelligent User Interface |
| K-NN | K-Nearest Neighbour |
| K-S | Kolmogorov Smirnov |
| KB | Knowledge Based |
| KCR | Kernel based Context aware Recommendation system |
| KLD | Kullback-Leibler Divergence |
| LDA | Latent Dirichlet Allocation |
| LLMs | Large Language Models |
| LNCM | Latent Neural Context Model |
| LR | Linear Regression |
| LSTM | Long-Short Term Memory |
| MAE | Mean Average Error |
| MAMO | Memory Augmented Meta-Optimization model |
| MAP | Mean Average Precision |
| MAPE | Mean Absolute Percentage Error |
| MCLDA | Multiple Correspondence Latent Dirichlet Allocation |
| MF | Matrix Factorization |
| MIA | Membership Inference Attack |
| MICE | Multiple Imputation by Chained Equations |
| MIT | Massachusetts Institute of Technology |
| MJLDA | Multiple Joint Latent Dirichlet Allocation |
| ML | Machine Learning |
| MLFQ | Multi Level Feedback Queue |
| MRR | Mean Reciprocal Rank |

| | |
|---|---|
| NAS | Neural network Architecture Search |
| NCF | Neural Collaborative Filtering |
| NeuMF | Neural Matrix Factorization |
| NLP | Natural Language Processing |
| NN | Neural Network |
| NVD | National Vulnerability Database |
| OODA | Observe Orient Decide Act |
| OSN | Online Social Network |
| OWA | Ordered Weighted Averaging |
| PATE | Private Aggregation of Teacher Ensembles |
| PATE GAN | Private Aggregation of Teacher Ensembles Generative Adversarial Network |
| PCA | Principal Component Analysis |
| PDF | Probabilistic Density Function |
| PHP | Hypertext Preprocessor |
| PMF | Probabilistic Matrix Factorization |
| PMML | Predictive Model Markup Language |
| POI | Point-Of-Interest |
| PSG | Partially Synthetic Generation |
| RF | Random Forest |
| RMSE | Root Mean Square Error |
| RNN | Recurrent Neural Network |
| ROAR | RecOmmendation Acceptance Rate |
| RS | Recommender System |
| RS-LOD | Recommender System with Linked Open Data |
| SASREC | Self Attention Sequential RECommendation model |
| SBN | Social Benchmarking Network |
| SCOAL | Simultaneous Co-clustering And Learning |
| SDDL | Synthetic Data Definition Language |
| SIEM | Security Information Event Management |
| SLA | Service Level Agreement |
| SLCM | Sequential Latent Context Model |
| SNOOKER | dataSet geNeratOr fOr helpdesK sERvices |
| SOC | Security Operation Center |
| SoCo | Social Contextual Recommendation system |
| SP | Sequential Pattern |
| SPM | Sequential Pattern Mining |
| SQL | Structured Query Language |
| SVM | Support Vector Machine |
| TADS | Task Assignment Decision Support |
| TCPOFA | Trust based Context aware Post Filtering Approach |
| TEL | Technology Enhanced Learning |
| TF | Tensor Factorization |
| TGAN | Temporal Generative Adversarial Network |
| TGSREC | Temporal Graph Sequential Recommender System |
| TRTS | Train on Real data, Test on Synthetic |
| TSGAN | Time Series Generative Adversarial Network |
| TUANDROMD | Tezpur University ANDROid Malware Dataset |
| TUMALWD | Tezpur University Windows MALware Dataset |
| TXT | Text |

| | |
|---|---|
| UCAM | Unstructured Context-Aware Model |
| UK | United Kingdom |
| USA | United States of America |
| VAE | Variational Autoencoder |
| VPN | Virtual Private Network |
| VULINTEL | Vulnerability IntelliSensor |
| WGAN | Wasserstein Generative Adversarial Network |
| XAI | Explainable Artificial Intelligence |
| XLSX | Excel open XML Spreadsheet |
| XML | Extensible Markup Language |

# Chapter 1

# Introduction

In an era dominated by digitalization and a growing data stream, businesses rely heavily on technology to improve their operations and productivity [Wilburn and Wilburn, 2018]. As companies take advantage of the opportunities of the digital landscape, they also encounter various challenges, including software glitches, hardware malfunctions, insufficient user training, and other issues. Although these limitations are concerning, many can be effectively managed through the intervention of helpdesk centers [Invensis, 2022]. These centers are crucial in troubleshooting problems, accessing critical resources, and other customer support operations [Zendesk, 2024]. Depending on their goals and the available resources, companies can establish internal helpdesk teams or outsource these services to external providers, ensuring flexible and agile support [Steiman, 2021]. When cyberattacks and other security threats target organizations, specialized helpdesk units known as security operator centers (SOCs) perform real-time monitoring, detection, and response to cyber crimes, thus fortifying the defense of businesses and protecting their assets [Vielberth et al., 2020].

## 1.1 Context and Motivation

Despite their troubleshooting and query resolution capabilities, helpdesk performance has been affected by the escalation in both frequency and complexity of cyber alarms [St. John and Swanston, 2024]. These threats, often manifested as security incidents within the client's premises, include various actions orchestrated by malicious actors, including data loss, system disruptions, and other consequences. Hackers' most prevalent tactics are malware, phishing, distributed denial-of-service (DDoS), and other attack vectors. Recently, statistics showed a growth of 128.7% in ransomware victims [Cybernews, 2023], and Microsoft reported that approximately 1700 DDoS attacks were addressed daily [Microsoft Threat Intelligence, 2023]. In response to these challenges, SOCs have emerged as a strategic evolution from traditional helpdesk centers.

These teams consist of individuals with a diverse set of skills tasked with the monitoring and protection of digital assets. However, the efficiency of their procedures is aggravated by the lack of qualified experts, which may result in the oversight of potential critical issues [McCann, 2023].

Gartner projections indicate that by 2025, more than half of cybersecurity incidents will occur due to the lack of skilled professionals and human error [Gartner, 2023].

The industry must recognize that relying solely on intelligent detection methods is insufficient to mitigate intrusions' repercussions effectively. Prioritizing the optimization of security operations is vital to minimize the response time to potential attacks and conserve resources. To achieve this goal, companies should provide precise instructions to SOC teams for skill refinement and develop intelligent decision support tools. Such frameworks can help security teams focus on critical alerts while automating the treatment of less important events. According to the Online Trust Alliance (OTA) Cyber Incident and Breach Trends Report in 2019, 93% of breaches could have been prevented through simple measures, such as regular software updates, email authentication to block fake email messages, and training people to recognize phishing attacks and other intrusions [OTA, 2019]. The remaining 7% represents more dangerous and complex situations requiring meticulous analysis.

SOC operators are also affected by the overwhelming volume of incoming alerts, resulting in delays in incident resolution and reduced efficiency. Although complete automation could improve operator productivity, it could raise concerns about incident adaptability, automated alert validation, and strategic decision-making [Vielberth et al., 2020, Nogueira, 2022]. With more automation, SOC operators can focus more on treating critical incidents. In this context, the implementation decision support tools such as a recommender system (RS) could accelerate incident prioritization, response actions, and overall incident treatment efficiency. These tools could generate personalized recommendations through historical data analysis, allowing helpdesks and SOCs to act with greater precision and efficacy. For example, in cybersecurity, these systems could discern the most optimal operator or action for a particular ticket [Ferreira et al., 2023].

One company that provides cybersecurity helpdesk support is S21sec[1]. This entity offers relevant knowledge that is instrumental in assessing the components introduced in this work. Their SOC teams utilize detection and treatment tools to manage incidents effectively. Furthermore, analysis and orchestration of incident treatment workflow within this organization is done using ticketing systems that monitor and track issues raised by customers and teams, among other frameworks [Inabo, 2023]. These platforms facilitate the prioritization, assignment, and resolution of incidents, enhancing operational efficiency. Section 2.5 provides a comprehensive overview of a SOC workflow confirmed by the good practices implemented by this company, elucidating how the research results may alleviate specific processes.

## 1.2 Hypotheses

Taking into account the context, we intend to improve the efficiency of SOCs through the usage of intelligent systems. For this purpose, the hypotheses of this study are as follows.

[1]Website available at https://www.s21sec.com/

**H1:** An RS for helpdesk operations can improve SOC's response time while treating the tickets with the fastest available procedure.

**H2:** A synthetic data generator focused on helpdesk services can help evaluate an RS in this context.

To validate our hypotheses, it is crucial to answer specific research questions. These answers will provide the necessary evidence to support our claims.

**RQ1** What approaches can build synthetic data based on real data for ticketing systems? Depending on the context and application, different methods for synthetic data generation may be employed. Some authors employ data distribution approaches, while others use more complex strategies involving ML and generative adversarial networks (GANs). Section 3.1.2 provides an overview of the research on this topic, while section 3.2.2.2 details the approach selected for the development of the proposed synthetic dataset generator;

**RQ2** Can we generate synthetic data that closely mimics real data regarding distribution and business logic? Synthetic data can replicate characteristics of real data, including specific feature distributions and other meta-features (evaluated in section 3.3.1). By understanding the underlying processes relevant in this context, we can integrate business logic operations composed of rules, conditions, and workflows that translate real-world helpdesk operations. This feature ensures that the generation and treatment of artificial data are coherent with real-data patterns and activities. Section 2.4 illustrates the tasks performed by these systems;

**RQ3** What is the impact of different team sizes on ticket handling, measured by the percentage of delayed tickets and average wait time? The dataset generation process, the number of operators in each team, and their allocation across distinct work shifts can influence incident resolution over time. Section 3.3.2 studies different data distributions to analyze delays in ticket processing and other time-related factors;

**RQ4** What is the influence of historical data and different team sizes on the computational performance of an RS and its scalability? The volume of historical data and different team sizes can greatly impact the performance of RSs, particularly regarding scalability and temporal efficiency. Section 4.4.1 provides a comprehensive analysis of how two dimensions of historical data volume influence the scalability and computational efficiency of CROSS: horizontally, based on the number of operators (team size), and temporally, based on the size of the time window applied;

**RQ5** How does the operator's acceptance level impact the performance of a context-aware RS? Strict and flexible SOC operators can influence the performance of an RS, as it may require additional computation to generate suggestions accepted by operators. In the absence of real-life operators, a probability factor can help simulate the acceptance or refusal of recommendations in a simulated environment. Section 4.4.2 investigates the impact of different likelihoods on CROSS performance;

**RQ6** How does the introduction of a context-aware RS impact the wait and response time to tickets? As discussed in chapter 4.1, RSs were developed to assist users across a wide

range of tasks. In this context, as detailed in section 4.4.2, the impact of RSs is evaluated through metrics related to waiting and response time and tickets solved after refusing $n$ times. Together with the previous research question, this helps validate different aspects of the proposed context-aware RS.

## 1.3 Methodology

To address the shortage of publicly available helpdesk datasets while exploring synthetic data generation, and provide empirical support for **RQ1**, we develop an artificial dataset generator. This generator is capable of producing realistic datasets that are applicable across multiple domains and contexts. The design of this system involves a comprehensive analysis of real data patterns and established helpdesk operations (ticket assignment, ticket prioritization, ticket scheduling, and ticket escalation). By examining these operational processes, we build datasets that accurately reflect real data distributions and treatment workflows, thereby addressing **RQ3**. The validity of the generated datasets is ensured through statistical measures that allow a comparison of feature distributions with real-world data. Furthermore, by experimenting with different team configurations within the dataset generator, we derive thoughts related to **RQ2**.

Considering the constraints of the application scenario, we present a context-aware RS designed to optimize decision-making processes in operational environments by identifying the fastest operator-procedure pairs. This system is trained and validated using the artificial generator described above. To address **RQ4**, we test different proximity thresholds, which facilitate the evaluation of performance metrics, such as average recommendation time and memory spent. In addition, the impact of the RS is analyzed in conjunction with various ROAR values. This assessment uses multiple evaluation measures, including average wait time and the difference between estimated and recommended action duration, providing robust evidence for **RQ5** and **RQ6**.

The capabilities of the RS could be improved in various ways, such as support for determining optimal SOC team sizes. This could be achieved through the integration of tools like AutoML, which can forecast potential ticket volumes. Although AutoML is not explored in the RS, its potential is demonstrated through a case study. This study highlights AutoML's effectiveness in predicting fluctuations in ticket influx and optimizing team configuration.

## 1.4 Contributions

This thesis presents relevant contributions that may assist the research community and companies to be more prepared for current and future needs, such as:

1. A synthetic dataset generator for helpdesk services (SNOOKER) that creates generic and security-oriented tickets with diverse characteristics. This tool customizes the treatment teams and simulates real-world operations, such as ticket treatment, prioritization, scheduling, and escalation;

Beyond generating geographically and temporally correlated data, SNOOKER's most relevant innovation lies in its adherence to business rule logic, particularly in the ticket processing phases. SNOOKER ensures that synthetic data aligns with constraints and workflows by mirroring real-world approaches. Most of the domain expertise embedded in these processes was provided by S21sec, further enhancing the realism of the generated data. Moreover, SNOOKER avoids privacy problems since it only relies on the meta-features of the real data, ensuring that no sensitive data is used.

An article about this tool called "SNOOKER: A Dataset Generator for Helpdesk Services" was published in the Journal of Intelligent Information Systems [Ferreira et al., 2024];

2. A context-aware RS (CROSS) that suggests the fastest operator-procedure pair, allowing SOC operators to spend less time evaluating ticket details and focus more on critical tasks;

   From the application perspective, CROSS introduces incident trait analysis and procedure crossover to handle incidents representing cold-start scenarios. In such cases, CROSS searches similar incident categories by analyzing feature distributions, utilizing a similarity matrix to pinpoint the most analogous groups. Once similar incidents are identified, CROSS builds a new action by performing a crossover of the steps from actions taken in similar categories. Furthermore, this system shows adaptability when the operator rejects a recommended treatment procedure by searching for the fastest alternative actions closely aligned with the operator's emulated behaviour.

   This work was documented in two articles: the first, titled "Recommender Systems in Cybersecurity", was published in the Journal Knowledge and Information Systems [Ferreira et al., 2023]; the second, "CROSS: Context-aware Recommender System", is currently under review;

3. A case study showcasing the integration of SNOOKER and CROSS to support decision-making regarding team size. For this purpose, a DSS supported by AutoML (CRYSTAL) is proposed to predict incident instances during specific periods based on historical patterns. With the predictions processed, it could optimize the team's size through different staff decisions: hiring or dismissing $n$ professionals or remaining without changes.

   From the application standpoint, this system analyzes the performance of each team and, based on the forecasts obtained, proposes informed decisions regarding personnel adjustments, including the potential dismissal or necessity for additional experts. This posture may prevent and anticipate potential complications during future ticket treatment.

   An article based on this case study called "CRYSTAL: Decision Support System with AutoML for Workforce Management" has been produced and is currently under review.

The combination of these three systems has the potential to enhance the performance of helpdesk centers by optimizing SOC teams, enabling them to resolve tickets using the fastest method available. Figure 1.1 illustrates the main contributions of this thesis: SNOOKER, CROSS, and CRYSTAL.

In this thesis, SNOOKER aggregates data from various sources (user, real dataset, and configuration file) to generate generic and domain-specific datasets that emulate real-world helpdesk

processes. These synthetic datasets can be used to train CROSS, allowing it to recommend the fastest operator-treatment procedure for each open ticket. CRYSTAL also receives and processes data from SNOOKER and employs automated machine learning for predictive analysis. Based on the forecast analysis, CRYSTAL suggests potential team size adjustments to SOC managers.

All these modules are detailed in the following chapters.

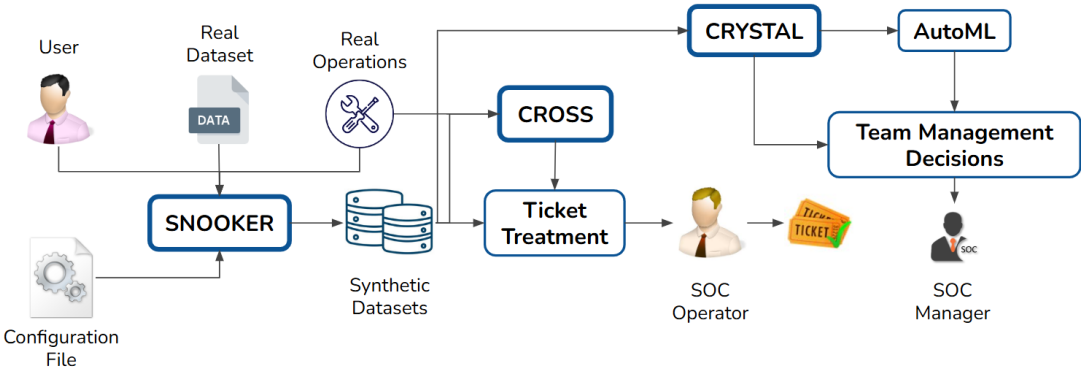



Figure 1.1: Thesis Worfklow

## 1.5 Document Structure

The remainder of this document is organized as follows:

- Chapter 2 provides the context and motivation of this thesis. It explores the importance of SOCs in protecting digital ecosystems against cyber intrusions and using ticketing systems for efficient incident management. Additionally, it provides a brief overview of SOC architecture based on literature confronted with practical good practices of a relevant market player such as S21sec;
- Chapter 3 inspects the concept behind synthetic data generation and introduces SNOOKER, an artificial dataset generator designed to create generic and domain-based tickets with diverse characteristics. This chapter starts by covering various strategies employed by the research community, from methods that generate artificial data partially to those that produce completely synthetic datasets. It then details the generation process of SNOOKER and its features, including tickets and incident types, among other properties. This tool also simulates various helpdesk processes, such as ticket assignment, prioritization, scheduling, and escalation. A series of experiments is conducted to evaluate the performance of SNOOKER in replicating real-world patterns, among other aspects;
- Chapter 4 discusses recommender systems, a central output of this thesis, and proposes CROSS, a context-aware RS. Besides examining the challenges, available techniques, and similarity measures in RSs, this chapter explains how CROSS identifies the fastest operator and treatment procedure combination for resolving each ticket. The resolution of novel incident types and recommendation adaptation upon operator refusal are two interesting

features conveyed by this module. Additionally, various scenarios and metrics are used to test CROSS's scalability and its impact on ticket treatment;

- Chapter 5 provides a brief overview of workforce management and how it can be enhanced with AutoML. Moreover, it introduces CRYSTAL, a DSS that supports team size management. Based on AutoML forecasting and historical data, this system makes team decisions, such as hiring or dismissing personnel, while determining the optimal number of operators required. Various experiments assess CRYSTAL's performance in providing management decisions under different team compositions;
- Chapter 6 discusses the conclusions reached from this research while addressing the research questions shown in Chapter 1. It also explores the potential integration of these tools within S21sec's infrastructure. Additionally, this chapter addresses some limitations of this study and identifies several research lines for future improvement.

# Chapter 2

# Security Operation Centers

SOCs are centralized units focused on real-time monitoring, detection, analysis, and response to cybersecurity threats [Scapicchio et al., 2024]. Comprising IT and operation technology (OT) experts equipped with advanced frameworks, such as intrusion detection systems (IDSs), digital forensic and incident response tools, security information event management (SIEM), security orchestration, automation and response (SOAR), case investigation and ticket handling systems, and other threat intelligence tools, SOCs continuously monitor network traffic, system logs, and security alerts to shield businesses against various malicious activities, including malware, phishing, ransomware, insider threats, and advanced persistent attacks (APTs) [Mughal, 2022]. When detecting an incident or vulnerability, SOCs investigate, inspect the threat's impact, and formulate response strategies to mitigate potential damage. Ultimately, SOCs ensure asset protection, business continuity, regulatory compliance, cost-effectiveness, proactive threat response, and improve customer trust, incident response, and risk management [Scapicchio et al., 2024].

This chapter establishes the foundation of this thesis, briefly introducing key concepts of SOCs and ticketing systems. Moreover, it provides a comprehensive overview corroborated by the practices at the S21sec company.

## 2.1 Roles of SOCs

According to Mughal, SOC teams are structured with experts fulfilling distinct roles: manager, security analyst, incident responder, threat intelligence operator, vulnerability management specialist, penetration tester, forensic analyst, and user and entity behaviour operators [Mughal, 2022].

Table 2.1: Roles related to SOCs (adapted from [Mughal, 2022])

| Role | Description |
|---|---|
| SOC Manager | Supervises security operations and manages SOC team, including training, hiring, and performance evaluation. Implements security protocols and collaborates with stakeholders to continuously improve SOC's capabilities. |

Roles related to SOCs

| Role | Description |
|---|---|
| Security Analyst | Analyzes security alerts to identify potential breaches, enriches them with additional data, and determines if they are false positives. |
| Incident Responder | Manages, responds, and contains security threats strategically to prevent further damage. |
| Threat Intelligence Operator | Studies emerging adversaries and shares relevant knowledge with SOC teams to enhance threat detection and response capabilities, promoting a proactive approach over a reactive one. |
| Vulnerability Management Specialist | Monitors, prioritizes and mitigates vulnerabilities in the organization's infrastructure, including scheduling patching activities. |
| Penetration Tester | Also known as pen tester or ethical hacker, simulates authorized real-world cyberattacks to assess an organization's security defenses and comprehend its vulnerabilities. |
| Forensic Analyst | Investigates security incidents, preserves evidence for vulnerability mitigation, and supports legal proceedings. |
| User and Entity Behaviour Operators | Monitor user and entity activities within system infrastructure using ML and behavioural analytics. |

The research community built alternative classifications regarding SOC composition. For example, Vielberth et al. proposed a structure consisting of a SOC manager, triage specialist, incident responder, and threat hunter [Vielberth et al., 2020]. Similarly, Scapicchio et al. advocated a more general configuration comprising a SOC manager, security engineers, security analysts, and threat hunters [Scapicchio et al., 2024]. Miloslavskaya followed a different view and classified SOCs according to counteraction capabilities, deployment scenarios, objectives, correlation method, implementation variant, and ownership [Miloslavskaya, 2016]. These categorizations highlight the absence of a universal accepted SOC taxonomy since organizational context and security objectives require different approaches.

SOC teams rely on security playbooks and established vulnerability guidelines to manage cybersecurity incidents effectively. Security playbooks, such as MITTRE ATT&CK [Corporation, 2018], SANS [Robin Dickerson, 2004], NIST [National Institute of Standards and Technology, 2024] offer response procedures tailored to address various types of cyberattacks. These playbooks provide in-depth details about the processes for detecting, analyzing, containing, and fixing incidents. Furthermore, these teams can keep track of existing vulnerabilities through databases, including Common Vulnerabilities and Exposures (CVE[1]), allowing SOC operators to stay informed about

[1]Website available at https://www.cve.org/

the latest threats. With these resources, SOC teams can enhance their incident response capabilities and mitigate risks more effectively.

Figure 2.1 represents a typical SOC architecture, showcasing the interactions between key components. The illustrated software elements are regularly updated and, in some cases, migrated to alternative platforms. These updates are designed to enhance the precision of incident detection, accelerate incident analysis and prioritization, and automate incident response treatment.

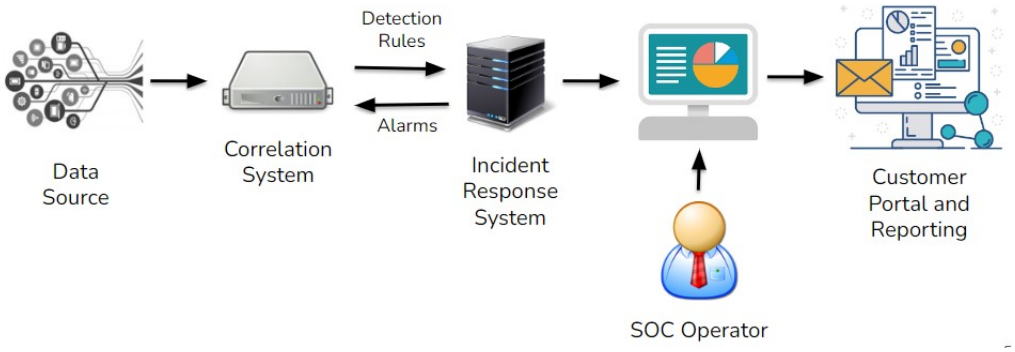



Figure 2.1: SOC Architecture Example

## 2.2 Challenges

Given the dynamic and complex nature of SOC operations, human and technological challenges can decrease SOC operators' performance and, consequently, incident analysis and treatment.

A prominent issue within SOCs is the scarcity of cybersecurity professionals and difficulty in retaining them [Agyepong et al., 2020a, Vielberth et al., 2020, Mughal, 2022]. The continuous evolution of cyberattacks necessitates that SOC operators possess the skills to perform incident response effectively. However, the supply of such experts falls short of the escalating demand, resulting in a talent shortage in this area. Another reason for this phenomenon is the high professional turnover rate [Agyepong et al., 2020a]. While skilled analysts can be recruited, retaining them is another challenge due to intense and high-pressure environments, financial opportunities, limited promotion and development opportunities, and other reasons [ISACA, 2022]. The proper training procedures can also help the efficiency of SOCs.

The escalating volume and complexity of security incidents pose significant obstacles to IT teams' monitoring, analysis, and response capabilities [Agyepong et al., 2020a]. This incident surge can overwhelm SOC operators, inducing alert fatigue and compromising their security posture. A critical consequence of this overload is the increased likelihood of classification errors, such as false negatives (FN) and false positives (FP) [Kokulu et al., 2019, Agyepong et al., 2020a, Vielberth et al., 2020]. FN cases arise from security threats going undetected or inaccurately labeled as benign events. Conversely, FP alarms emerge when SOC operators misclassify events as malicious, leading to unnecessary analysis and diverting resources from more critical tasks [Hossain and Islam, 2023]. Furthermore, the integration of diverse data sources alongside data heterogeneity and

increasingly sophisticated attacks, significantly increases the complexity of the SOC operator's workload [Vielberth et al., 2020, Mughal, 2022].

The increased complexity in IT environments, including cloud, incident management processes, and other infrastructures, makes it difficult for SOC operators to maintain situational awareness during ticket treatment [Kokulu et al., 2019, Agyepong et al., 2020a, Vielberth et al., 2020, Mughal, 2022]. The modern nature of IT structures requires SOC operators to navigate through complex networks and diverse systems while mitigating incidents effectively. Managing various technologies can be a time-consuming task, requiring continuous maintenance and configuration to keep SOC operations running smoothly. Furthermore, due to budget constraints or other limitations, some frameworks may suffer from poor usability, reducing the effectiveness of SOCs. Compliance with regulation protocols such as the General Data Protection Regulation (GDPR) can also create privacy questions that may hinder the speed and effectiveness of incident analysis and response [Vielberth et al., 2020, Mughal, 2022].

The collaboration between operators, analysts, and security teams is essential to ensure a suitable treatment response [Kokulu et al., 2019, Agyepong et al., 2020a, Vielberth et al., 2020]. However, in several cases, isolated information and time constraints hinder the coordination between them, leading to delays in incident response.

The lack of comprehensive technical metrics hinders the evaluation of the SOC operator's performance [Kokulu et al., 2019, Agyepong et al., 2020a, Vielberth et al., 2020, Agyepong et al., 2020b, Forsberg and Frantti, 2023]. Existing metrics focus on operational activities, which prove insufficient for SOC performance evaluation. Without standard metrics, organizations may find it difficult to accurately assess the efficiency of their SOC teams and monitor their progress over time.

The satisfaction within SOCs is affected by the monotony of routine tasks, which include filtering through multiple alerts, manually correlating data, and conducting incident investigations [Agyepong et al., 2020a, Vielberth et al., 2020]. These repetitive processes contribute to increased fatigue and burnout among operators and analysts, leading to delays in incident response times. Failure to manage these processes efficiently can overwhelm them, leading to missed or delayed responses to potential security incidents. Moreover, the high pressure in SOC environments poses a risk for professionals' mental health [Agyepong et al., 2020a]. The constant demand for rapid response and vigilance against an array of threats, coupled with the stress of alert investigation and other taxing tasks, increases the likelihood of burnout among SOC personnel. Some of the mentioned problems (alert fatigue and increased workload) could be addressed with automation functionalities, enabling SOC operators and analysts to concentrate on addressing critical incidents. Nonetheless, the complexity of many scenarios and treatment protocols raises questions regarding the automation level implemented [Vielberth et al., 2020]. Integration of SIEMs and SOARs alongside visualization tools has been instrumental in alleviating some of the burden associated with tasks, such as alert analysis, incident response, and other processes [Schinagl et al., 2015]. While SIEMs aggregate and analyze security data within a company, speeding up threat detection and security event management, SOARs help automate repetitive operations, improve collaboration between security teams, and streamline incident treatment.

## 2.3 Evaluation

The scientific community explored several metrics pertinent to assessing SOC operator performance. In 2020, Agyepong et al. explored various challenges and metrics that affect SOCs [Agyepong et al., 2020a]. They categorize twelve challenge groups that impact SOC efficiency, including high alert volumes, false positives, complex attack scenarios, analyst burnout, excessive workload, and other issues. The authors also discuss quantitative and qualitative metrics for assessing SOC performance. Quantitative metrics are objective and measurable, and include indicators such as the time to detect (TTD) an incident, the average time taken to respond (ATTR), and the time spent creating each ticket, the number of alerts analyzed/not analyzed, the number of tickets closed per day and within a timeframe, the time spent on operations by an analyst, and time spent on each ticket. On the other hand, qualitative measures can be subjective and cover aspects related to the quality of incident reports and analysis, the level of competency and experience of an analyst, and the use of success stories. Moreover, they present a mapping framework correlating SOC challenges with these metrics. For example, the TTD of an incident can be linked to the complexity of an attack. However, specific challenges, such as burnout, remain difficult to quantify and are unrelated to existing measures. The same researchers interviewed various operators and, based on the main functions performed by SOC analysts and metrics detailed previously, proposed a framework for evaluating SOC analysts, which shares certain features with the NIST taxonomy. [Agyepong et al., 2020b]. Through these interviews, they further defined eleven global SOC function groups and five metrics, including the number of incidents raised, the time taken to detect and respond to incidents, the number of incidents closed, and the quality of analysis and incident reporting. Their framework could support a more comprehensive assessment of SOC operators' performance while also aiding in developing new performance metrics. In 2023, Agyepong et al. developed SOC-AAM, a SOC analyst assessment method [Agyepong et al., 2023]. This approach assigns weights to the SOC functions established earlier, informed by the feedback from a Delphi panel of SOC professionals and the principles of the analytic hierarchy process, to measure the performance of SOC analysts. The SOC-AAM was evaluated in two organizations, using the method adoption model [Paz et al., 2015] and five research questions. Their analysis suggests that SOC managers can aggregate and quantify an operator's performance in a structured manner through the proposed technique.

Vielberth et al. surveyed the SOC's domain, covering its components, challenges, and existing performance measures [Vielberth et al., 2020]. To improve the security posture and clarify SOC structures, the authors examined SOC architecture using the People, Processes, Technology, Governance, and Compliance (PPTGC) framework, detailing each group's roles, processes, challenges, and metrics. They further explored factors influencing operational decisions, such as cost, time, size, industry sector, regulations, privacy, etc. Evaluation metrics, although they occasionally may overlap, are categorized into four clusters: general, people, technical, and governance and compliance. General metrics include coverage and performance measures. The people category focuses on metrics involving SOC operator activities and workflows, such as workload and incidents resolved per shift. Technical metrics assess threat (e.g., threat actor attribution), vulnerability (e.g., exposure

or severity), risk (e.g., risk posture and system risk), alert (e.g., alert frequency and criticality), incident (e.g., incident volume and attack success rate), and resilience (e.g., response time and attack noise level) metrics. Lastly, the governance and compliance metrics assess adherence to regularity requirements (e.g., policy violation volume and maturity level). The authors identify various challenges associated with SOC activity, including rising complexity, integration of domain knowledge, privacy regulations, and other limitations. They advocate for clearly understanding PPTGC components as essential to SOC efficiency.

Forsberg and Frantti also conducted an extensive literature review on performance metrics for SOC evaluation and introduced a theoretical framework for developing technical metrics in this field [Forsberg and Frantti, 2023]. Their study begins with a comprehensive examination of the state-of-the-art, presenting thirty distinct operational performance measures. Given that the prior work described focuses most on operational metrics, the authors devised a methodology to refine existing technical and non-technical metrics and to introduce novel metrics to address inherent limitations. For a metric to be considered valid, it must meet several criteria: be clearly defined (including its ownership), have a well-defined objective, be independent, be explainable, be aligned with success factors, and meet quality criteria, such as accuracy, measurability, meaningfulness, and usability. The authors validated their approach by measuring specific metric groups within a SOC, including the distribution of detections among the unified kill chain (framework that models all stages of cyberattacks from reconnaissance to post-exploitation activities), the number of verifiable monitoring (tracks rules that can be validated to confirm detection features), the distribution of detections by source (compares existing detection methodologies with custom analytics developed by SOCs), and the technical accuracy of the analysis (quantitative view of operator accuracy across MITRE ATT&CK tactics). The results of the experiments indicated that three out of the four developed metrics were considered valid with the proposed metric creation framework. The only exception was the metric related to the number of verifiable monitoring rules, which was considered invalid due to its high bias. Some metrics, such as the distribution of detections among the unified kill chain, showed partial successful outcomes, particularly when relying on the original detection source, which was partially available. Although the authors propose this methodology for SOC operational assessment, they note that these experiments primarily focused on evaluating the detection functions within SOCs.

It is important to note that SOC performance metrics are not limited to those discussed by the scientific community. Other entities employ additional measures, such as [Dempsey et al., 2011, Wickramasinghe, 2023, Cassetto, 2024, Automation, 2023, Security, 2024]:

- Dwell time – total time an intruder remains undetected within a system before being detected and handled;
- Incident cost – each incident can result in direct and indirect costs. The first concept is associated with time and resources spent on detection and response, while indirect costs pertain to customer impact, reputational damage, and other effects;
- Incident Escalation rate – the proportion of tickets escalated from lower-tier teams to more expert ones;

- Incident Recurrence rate – measures the likelihood of repeated incidents of the same type;

Compared to the previous analysis, SOC-CMM provides a more exhaustive and technical list of metrics categorized into five groups: business, people, process, technology, and services [van Os, 2007]. Each metric is detailed in five levels: type, description, level, target, and differentiation. Some of these metrics are used in section 4.4.2 for evaluating the proposed RS, while additional metrics are suggested for further analysis.

## 2.4 Ticketing systems

A ticketing system is a helpdesk software designed to manage customer issues from submission to treatment [Inabo, 2023]. These frameworks operate as central repositories, tracking diverse issues, inquiries, and service requests from operators and clients. Typically, ticketing systems feature a customizable interface to streamline the monitoring, management, and prioritization of ticket-related activities. For example, they convert each customer request into a ticket containing essential details such as the client's details, issue description, status, and other pertinent properties, and assign a team/operator to perform diagnosis and treatment. Various organizations employ ticketing systems in their workflow due to [Masternak, 2023, ITarian, 2023]:

- Centralized services – the management and tracking of all tickets within a single location. This arrangement expedites the support teams' accessibility and processing of tickets. Moreover, it maintains comprehensive records documenting all interactions between customers and support analysts, including customer history, previous requests, and other analytics;
- Standard customer support – utilize service level agreements (SLA) to keep customer expectations met by the service providers and stakeholders. It ensures accountability and transparency in the decision-support task;
- Improved communication and collaboration – can unify customer-related issues into a single thread/tool, preventing the loss of pertinent data and the existence of multiple unnecessary channels. Furthermore, it allows greater cooperation between support agents and, consequently, more effective solutions to tickets;
- Enhanced performance tracking – allow reporting and analytic capabilities, enabling companies to track diverse performance metrics, including first contact resolution, SLA violations, backlog tickets, and user satisfaction [Engine, 2023];
- Streamlined operations – can automate various operations, such as ticket creation, assignment to operators based on predefined rules, prioritization considering their criticality, and escalation of complex tickets requiring in-depth investigation. Moreover, they can minimize manual effort (for example, handling repetitive alerts) and minimize response times (for instance, history logs may reduce analysis time).

One of the primary operations performed by these systems is the management of unresolved incidents. The ticket treatment includes ticket prioritization, assignment, scheduling, escalation tasks,

and other operations. The assigned operator addresses the ticket through established workbooks and guidelines, such as those in section 2.1.

The ticket prioritization determines the order in which tickets are addressed and can be tackled with distinct techniques [GRIDLEX, 2023]:

- Round robin – tickets are distributed sequentially in a cyclical manner;
- Skill-based – tickets are assigned to SOC operators according to their level of expertise;
- Priority-based – tickets are fixed based on their priority level, with methods involving the classification by severity (critical, high, medium, and low), the impact of the issue (prioritizing those affecting a larger number of clients or structures), and adherence to SLAs established with customers [GRIDLEX, 2024a];
- First come, first served – tickets are processed by the order they arrive at the system.

Regarding ticket scheduling, tickets can be closed immediately, delayed for later analysis and closure, placed on hold status, escalated to more experienced teams, or reassigned to a different SOC operator after a predefined period. This process depends on various aspects, including the operator's availability, work shifts defined, implemented SLAs, and other critical factors. In a hierarchical workflow, tickets can be escalated to other teams when their complexity surpasses the expertise of the current team, when input from multiple departments is required, or when there is a risk of breaching the defined SLAs [GRIDLEX, 2024b].

Although ticketing systems can perform these tasks and others (root cause analysis, workflow automation, and incident feedback, among others), they are insufficient. SOC teams require additional tools to improve threat detection, analysis, and response. Moreover, the efficiency of ticketing systems hinges on their configuration and requirements, which are defined alongside the implemented SOC methodologies.

## 2.5 S21sec

Established in 2000, S21sec is a cybersecurity company in Spain, Portugal, Mexico, and the UK, specializing in delivering security services across diverse sectors, such as hospitality and leisure, telecommunications, oil and gas, and finance. Four years later, the company was acquired by SONAE IM, the technology investment sector of the SONAE group. In 2018, SONAE IM merged S21sec with Nextel S.A. to strengthen and expand its cybersecurity capabilities.[2] Four years later, S21sec moved to the portfolio of another major technological group, Thales.[3]

Their services comprise threat management, breach detection and resolution, incident response and recovery capabilities within organizations, prevention techniques, employee education on cybersecurity best practices, and alignment of business goals with cybersecurity principles.

[2]Information available at https://brpx.com/sonae-im-strengthens-its-cybersecurity-position-in-iberia/

[3]Website available at https://www.thalesgroup.com/en/worldwide/group/press_release/thales-signs-agreement-sonae-investment-management-acquire-s21sec-and

Considering the SOC taxonomy delineated in Table 2.1, S21sec teams integrate various SOC managers (5%), security analysts (65%), with 27% being incident responders, threat intelligence operators (18%), penetration testers (7%), and forensic analysts (5%). The hierarchical structure of their security teams comprises three tiers: the first provides a preliminary response, the intermediary conducts investigations into the case, and the final devises resolutions for the origin of the security problems. Figure 2.2 illustrates the simplified workflow of S21sec's infrastructure, comprising four systems: SIEM, SOAR, Ticketing, and Monitoring. Once incidents are collected from the client premises, the SIEM module handles them through event ingestion, normalization, aggregation, and correlation. The SOAR system then manages incident ingestion from SIEM, performing event enrichment, triage, and automating workflow. When alerts are escalated to tickets, the Ticketing component oversees the incident response, ticketing tracking, and customer communication. Meanwhile, the Monitoring module examines the incidents and evaluates the performance of SOC operators.

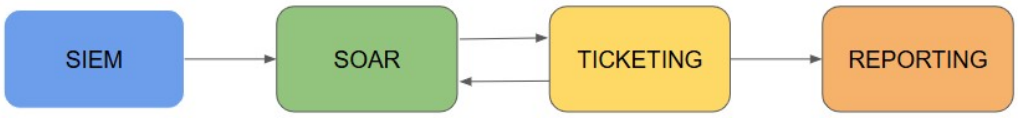



Figure 2.2: Schematic incident flow at S21Sec's SOC

From S21sec's perspective, the quality of service and performance of SOCs is assessed across different team tiers using two essential groups of metrics: operational metrics and service metrics. The first category focuses on incident treatment efficiency, comprising two metrics: response time and mitigation incident resolution. The response time refers to the duration taken for the initial assessment of an incident, which includes preliminary analysis, triage, impact assessment, escalation, and other processes. In contrast, incident mitigation resolution measures the total time spent resolving the incident from its identification to closure. Both metrics closely correspond to TTD and ATTR metrics described in section 2.3, respectively. Service metrics involve a broader range of performance indicators, including false positive rate, customer-raised incident rate, error rate reduction, customer satisfaction, cost per incident, technical skill and leadership evolution, and talent generation and retention.

# Chapter 3

# SNOOKER: Dataset Generator for Helpdesk Services

The scarcity of datasets with high-quality features poses several issues for the industry and research community. Their absence limits the training and validation of intelligent systems that may help improve many tasks (automation and decision-making). The helpdesk area is no exception since there are datasets that have few entries or are short in information. For example, in this domain, many datasets lack data on the actions taken by IT operators to handle cybersecurity tickets. Nonetheless, some techniques like data augmentation [Dyk and Meng, 2001] and synthetic data generation [Chen et al., 2021] have been used to improve these limitations. Unlike data augmentation, which improves existing data through different types of transformations, synthetic data generation can create new data. The ability to build data that preserves privacy while making it impossible to distinguish from real data motivated our study into synthetic data generation.

This chapter starts by exploring synthetic data generation, detailing available techniques and the literature review performed in this field. It then focuses on the development of SNOOKER (dataSet geNeratOr fOr helpdesK sERvices), a generator that addresses various challenges in ticket generation and treatment [Ferreira et al., 2024]. Its primary goal is to generate generic and domain-specific datasets that emulate real-world behaviours, such as ticket treatment, prioritization, and scheduling, while preserving privacy and data utility. Finally, the chapter presents and discusses the evaluation of SNOOKER, comparing the generated datasets to a real-world dataset and analyzing the impact of team dimensionality.

## 3.1 Synthetic Data Generation

Synthetic data is defined as information built by a computer program that tries to copy real-world events or behaviors [Andrews, 1996]. It is a topic that has shown some promise in areas like security [Skopik et al., 2014], healthcare [Tucker et al., 2020], robotics [Martinez-Gonzalez et al., 2020], and other domains. Table 3.1 provides some knowledge about the benefits and limitations of this type of data.

Table 3.1: Advantages and Limitations of artificial data

| **Advantages [Dilmegani, 2020, Joshi, 2019]** | **Limitations [Hao et al., 2024]** |
|---|---|
| Allows confidentiality and the simulation of unforeseen scenarios | Raises ethical and trust concerns |
| The time required to build artificial data is significantly lower compared to real data acquisition | Can be influenced by biases in the generation models, leading to a failure in replicating real-world events |
| Eases the development and prototyping with fewer constraints (time and money) | May suffer from insufficient noise level and over-smoothing |
| Can improve models by training them with all the conditions needed | Can neglect temporal and dynamic features |

Synthetic data can be classified as partially or fully synthetic [Dilmegani, 2020]. In the first category, only some information is changed to prevent a high risk of disclosure. These datasets typically maintain the original statistical relationships and patterns while modifying sensitive information to reduce the identification risk. On the other hand, fully synthetic datasets are generated independently and do not include any real values from the source dataset. These aim to replicate the statistical properties and complexity of real data without including any identifiable data. In some cases, real-world and synthetic data are combined to take advantage of fully and partially synthetic data (hybrid approach).

Analyzing real dataset properties, also known as meta-features, may assist synthetic data generation in building similar but artificial datasets without compromising data privacy.

### 3.1.1 Traditional Methods

Artificial data can be created with three different methodologies: Distribution-based, Agent-based modeling (ABM), and Deep Learning (DL) models [Kantarci, 2020].

Distribution-based techniques allow the generation of artificial data with or without knowledge about the distribution relative to a real dataset. When original data is unavailable, analysts can still generate synthetic data with any distribution (normal, exponential, and others) if they understand the desired distribution. The quality and utility depend on the knowledge of the distribution applied. If real-world data is available, users can acquire synthetic data by discovering the best-fit distributions for the given original data. The Monte Carlo method can also be applied if the distribution and its parameters are known [Miok et al., 2019]. While these techniques are known for their simplicity, interpretability, and capacity to maintain the statistical properties of the original data, they may struggle to capture data relationships and hidden patterns within data [Hao et al., 2024]. As a result, the generated data may lack realism and may not be ideal for tasks such as model training and system evaluation.

ABM techniques build synthetic data by simulating the behaviour of autonomous decision-making entities known as agents [Bonabeau, 2002]. These entities interact with one another,

exhibiting varied behaviours and patterns. Understanding these interactions enables the development of models that produce data that, although it appears random, serves different purposes such as capturing agent heterogeneity, representing uncertainty, and the appearance of novel patterns. Despite its flexibility, ability to enhance realism, and capacity to capture emergent phenomena, ABM faces challenges in social, political, and economic areas and requires high computational resources. Bonabeau studied applications of ABMs, including flow simulation, organizational simulation, market simulation, diffusion simulation, and some techniques, highlighting their benefits and drawbacks [Bonabeau, 2002]. Despite their ability to model highly adaptable heterogeneous environments, ABM approaches are computationally expensive and sensitive to the choice of parameters.

Regarding DL, the main techniques available are Generative Adversarial Networks (GAN's) [Torres, 2018], Variational Auto Encoders (VAE's) [Wan et al., 2017], and Autoregressive models [Chen et al., 2018b].

GAN models consist of two NNs: a generator and a discriminator [Goodfellow et al., 2014]. As shown in Fig.3.1, the generator creates artificial data based on the original dataset, while the discriminator assesses and distinguishes between real data and the generator's output. This process may progressively refine both networks, leading to the generation of more realistic datasets.

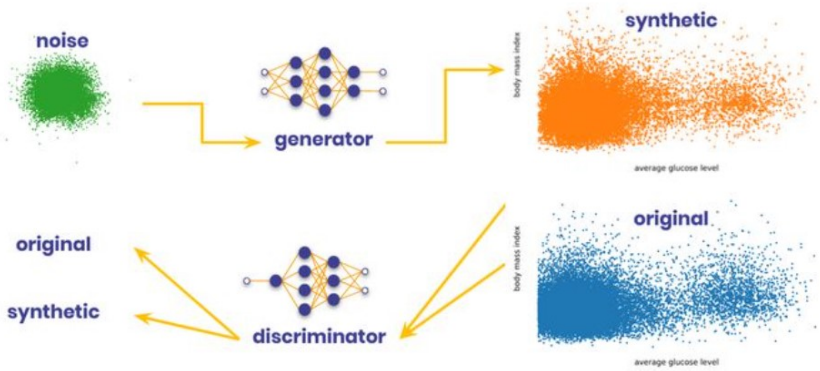



Figure 3.1: Synthetic Generation with GAN [Wehmeyer, 2021]

In contrast to GANs, VAEs utilize encoders and decoders to build new data. The encoder compresses the real data into a latent space, which can then be transformed into its original space with decoder structures. The objective is to reconstruct the input as closely as possible while minimizing the reconstruction error. Figure 3.2 presents this process more clearly.

GAN models often generate results with higher fidelity than VAE techniques; however, they are more difficult to train and require careful tuning, particularly on identifying the optimal point to stop the training [Wehmeyer, 2021]. Depending on the type and quantity of accessible real data, a hybrid approach combining the latent representation learning of VAEs with the realistic data generation of GANS may provide the best performance.

Overall, DL methodologies can manage large and complex datasets, do not require explicit feature engineering, and can be generalized by uncovering hidden patterns. Nonetheless, they face

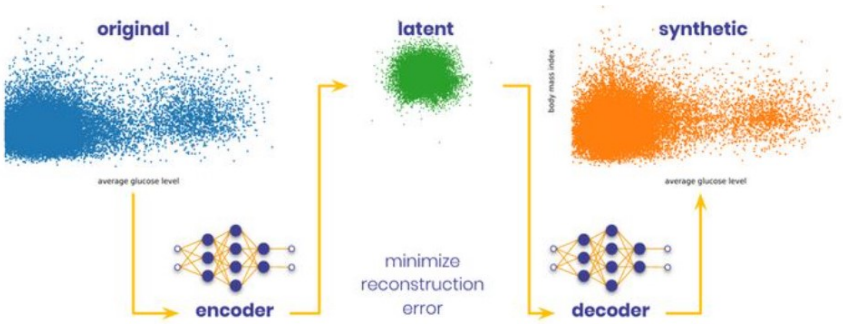



Figure 3.2: Synthetic Generation with VAE [Wehmeyer, 2021]

several challenges, such as high computational demands, limited interpretability, and require vast sample sizes to reach accurate results [Ahmed et al., 2023].

Meta-features are unique properties derived from dataset attributes, usually employed in ML and data analysis [Reif et al., 2014]. Such properties detail a dataset's structure, complexity, and statistical characteristics. While primarily used in meta-learning, meta-features serve multiple purposes, including summarizing dataset benchmarks for ML experiments, identifying similar datasets for a particular case, and guiding the generation of artificial datasets [Rivolli et al., 2022]. In synthetic data generation, knowledge of real meta-features can create data that is closely aligned with real data. These properties can be grouped into the following categories [Lemos et al., 2022]:

- General – also known as simple measures, is information easily collected without requiring substantial computational resources. It includes the number of attributes (nrAttr), the number of classes (nrClass), the total missing number (nrMissing), and other characteristics;
- Statistical – extracts statistical information from the dataset, such as average (mean), standard deviation (sd), minimum (min), maximum (max), and many other properties;
- Information Theoretic – captures the amount of information and its complexity with joint entropy (jointEnt), the entropy of target values (classEnt), and other measures;
- Model-based – data retrieved from predictive models, usually DTs and represented by the number of leaves (leaves) and number of distinct paths (leavesBranch), among other metrics;
- Landmarking – employs the performance of learning methods to characterize datasets, including metrics like best node (bestNone), naive bayes (naiveBayes), etc;
- Other – includes clustering, complexity, data distribution, case-based, concept-based, structural information, and time-based features.

According to Lemos et al., all meta-features above, except the last class, have the potential of being applied in synthetic data generation [Lemos et al., 2022]. Although the author highlights its applicability in this context, he acknowledges that changing one meta-feature may inadvertently affect others. The reason for discarding this category is the possibility of making the generation resource-intensive and time-consuming.

### 3.1.2 Related Work

This section addresses the related work of synthetic data generation regarding existing surveys in the area, cybersecurity data generation, usage of GANs and VAEs, evaluation methodologies, privacy preservation, multidimensional data, applications, meta-learning, and data augmentation strategies (in this order).

In 2015, Filchenkov and Pendryak studied the optimal set of meta-features for feature selection algorithms recommendation [Filchenkov and Pendryak, 2015]. Their analysis evaluated various classification methods, including naive Bayes, C4.5, PART, BN, and IB3 (nearest neighbour), using distinct metrics and meta-features sets (statistical, general, and others). The authors also introduced an approach for meta-feature engineering based on the classifier model and its best parameterization.

Following two years, Surendra and Mohan examined many data generation methods for privacy-preserving data publishing to safeguard personal data confidentiality [Surendra and Mohan, 2017]. For each work reviewed, they described the data generation type (partial or complete), the primary method employed, and its associated limitations.

A year later, Vanschoren provided an in-depth overview of meta-learning, along with a taxonomy based on the type of meta-data [Vanschoren, 2018]. Such categorization considers learning from model evaluations, learning from curves, task properties, and prior models.

In 2019, Popić et al. surveyed a few dataset generators designed to mimic actual data for testing scenarios [Popić et al., 2019]. They detailed the architecture of these generators, outlining their advantages, limitations, and usages. Bellovin et al. explored synthetic data as a solution to privacy concerns [Bellovin et al., 2019]. They discussed ML strategies for creating synthetic data, such as NNs, RNNs, and GANs. The paper also includes a detailed case study that elucidates the process of generating and evaluating synthetic data while addressing the legal implications of its usage. The authors argue that synthetic data offers greater efficiency than traditional anonymization approaches and advocate for necessary adjustments to privacy regulations. Nikolenko examined the impact of synthetic data on computer vision applications and other fields [Nikolenko, 2019]. The author studied synthetic datasets frequently employed in computer vision, simulated environments, and applications in neural programming, bioinformatics, and natural language processing (NLP) domains. Furthermore, he scrutinized approaches for improving synthetic data generation through GANs, addressed the synthetic-to-real adaptation challenge, and discussed privacy concerns related to synthetic data.

In 2021, Faez et al. investigated DL-based graph generators (DCGs), their methods, applications, datasets, and metrics employed [Faez et al., 2021]. They classified existing approaches into five groups: autoregressive-based, autoencoder-based, reinforcement learning-based, adversarial, and flow-based. Dankar and Ibrahim researched various synthetic data generation settings using four synthetic data generators: Synthetic Data Vault, Data Synthesizer, Synthpop parametric, and non-parametric [Dankar and Ibrahim, 2021]. They specifically assessed how data preprocessing affects the utility of artificial data, whether synthetic datasets require tuning when building supervised ML

models, and whether the results obtained from these models can enhance the synthetic generators. Additionally, they inspected the application of the metric propensity score in predicting the accuracy of the ML models derived from the synthetic data. Chen et al. debated the application of synthetic data in medical and healthcare solutions [Chen et al., 2021]. While acknowledging its potential to enhance the diversity and resilience of AI models, the authors delineated several challenges associated with its integration in these critical domains. These challenges include the lack of standardized clinical quality measures and evaluation metrics, the interoperability limitations in electronic medical records, and the need for more regularization.

One year later, Rivoli et al. provided an overview of the meta-learning paradigm, including an extensive discussion of a taxonomy comprising diverse meta-features [Rivolli et al., 2022]. This taxonomy categorizes the meta-features into various groups: simple, statistical, information-theoretic, model-based, landmarking, and clustering, as well as distance- and time-related measures. The authors presented each class's meta-features, properties, and application scenarios. Moreover, the study presents various dataset characterization tools, including the Matlab Statistics Toolbox [MathWorks, 2005], OpenML [Vanschoren et al., 2014], among others. Lemos et al. inspected the meta-features described earlier by Rivoli et al. and diverse methodologies for producing artificial data [Lemos et al., 2022]. The authors employed the Python Meta Feature Extractor (PyFME[1]) to extract meta-features from datasets generated by an artificial generator called SNOOKER [Ferreira et al., 2024]. His study also examines the role of meta-features in enhancing this tool's generative process.

Two years ago, Gonzales et al. conducted a thorough review of the use of synthetic health data across seven applications, such as simulation and prediction research, hypothesis, methods, algorithm testing, epidemiological investigations, and public health inquiries [Gonzales et al., 2023]. They also researched the role of synthetic data in IT development, education initiatives, and training endeavors. This review highlighted the public release of datasets and linking data. A detailed explication of diverse synthetic health datasets and their properties was also given.

In 2011, O'Shaughnessy and Gray developed a dataset generator to improve information discovery and network security analysis infrastructures [O'Shaughnessy and Gray, 2011]. This tool produces datasets that simulate different attack patterns usually used by intruders, like Denial of Service (DoS), Probing, and SQL injection attacks using algorithms coded by the authors. For its validation, a C4.5 DT was trained on datasets containing SQL injection and probing signatures (no DoS was available) to check its performance. The results showed high accuracy for SQL injection attacks and acceptable values for the probing dataset.

Three years later, Boggs et al. introduced a framework called Wind Tunnel, which focused on synthesizing standard and attack data for PHP web applications in multiple layers[Boggs et al., 2014]. This framework enables independent testing of security controls against specific attacks. Standard traffic is created with Firefox via Selenium[2] and the attack data with Metasploit[3]. Wind

[1]Documentation available at https://pymfe.readthedocs.io/en/latest/index.html
[2]Documentation available at https://www.selenium.dev/
[3]Documentation available at https://www.metasploit.com/

Tunnel also includes a monitoring tool that supervises multiple layers, such as transmission control protocol packets, database queries, and host system calls. It incorporates usernames, passwords, English text, and images as data sources to reach more realism. The authors employed nineteen sensors across database layers, file access, system call, and network to evaluate Wind Tunnel's performance, using eleven datasets: nine generated by Wind Tunnel and two private datasets (the Columbia University Computer Science department web server and ISCX 2012 Intrusion Detection Dataset [Shiravi et al., 2012]). These materials were used to assess how each security control behaved under varying dataset conditions, evaluate synthetic datasets' predictive ability for production datasets, compare the accuracy of security control across layers, and analyze the security controls' combined performance. According to the authors, Wind Tunnel successfully created synthetic datasets suitable for testing security controls.

In 2017, Pendleton and Xu built a dataset generator to improve system call (syscalls) datasets by incorporating structural and contextual information [Pendleton and Xu, 2017]. Existing intrusion detection datasets often suffer from structural, typological, and quantitative issues that limit the development of host IDSs. This generator addresses these challenges by delivering structured syscalls. This tool comprises a syscall collector, a containerized execution environment, and a dispatcher to facilitate the generation task.

Two years later, Kim and Kim proposed a cyber threat dataset generation system called CTIMiner to promote Cyber Threat Intelligence [Kim and Kim, 2019]. This tool collects data from published cyber reports and malware repositories, storing the information in a database. CTIMiner operates in four main phases: parsing indicators of compromise, collecting analysis data, filtering and storing data, and processing results from the initial phases. To evaluate CTIMiner, the authors demonstrated an application where correlation analysis was conducted using a correlation graph of a malware information-sharing platform.

In 2020, Meyer-Berg et al. addressed the dataset shortage for training IoT anomaly detection systems in a secure environment [Meyer-Berg et al., 2020]. Their proposed method has a network model and user interface that facilitate the training and evaluation of various anomaly detection methods. It was designed to be extensible to custom protocols and smart devices, with simulation capabilities. The authors tested their approach in a smart home environment under two scenarios: DoS and DDoS attacks, and the other focusing on integrity tampering, using K-means for anomaly detection. The results demonstrated high adaptability in data analysis algorithms while generating large datasets and accelerating algorithm training. Borah et al. created two frameworks for generating malware datasets: the TUMALWD (Tezpur University Windows MALware Dataset) and TUANDROMD (Tezpur University ANDROid Malware Dataset) for Windows and Android, respectively [Borah et al., 2020]. Both generators employ data collection and storage, data analysis, and feature engineering. The TUMALWD focuses on host-level and network features, while TUANDROMD emphasizes permission and API-based features. For evaluation, five supervised algorithms were used for benchmarking: random forest (RF), extra tree, adaptive boosting, extreme gradient boosting, and gradient boosting. Both datasets achieved similar high classification accuracy, ranging from 96% to 99%. Le et al. analyzed the role of GAN models in cybersecurity

through their technique, CyberGAN, which generates domain-specific datasets [Le et al., 2020]. CyberGAN trains its discriminator using normal and malicious operations, while the generator creates cybersecurity datasets, like firewall activity, that mimic real data. Both components are fully connected and optimized using a custom loss function with leaky ReLU activation. The authors evaluated CyberGAN at different levels, assessing the fidelity and utility of the generated datasets with network structure and KLD metrics and their suitability for ML tasks. The accuracy assessment in this experiment utilized three datasets: firewall activity, credit card fraud dataset, and CTU-13 botnet dataset. Furthermore, they compared CyberGAN's architecture to that of CTGAN. Although CyberGAN showed promising results in these cases, the authors noted that it may not fully guarantee privacy protection, as GANs learn from the underlying structure and distribution of the training data.

In 2017, Creswell et al. presented an overview of GANs, discussing their architectures, such as fully connected, convolutional, conditional, inference models, and adversarial autoencoders [Creswell et al., 2018]. They studied challenges associated with the training phase and explored GAN applications.

A year later, Torres studied the role of GANs in data generation and proposed a method consisting of six steps: schema detection (understanding data types), pattern analysis (examining correlations between variables), feature engineering (data normalization), ML model training (creating and training a model with random original data), data production (generating synthetic data) and feature reverse-engineering (data de-normalization) [Torres, 2018]. The study introduced a Wasserstein Conditional GAN using feed-forward NN for both the generator and discriminator, and a synthetic generator based on recurrent NNs for evaluation purposes. Both techniques were tested on training time and data generation times, pattern preservation (Euclidean distance), data distribution preservation, and quality of the generated text in three different datasets. The results indicated that the GAN-based solution excelled in data correlation and text quality preservation. Still, it lagged behind the non-GAN-based solution in data generation time, training time, and preservation of the statistical distribution of the original data. Xu and Veeramachaneni implemented a synthetic tabular data generator called TGAN, designed for health and educational domains [Xu and Veeramachaneni, 2018]. TGAN employs a series of transformations to handle various data types (numerical, categorical, time, and others) and distributions (multimodal, long tail, and others). It uses an LSTM as the generator and a Multi-Layer Perceptron as the discriminator. The evaluation involved three different datasets (Census-Income, KDD Cup 1999, and the Covertype) to determine whether the synthetic datasets generated by TGAN are suitable for ML purposes. The performance of TGAN was compared against three generation methods: Gaussian Copula, BN for each column, and Bayesian Network between columns. The results demonstrated that TGAN outperformed Gaussian Copula and BN for individual columns in terms of ML performance and data correlation.

In 2019, Lu et al. developed a synthetic generator using GANs to assess the utility and risk of re-identification of the generated output [Lu et al., 2019]. Their GAN approach learns the data distribution from original datasets and builds artificial data via iterative sampling. For the evaluation, they measured data utility by analyzing correlation matrices and conducting ML tasks,

while the risk of re-identification was tested using differential testing and record linkage techniques. The evaluation covered four datasets, using numerical, categorical, adult, and mixed values (the last two combine numerical and categorical data). Although the results indicated robust data utility, no discussion is provided regarding the outcome related to the risk of re-identification evaluation (only tables are provided). Yoon et al. introduced time-series GAN (TimeGAN), an approach capable of synthesizing realistic datasets while preserving temporal dynamics [Yoon et al., 2019a]. Such a tool comprises embedding and recovery functions, the generator, and the discriminator. The first two elements provide the latent space, and the remaining components work in the same created space. The latent dynamics of both the real and synthetic data are controlled with a supervised loss to learn temporal relationships. TimeGAN was compared with other solutions [Esteban et al., 2017, Mogren, 2016, Sutskever et al., 2011, Goyal et al., 2016, van den Oord et al., 2016, Donahue et al., 2018] in terms of visualization, discriminative score, and predictive score in four types of time-series data (sines, stocks, energy, and events). The results demonstrated that TimeGAN outperforms the techniques mentioned previously in the metrics. Baowaly et al. proposed an improved version of an existing medical GAN (medGAN) based on boundary-seeking to generate more realistic datasets [Baowaly et al., 2019]. While both approaches use autoencoders and minibatch averaging, they differ in the GAN used, with boundary GAN offering improved training stability and convergence. Their performance was evaluated on two datasets: MIMIN-III[4] and NHIRD[5]. The evaluation methods included dimension-wise average and K-S tests, association rule mining, and predictive analysis. In all scenarios, medBGAN outperformed the classic medGAN. In the same year, Vega-Márquez et al. studied the utility of the datasets generated from conditional GAN (CGAN) with credit card fraud detection datasets [Vega-Márquez et al., 2019]. Unlike standard GANs, CGANs incorporate additional information in the generator and discriminator. The evaluation focused on the correlation between original and synthetic data and the accuracy achieved by XGBoost. For the correlation analysis, metrics such as covariance, Pearson's, and Spearman's correlation coefficient were employed, revealing values close to 0, indicating a minimal relationship between variables. The performance of XGBoost was assessed based on accuracy, F1-score, and AUC for both real and synthetic datasets. The results showed identical performance, suggesting that CGAN possesses utility comparable to actual data.

In 2020, Smith and Smith invented TSGAN (Time-Series GAN) to create synthetic time-series data through two Wasserstein GAN (WGAN) [Smith and Smith, 2020]. The first WGAN transforms random latent vectors into spectrogram images, and the second builds time-series data from these images. TSGAN was tested on seventy datasets specialized in time-series information (from sensors, motion, and other sources) with metrics like Fréchet Inception Score and accuracy in three scenarios: TSTR (train on synthetic data, test on real), TRTS (train on real data, test on synthetic), and TRTR (training on real and testing on real). Regarding the inception score, the results showed that TSGAN surpassed WGAN in fifty-six datasets. In the remaining datasets, the authors assumed that WGAN outperformed TSGAN due to poor generation performance in both

[4]Dataset available at https://physionet.org/content/mimiciii/1.4/
[5]Dataset available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6367203/

approaches. Regarding accuracy, TSGAN achieved higher accuracy in TRTS and TSTR (11% and 15%, respectively) but had lower accuracy in TRTR (4%).

In 2021, Little et al. investigated the performance of tabular GANs (TGANs) against statistical synthetic techniques using utility and disclosure risk metrics [Little et al., 2021]. They compared statistical techniques, specifically the Synthpop [Nowok et al., 2016] and DataSynthesizer [Ping et al., 2017] with GAN-based methods, such as the CTGAN [Xu et al., 2019b] and TableGAN [Park et al., 2018]. The evaluation focused on the disclosure risk through the targeted correct attribution probability metric and utility measures like the confidence interval overlaps, the ratio of estimates, and the propensity mean-squared error. This study utilized a 1991 Individual Sample of Anonymised Records of the British Census [6]. Among the methods mentioned, Synthpop achieved the highest utility but also the greatest risk of exposing sensitive data, while TableGAN exhibited the lowest risk, albeit with poor-quality data. DataSynthesizer and CTGAN produced intermediate results relative to the other methods. Mozo et al. proposed an optimized version of a WGAN to build synthetic datasets that simulate crypto mining attacks and regular traffic, increasing the availability of datasets and avoiding any privacy issues from real datasets [Mozo et al., 2021]. They introduced a heuristic to optimize the stop criteria of GAN training with the best-performing GAN settings. In addition to the F1-score, they also presented two novel metrics (L distance and Jaccard coefficient) to assess the quality of datasets generated compared to authentic data. For evaluation, the authors experimented with several scenarios where an ML classifier was trained with real data from the Mouseworld lab [Pastor et al., 2018], data collected from a simple mean-based generator, and synthetic datasets from standard and improved WGANs. The results showed that the proposed method could build suitable datasets (similar to real datasets) for ML-based crypto-mining detectors while avoiding privacy violations.

GANs have found relevant applications in medical image synthesis. For instance, Guibas et al. developed a dual GAN, consisting of stage-I GAN and stage-II GAN, to create medical images [Guibas et al., 2017]. Han et al. designed a GAN-based method to generate multi-sequence brain magnetic resonance images [Han et al., 2018]. Islam and Zhang published a method employing GANs to produce synthetic brain pet images under three stages of Alzheimer's disease: control, mild cognitive impairment, and Alzheimer's [Islam and Zhang, 2020]. Other areas have been exploring GANs to synthesize images [Abady et al., 2020].

While exploring VAEs for dataset generation is less extensive than that of GANs, they remain a valid approach. In 2017, Wan et al. implemented a synthetic generation method using VAEs to address imbalanced learning [Wan et al., 2017]. VAEs are effective for complex data distribution since they make weak assumptions about the data and can produce balanced datasets. The approach starts by sampling from a latent variable and then generates new samples based on the conditional distribution of these values, similar to the original data. The authors evaluated their approach using the MNIST and affNIST datasets, comparing it against SMOTE and ADASYN [Han et al., 2005, He et al., 2008]. They assessed performance using precision, recall, specificity, f1_score,

[6] Dataset available at https://www.ons.gov.uk/census/2011census/2011censusdata/censusdata18011991

and g_mean. On the MNIST dataset, their method outperformed all the competitors in all metrics. However, on the affNIST dataset, the traditional SMOTE achieved better precision and g_mean, while the proposed VAE method excelled in the remaining measures.

Two years ago, Razghandi et al. combined VAE with GAN to generate different synthetic time-series data for smart home environments [Razghandi et al., 2022]. This integration allowed the creation of samples without prior data analysis, effectively addressing the issue of GAN mode collapse, a common problem during GAN's training phase. They compared the output from VAE-GAN and the vanilla GAN against the real data distribution for evaluation. The metrics used were the KLD, maximum mean discrepancy, Wasserstein distance, and statistical parameters (near-peak load, near-base load, and others). The results favored VAE-GAN compared to the traditional GAN.

In 2018, Dandekar et al. compared partial synthetic generation (PSG) aganst fully synthetic generation (FSG) [Dandekar et al., 2018]. According to the authors, PSG preserves most of the original data's structure and utility by changing only sensitive data (for example, age). In contrast, FSG prioritizes privacy by synthesizing the entire dataset. They assessed these methodologies across various classifiers, including LR, DT, RF, and NN generators, focusing on the utility and disclosure risk. The utility was gauged using KLD and overlap, while the disclosure risk was measured using the Reiter and Drechsler algorithm [Drechsler and Reiter, 2008]. Experimental findings revealed that DT provided the highest data utility, with KLD values of 0.53 for real datasets and 0.58 for artificial ones. In contrast, NNs showed the lowest disclosure risk, with a value of 0.03. Furthermore, the runtime analysis revealed different efficiencies among the classifiers: DT recorded runtimes of 0.048 seconds for PSG and 0.533 seconds for FSG, while LR spent 0.040 seconds for PSG and 0.068 seconds for FSG.

In 2020, Goncalves et al. addressed synthetic data generation for electronic medical records using various evaluation criteria [Goncalves et al., 2020]. They examined probabilistic models, classification-based imputation, and GAN methods, analyzing them through KLD, pairwise correlation difference, log-cluster, support coverage, and cross-classification metrics. The findings revealed that no singular method outperformed the metrics mentioned. The authors also highlighted that GANs could not generate realistic electronic health samples. Other methods explored the concept of artificial data in the healthcare domain using a data-driven approach [Buczak et al., 2010], naive bayes [Aviñó et al., 2018], and GANs [Arvanitis et al., 2022].

In 2016, Nettleton designed a graph-based dataset generator to simulate artificial online social networks (OSNs) [Nettleton, 2016]. The architecture proposed comprises three main modules: topology definition, data definition, and data population. In the initial phase, the generator employs RMat [Chakrabarti et al., 2004] to build non-overlapping topologies or uses real OSN overlapping topologies. Communities are then identified with the Louvain approach [Blondel et al., 2008]. The data definition phase delineates the target profiles, the target profile for each profile, and the target distribution for each attribute value. The last stage manages parameter configurations, allocates data profiles to seed nodes, and facilitates data propagation from seeds to neighbours. The authors analyzed their solution by comparing synthetic topologies created by RMat with real-world OSN topologies. Their experiments highlighted that artificial topologies yielded cleaner assignments

than real OSNs. Ayala-Rivera et al. introduced COCOA, a domain-based framework capable of synthesizing microdata for evaluating anonymization methods [Ayala-Rivera et al., 2016]. COCOA replicates a dataset's aggregated statistics by utilizing frequency distributions of multiple attributes as probability weights [Ayala-Rivera et al., 2013]. Guided by user input, it employs three attribute generators: distribution-based, attribute-based, and a combination of both. Performance is assessed using dataset diversity (via PCA), utility (generalized information loss), and resource usage (CPU and memory). Experimental results show that the COCOA exhibits diversity in dataset generation and varied generalized information loss values across various datasets, indicating its proficiency for testing. Moreover, the results underline COCOA's efficiency, generating large datasets in 4 seconds while using 35% of memory (8 GB of RAM and 450HD GB).

After one year, Ping et al. established DataSynthesizer, a publicly available artificial dataset generator [Ping et al., 2017]. It consists of three components: DataDescriber, DataGenerator, and ModelInspector. The first module analyzes a real dataset and infers the domain, data types, correlations, and attribute distributions, adding noise to preserve privacy. The DataGenerator uses this knowledge to create new data through uniform sampling, while the ModelInspector evaluates the accuracy by comparing the similarity between the real and artificial datasets. The system preserves statistical properties, as similar pairwise correlations in both datasets demonstrate.

In 2019, Yoon et al. introduced PATE-GAN, a platform merging a customized version of the Private Aggregation of Teacher Ensembles (PATE) with GANs to build differential private datasets [Yoon et al., 2019b]. By replacing the GAN discriminator with a PATE structure, PATE-GAN ensures privacy while generating data. The authors evaluated its utility across various experiments, comparing it to DPGAN [Xie et al., 2018], and traditional GANs. Results using metrics, such as the area under the receiver operating characteristics curve and the area under the precision-recall curve, revealed that PATE-GAN outperformed DPGAN. The authors also implemented a novel metric, synthetic ranking agreement, to test predictive model ranking more efficiently.

Two years later, Kuppa et al. explored privacy in artificial datasets from the attacker and auditor perspectives, focusing on membership inference attacks (MIA) [Kuppa et al., 2021]. MIA targets the identification of individual membership in ML models trained on synthetic data. The authors developed an instance-level privacy score to filter synthetic samples vulnerable to MIA. They evaluated membership privacy leakage with various methodologies [Yoon et al., 2019b, Xie et al., 2018, Smith and Smith, 2020, Xu et al., 2019b] with attack accuracy as a key metric. Empirical observations revealed that the privacy score reduced attack accuracy across all methods and datasets. Additionally, the study highlighted that larger datasets increase the risk of privacy leakage.

In 2006, Pei and Zaiane proposed a technique for synthesizing multidimensional datasets for clustering and outlier analysis, using distribution and transformation methods [Pei and Zaïane, 2006]. This approach accommodates diverse user specifications, such as number, size, shape, location of clusters, and other meta-features. The dataset generation process employs normal and uniform probability distributions and transformations like linear and Box-Muller. It operates on three levels: difficulty (easy, medium, difficult), density, and outliers. The difficulty level defines cluster shape and location, density controls size and compactness, and the outlier level

governs the dispersion and density of the outliers in the dataset. This approach was evaluated on two fronts: execution time for generating large datasets and efficiency in clustering analysis and outlier detection, assessed visually [Guha et al., 1998, Karypis et al., 1999, Sheikholeslami et al., 1998, CHEESEMAN et al., 1988]. The experiments showed that this approach has a linear execution time, independent of the difficulty level, and that WaveCluster and AutoClass algorithms performed best in visual tests.

Five years later, Alburquerque et al. developed a high-dimensional dataset generation framework based on statistical distributions [Albuquerque et al., 2011]. Users input meta-features, such as dimensionality, sample size, class distribution, default probability distribution, and data type properties, which are integrated into an n-dimensional probability density function (PDF) to create a base dataset. Patterns and trends can be added to selected dimensions using generator objects, including one-dimensional and two-dimensional PDFs and a probability distribution plane. Random perturbations are incorporated to emulate irregularities. This methodology was scrutinized on dataset generation time and fidelity to real data through visual analytics. Experimental findings demonstrate that the generation time escalates with dimensionality, and the method effectively models non-orthogonal trends and outlier behaviour.

In 2005, Giannotti et al. proposed a framework for generating artificial movement data through the reconstruction of cellular network trajectories [Giannotti et al., 2005]. This system comprises a synthetic trajectory generator, a log generator, and a trajectory reconstruction module. The first structure builds spatiotemporal data to simulate user movements based on the user requirements. The log module utilizes these instances and an antenna coverage map to produce antenna logs. The last component then rebuilds the original trajectories using these logs and the antenna coverage map. The platform was assessed under different scenarios, including slow motion without obstacles and faster motion with barriers. Visual demonstrations confirmed its ability to emulate cellular data while ensuring privacy protection.

A year later, Lin et al. implemented IDSG, a dataset generator for training and testing information discovery and analysis systems while mitigating privacy and legal problems [Lin et al., 2006]. IDSG contains three modules: an XML schema governing data structure, data generators tasked with generating data, and a data generation engine orchestrating these components. Although the results are limited, the authors highlighted IDSG's capacity to generate 1.8 million credit card transactions per hour, showcasing its scalability.

In 2008, Omari et al. developed TARtool, a synthetic dataset generator tailored for Market Basket analysis [Omari et al., 2008]. Building on ARtool[7], it integrates association rule mining with temporal dynamics to augment the generation of datasets in retail and e-commerce. This tool also models transaction timing using Poisson distributions and stochastic factors. Its performance was benchmarked to the data mining tool WEKA, focusing on frequent item identification and the computational cost of building .db and .dat (ASCII with timestamps) files. The tests demonstrated that both methodologies yielded comparable outputs, though TARtool generated .dat files faster at the expense of increased processing time. Eno and Thompson investigated the inversion mapping

[7]Documentation available at https://depts.washington.edu/acelab/proj/art/

of data mining patterns to create artificial datasets of arbitrary sizes [Eno and Thompson, 2008]. Their approach hinges on three main components: the synthetic data definition language (SDDL), describing the generation process; the parallel synthetic data generator (PSDG), responsible for building SDDL-based datasets; and the predictive model markup language (PMML), facilitating the transfer of data mining models between various software packages. Their approach encodes a DT as a PMML file to instantiate an SDDL file, which PSDG then uses for dataset generation. By assessing the misclassification ratio between the real DT and the synthetic dataset, the authors discerned notable similarities indicative of the fidelity in preserving authentic patterns.

In 2013, Lopes and Smith-Miles introduced a methodology for generating instances for branch problems, focusing on the iterative refinement of the instance generator by comparing the real world with initially generated data [Lopes and Smith-Miles, 2013]. They utilized feedback from a classifier to adjust the generator, enhancing the resemblance of synthetic samples to real instances. The expected applicability coefficient, a novel metric, facilitates this tuning process. The effectiveness of this approach was evaluated on the Udine Timetabling problem using an instance generator [Burke et al., 2010]. The results showed improvements with the proposed technique in the instance generator through comprehensive analysis and empirical investigation.

A year later, Aydin et al. presented a spatiotemporal synthetic dataset generator that builds instances characterized by dynamic changes in both location and shape [Aydin et al., 2014]. This methodology builds random feature types, which encapsulate the spatial and temporal dynamics of the instances. These meta-features are then aggregated based on the spatial neighbourhood to form distinct patterns, such as creation. Its performance was assessed against three spatiotemporal co-occurrence pattern methods [Pillai et al., 2013]. Experimental observations highlighted ERMO-DG's utility in benchmarking spatiotemporal pattern mining algorithms.

In 2016, Spasic et al. suggested a modified version of the resource description framework dataset generator for the social benchmarking network (SBN) [Spasić et al., 2016]. Using the Virtuoso procedure, they aimed to enhance the structural fidelity of synthetic datasets compared to real-world data. To achieve a coherence level of 0.6, the authors modified the SBN generator, including removing location data from most posts and comments, adding comments in graphic interchange format, incorporating user mentions and links, and improving post visibility. A comparative analysis of datasets generated from the original and refined SBN approaches revealed a decrease in coherence level from 0.8 to 0.6, demonstrating their modification's effectiveness.

In 2017, del Carmen Rodríguez-Hernández et al. examined DataGenCARS, a framework engineered for creating datasets across diverse domains, with the focus on integrating context information crucial for CARS evaluation [del Carmen Rodríguez-Hernández et al., 2017]. Data-GenCARS allows for configuring user, context, and item type schemas, user profiles, ratings, and item attribute generation. This tool captures statistical properties from the original dataset and employs attribute, instance, and rating generators to produce synthetic samples. DataGenCARS's performance is evaluated by its capacity to generate fully synthetic datasets for RSs evaluation and to create realistic datasets through metrics such as mean average error (MAE), F-measure, recall, and precision. Initial testing showcased DataGenCARS's adeptness in building different datasets

with distinct features and context depth, while subsequent experiments highlighted its proficiency in crafting datasets mirroring authentic data features.

Five years ago, Mendonça et al. studied a dataset generator for information visualization and ML testing [Mendonça et al., 2020]. This platform offers user customization and consists of several modules: the manager, data model and its dimensions, generators, and data output. The manager oversees submodules related to data, communications, and visualization. The data model defines the generators and dataset dimensions, guiding the data creation process. Generators can produce values based on various types, such as random, geometric, accessory, functional, and sequential. The data output module facilitates the exportation of information generated during this process. The framework's efficiency was evaluated using visual analytics across distinct settings, including varying outlier quantities, class separation, missing values, and class imbalance. Experimental results demonstrated its capacity to build synthetic datasets that are useful for ML testing.

In 2012, Reif et al. designed a dataset generator for meta-learning systems to produce datasets with specific mean kurtosis and skewness [Reif et al., 2012]. The approach uses genetic algorithms to optimize dataset generation [De Rainville et al., 2012]. The generator was tested in four separate runs, focusing on properties like mean skew, min skew, max skew, naive Bayes, and nearest neighbor. Despite using identical generation parameters, the generator created four distinct datasets, each exhibiting different distributions of the targeted properties.

In 2021, Jomaa et al. introduced Dataset2Vec, a meta-feature extractor combining meta-features from tabular data with those learned by deep NNs [Jomaa et al., 2021]. Dataset2Vec uses the Kolmogorov-Arnold representation theorem [Kůrková, 1991] to represent datasets as sets of target pairs hierarchically and uses the DeepSet model [Zaheer et al., 2017] for meta-features regression. The authors also proposed a dataset similarity learning task to help Dataset2Vec recognize whether two samples come from the same dataset. Performance was evaluated through a hyperparameter optimization task, comparing Dataset2Vec to various baseline methods [Seeger, 2004, Bergstra et al., 2011, Hutter et al., 2011, Bergstra and Bengio, 2012, Springenberg et al., 2016] using the average distance as the minimum metric. The observed results showed that Dataset2Vec meta-features outperform traditional meta-features.

In 2017, Forestier et al. suggested strategies for synthesizing time series using dynamic time warping (DTW), focusing on distinct average weighting methods [Forestier et al., 2017]. The authors proposed three variations of DTW Barycentric Averaging: "average all", which computes the average of all time series; "average selected", which selectively averages a group of near time series; and "average selected with distance" (ASD) that incorporates the relative distance between selected time series and their closest neighbour. The performance of these techniques was evaluated against windows wrapping [Le Guennec et al., 2016] using the metric error rate. Experimental findings indicated that ASD outperformed alternative methods, highlighting its utility in scenarios with limited original samples.

In 2021, Silva et al. devised SAGAD, a data augmentation technique to solve small-scale tabular problems by employing the feature relationships within the training dataset [Silva et al., 2021]. They also proposed an extension known as data generation based on complexity per classes

(DABEL), which refines the generated data by iteratively accounting for class ambiguity. DABEL computes the loss function per epoch to improve synthetic data quality. SAGAD and DABEL were evaluated against GANT [Ashrapov, 2020] and SMOTE [Chawla et al., 2011] using the F1-score and one-tailed t-test. The observed results demonstrated that SAGAD and DABEL performed differently depending on the dataset size, with both particularly effective for smaller datasets.

Table 3.2 organizes the literature review into specific categories to facilitate its understanding. These include surveys that provide a broader overview of the field, application-driven studies targeting synthetic data generation, particularly in the context of cybersecurity (domain explored in SNOOKER), privacy-oriented approaches that address data protection, evaluation and benchmarking studies that assess the performance of existing tools, research on data augmentation and use of multidimensional synthetic data, and the main approaches employed in this field, including meta-learning, GANs and VAEs.

Table 3.2: Synthetic data generation related work summary

| Ref | Survey | Applications | Cybersecurity | Privacy | Evaluation | Data Augmentation | Multidimensional Dataset | Meta Learning | GAN | VAE |
|---|---|---|---|---|---|---|---|---|---|---|
| [Filchenkov and Pendryak, 2015] | ✓ | - | - | - | - | - | - | ✓ | - | - |
| [Surendra and Mohan, 2017] | ✓ | - | - | ✓ | - | - | - | - | - | - |
| [Vanschoren, 2018] | ✓ | - | - | - | - | - | - | ✓ | - | - |
| [Popić et al., 2019] | ✓ | - | ✓ | - | - | - | - | - | - | - |
| [Bellovin et al., 2019] | ✓ | - | - | ✓ | - | - | - | - | ✓ | - |
| [Faez et al., 2021] | ✓ | ✓ | - | - | - | - | - | - | ✓ | ✓ |
| [Dankar and Ibrahim, 2021] | ✓ | - | - | - | ✓ | - | - | - | - | - |
| [Chen et al., 2021] | ✓ | - | - | ✓ | ✓ | - | - | - | ✓ | - |
| [Nikolenko, 2019] | ✓ | ✓ | - | ✓ | - | - | - | - | ✓ | - |
| [Rivolli et al., 2022] | ✓ | - | - | - | - | - | - | ✓ | - | - |
| [Lemos et al., 2022] | ✓ | - | - | - | - | - | - | ✓ | - | - |
| [Gonzales et al., 2023] | ✓ | ✓ | ✓ | - | - | - | - | - | - | - |
| [O'Shaughnessy and Gray, 2011] | - | ✓ | ✓ | - | - | - | - | - | - | - |
| [Boggs et al., 2014] | - | ✓ | ✓ | - | - | - | - | - | - | - |
| [Pendleton and Xu, 2017] | - | ✓ | ✓ | - | - | - | - | - | - | - |
| [Kim and Kim, 2019] | - | ✓ | ✓ | - | - | - | - | - | - | - |
| [Meyer-Berg et al., 2020] | - | ✓ | ✓ | - | ✓ | - | - | - | - | - |
| [Borah et al., 2020] | - | ✓ | ✓ | - | - | - | - | - | - | - |
| [Le et al., 2020] | - | ✓ | ✓ | - | - | - | - | - | ✓ | - |
| [Creswell et al., 2018] | ✓ | ✓ | - | - | - | - | ✓ | - | ✓ | - |
| [Torres, 2018] | - | ✓ | - | - | - | - | ✓ | - | ✓ | - |
| [Xu and Veeramachaneni, 2018] | - | ✓ | - | - | - | - | - | - | ✓ | - |
| [Lu et al., 2019] | - | - | - | ✓ | - | - | - | - | ✓ | - |
| [Yoon et al., 2019a] | - | ✓ | - | - | - | - | - | - | ✓ | - |
| [Baowaly et al., 2019] | - | ✓ | - | - | - | - | - | - | ✓ | - |

Table 3.2: Synthetic data generation related work summary

| Ref | Survey | Applications | Cybersecurity | Privacy | Evaluation | Data Augmentation | Multidimensional Dataset | Meta Learning | GAN | VAE |
|---|---|---|---|---|---|---|---|---|---|---|
| [Smith and Smith, 2020] | - | ✓ | - | - | - | - | - | - | ✓ | - |
| [Vega-Márquez et al., 2019] | - | ✓ | - | - | - | - | - | - | ✓ | - |
| [Little et al., 2021] | - | - | - | - | ✓ | - | - | - | ✓ | - |
| [Mozo et al., 2021] | - | ✓ | - | ✓ | ✓ | - | - | - | ✓ | - |
| [Wan et al., 2017] | - | - | - | - | - | - | ✓ | - | - | ✓ |
| [Razghandi et al., 2022] | - | ✓ | - | - | - | - | - | - | ✓ | ✓ |
| [Lin et al., 2006] | - | ✓ | - | - | - | - | - | - | - | - |
| [Reif et al., 2012] | - | - | - | - | - | - | - | ✓ | - | - |
| [Dandekar et al., 2018] | - | - | - | ✓ | ✓ | - | - | - | - | - |
| [Goncalves et al., 2020] | - | - | - | ✓ | ✓ | ✓ | - | - | ✓ | - |
| [Nettleton, 2016] | - | - | - | ✓ | - | - | - | - | - | - |
| [Ayala-Rivera et al., 2016] | - | ✓ | - | ✓ | - | - | ✓ | - | - | - |
| [Ping et al., 2017] | - | ✓ | - | ✓ | - | - | - | - | - | - |
| [Yoon et al., 2019b] | - | ✓ | - | ✓ | - | - | - | - | ✓ | - |
| [Kuppa et al., 2021] | - | - | ✓ | ✓ | - | - | - | - | ✓ | - |
| [Pei and Zaïane, 2006] | - | ✓ | - | - | - | - | ✓ | ✓ | - | - |
| [Albuquerque et al., 2011] | - | - | - | - | - | - | ✓ | ✓ | - | - |
| [Giannotti et al., 2005] | - | ✓ | - | - | - | - | - | - | - | - |
| [Omari et al., 2008] | - | ✓ | - | - | - | ✓ | ✓ | ✓ | - | - |
| [Eno and Thompson, 2008] | - | ✓ | - | - | - | - | - | - | - | - |
| [Lopes and Smith-Miles, 2013] | - | ✓ | - | - | - | - | - | - | - | - |
| [Aydin et al., 2014] | - | ✓ | - | - | - | - | - | ✓ | - | - |
| [Spasić et al., 2016] | - | ✓ | - | - | - | - | - | - | - | - |
| [del Carmen Rodríguez-Hernández et al., 2017] | - | ✓ | - | - | ✓ | - | - | ✓ | - | - |
| [Mendonça et al., 2020] | - | ✓ | - | - | ✓ | - | - | ✓ | - | - |
| [Jomaa et al., 2021] | - | ✓ | - | - | - | - | - | ✓ | - | - |
| [Forestier et al., 2017] | - | ✓ | - | - | - | - | ✓ | - | - | - |
| [Silva et al., 2021] | - | ✓ | - | - | - | - | ✓ | - | - | - |

Although various approaches have been proposed, GANs have gathered substantially more attention within this domain. Furthermore, works tailored explicitly to the cybersecurity field, along with the presence of surveys, are examined.

Over the years, various open-source systems in this area have been introduced. The following analysis excludes domain-specific tools focusing on particular application areas such as healthcare, genetics, and other specialized fields.

Synthetic data vault (SDV[8]) is among the first synthetic generators dedicated to creating tabular,

[8]Documentation available, respectively, from https://sdv.dev/ and https://sdv.dev/Copulas/

relational, and time series data. This project includes Copulas[8], CTGAN [Xu et al., 2019b], and TGAN[9], three packages designed for data generation using both statistical and DL techniques. In contrast, DoppelGANger generates high-fidelity time series data with metadata using GANs [Lin et al., 2020]. YData[10] provides tools for generating tabular data and time series with various GAN implementations. Strategies using declarative models and highly customizable interfaces, like Sinner [Mannino and Abouzied, 2020] and Synth[11] can also provide interesting results. Other tools, including Faker[12], Pydbgen[12], Twinify[12], Smart noise synthesizer[12], DP_WGAN_UCLANESI[12], among others contribute to the diversity of synthetic dataset generators. While several of these tools can create different types of data, they do not emulate business logic specific to a domain, such as cybersecurity incident treatment. As described in section 3.2, the proposed dataset generator addresses this limitation by building more realistic datasets that include domain-relevant operations.

Regarding online communities focused on synthetic data, several key platforms stand out. The strategic data project (OpenSDP[13]) offers a range of educational tools and resources for analysts across various areas; OpenSynthetics[14] serves as a hub for synthetic data tailored to computer vision and ML applications; and, GenRocket[15], fosters a community engaged dedicated to exploring and exchanging ideas about synthetic data. Additionally, the SDV Slack channel provides support and insights about this topic.

### 3.1.3 Discussion

Based on the previous analysis, we can address **RQ1** - "What approaches can build synthetic data based on real data for ticketing systems". In this context, synthetic data generation can be achieved with data distribution, GANs, and other ML strategies. Although the second and third groups could provide more in-depth results, the first category is computationally lighter and allows greater control over generating different features. Regarding evaluation, many metrics targeting structural similarity, bi-variable correlations, statistical dependency difference, prediction accuracy, and disclosure risk have been used to test such approaches [Rachkovskij and Kussul, 1998].

After analyzing the state-of-the-art, some research points can be identified to enhance the reliability and realism of synthetic dataset generators. A critical feature that could improve the quality of these datasets is the incorporation of business logic, as discussed in **RQ2** - "Can we generate synthetic data that closely mimics real data regarding distribution and business logic?". By embedding domain-specific rules and processes into the data generation tool, synthetic datasets can more accurately reflect the operational realities of various fields, such as cybersecurity, finance, and

[9]Documentation available at https://github.com/sdv-dev/TGAN
[10]Documentation available at https://docs.synthetic.ydata.ai/1.4/
[11]Documentation available at https://www.getsynth.com/docs/getting_started/synth
[12]Documentation available, respectively, from https://faker.readthedocs.io/en/stable/, https://github.com/tirthajyoti/pydbgen, https://github.com/DPBayes/twinify, https://github.com/opendp/smartnoise-sdk and https://github.com/nesl/nist_differential_privacy_synthetic_data_challenge
[13]Website available at https://opensdp.github.io/data/
[14]Website available at https://opensynthetics.com/
[15]Website available at https://community.genrocket.com/

healthcare. This integration facilitates the simulation of compliance-related scenarios, allowing, for example, the generation of datasets that replicate processes, such as ticket assignment, incident treatment, and other cybersecurity operations. As a result, these datasets improve the relevance of training data for intelligent systems and support organizations in testing and refining their operational strategies in a controlled and realistic environment. Synthetic data also addresses one of the main motivations for synthesizing data: the scarcity of available datasets. Existing datasets may be limited in data quantity or quality.

Another potential improvement for synthetic dataset generators is the ability to generate temporal data correlated with specific geographical locations. Numerous tools can build time series data and geographical locations, but they overlook the importance of incorporating timezone data, which can impact time-sensitive operations in various domains. Timezone awareness enables the generation of data that accurately reflects local times, simulating more realistic scenarios.

## 3.2 Proposed Architecture for SNOOKER

SNOOKER creates generic datasets for general-purpose use cases that are independent of context and domain-based synthetic datasets tailored to specific industries or applications. Both dataset types are designed to simulate real-world behaviours, such as ticket treatment, ticket prioritization, and other activities. This framework emulates the procedures used by IT teams, incorporating features such as ticket escalation, prioritization, and scheduling, and other characteristics typical of helpdesk systems. Such a tool is supported by an interface where users can configure various aspects of the dataset generation. Different teams treat these alerts, each comprising different types of operators with distinct characteristics like work shift, speed, and other features. For clarity, unless otherwise specified, the term operators includes all the roles explored in section 2.1.

Figure 3.3 illustrates SNOOKER's main components: dataset customization and generation (elaborated in sections 3.2.1 and 3.2.2, respectively). In the first stage, users configure generation parameters and optional configurations, as outlined in section 3.2.1.1. The second phase manages helpdesk operations, such as ticket creation, ticket enrichment using domain-specific data (defined in optional setup), team and operator assignment, and ticket scheduling and escalation. The operations shown with a dashed style are related to the application domain and can be adapted to other fields besides cybersecurity.

### 3.2.1 Dataset Customization

The dataset customization (yellow shapes in Fig. 3.3) is supported by an interface developed with PyQT[16] and allows users to follow a standard generation using features from a configuration file or manually customize various generation attributes, such as tickets, families, SOC teams, operators, and optional data. For example, users can personalize the IP type (in the context of cybersecurity), the format type, and the desired output parameters. This generator provides multiple levels of

[16]Documentation available at https://wiki.python.org/moin/PyQt

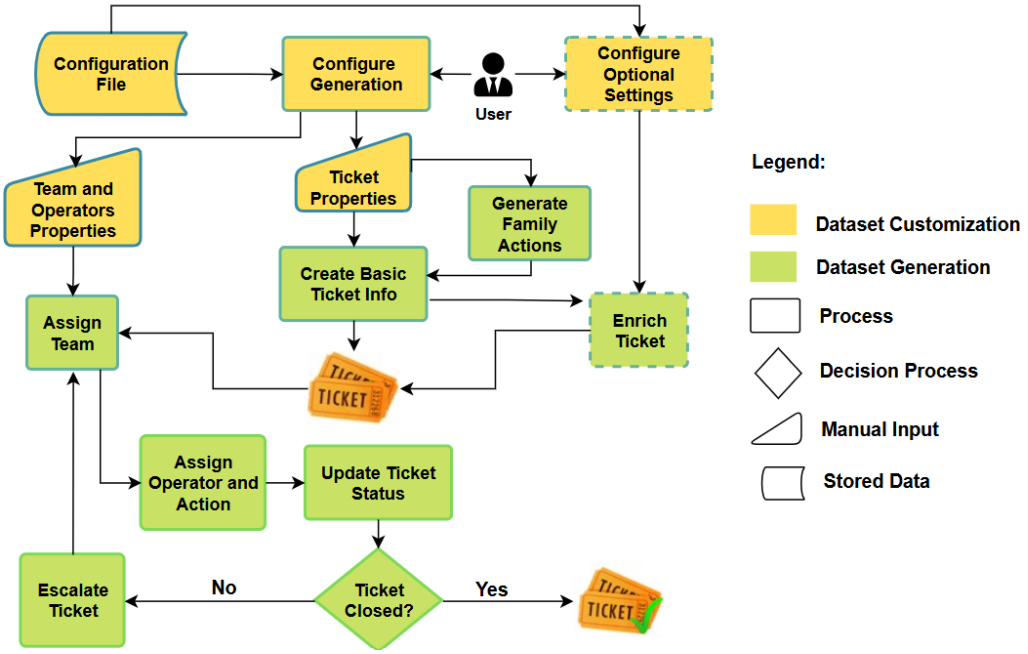



Figure 3.3: SNOOKER Architecture

granularity, allowing fine-tune control over these parameters, allowing the creation of personalized datasets.

When users decide to include real information in the generation task, SNOOKER exclusively relies on the meta-features derived from the dataset, thereby mitigating potential privacy issues.

#### 3.2.1.1 Customizable Parameters

This framework is equipped with various functionalities that require user-configurable settings and external files. Each setting is detailed with the type, description, impact, and example. The examples presented for each setting were derived from discussions with S21sec professionals and random selection.

Table 3.3: Generation Settings

| Settings | Type | Description | Example |
|---|---|---|---|
| Mode | STRING | Custom or standard generation | Standard |
| Output | STRING | Format in CSV, XLSX, or TXT | CSV |
| Source | STRING | Format stored in database, CSV, or XLSX | CSV |
| Debug | BOOLEAN | Debugs activities related to ticket generation and processing | False |
| Log data | BOOLEAN | Log generation and treatment operations | False |
| Seed | INT | Seed in random tasks | 1 |

| | | | |
|---|---|---|---|
| Ticket similarity detection | BOOLEAN | Detect similar tickets | True |
| Action disparity detection | BOOLEAN | Detect distinct procedures based on the max similarity | True |
| Action max disparity | INT | Maximum number of differences between operator and subfamily actions | 3 |

As shown in Table 3.3, users can generate synthetic datasets using the standard approach or customized configuration based on the parameters depicted in Tables 3.4, 3.5, 3.6, and 3.7. Users can further adjust other aspects of the generation process by selecting the source and output data formats, specifying the seed for reproducibility during testing, and configuring options for logging, debugging, ticket similarity, and treatment action similarity.

The generated dataset, available in CSV, XLSX, or TXT, includes diverse features, including timestamps (raised, allocated, and closed), family (incident type), subfamily (sub-incident type), team, operator, resolution action, resolution duration, and status. The source data, provided in a database, CSV, or XLSX format, should contain knowledge regarding the creation datetime, family, subfamily, the action steps performed by operators, and their corresponding timestamps. The debug option provides users with detailed information about the operations conducted during ticket generation and treatment, enabling them to find and analyze potential errors. Meanwhile, the log function records all the processes followed during these tasks. The similarity detection option allows SNOOKER to identify and monitor similar tickets (in terms of subfamily and client), simulating cases where repeated attacks within a short time may indicate suspicious activity. The last two features emulate scenarios where the action mainly used to address a specific subfamily differs from the action selected by the operator, measured using the Levenshtein distance. The disparity detection property enables or disables this functionality, while the max disparity parameter dictates the maximum allowable difference between the verified actions. The usage of these two properties is further elaborated in section 3.2.2.4.

Table 3.4: Ticket Settings. T: Tickets Generation; TD: Ticket Distribution; TDUR: Tickets Duration; TT: Transferred Tickets; TSIM: Ticket Similarity

| **Settings** | **Type** | **Description** | **Impact** | **Example** |
|---|---|---|---|---|
| Train and Test | INT | Number of train and test tickets | T | 3514 and 0 |
| Growth rate | FLOAT | Ticket growth rate in each month | T | 2% |
| Priority levels | INT | Ticket's priorities | T | 5 |
| Initial date | STRING | Initial date for ticket generation | T | April 2020 |
| End date | STRING | Final date for ticket generation | T | April 2021 |
| Clients | INT | Number of clients | TSIM | 4 |

Ticket Settings

| Settings | Type | Description | Impact | Example |
|---|---|---|---|---|
| Escalation probability | FLOAT | Escalation likelihood | TT | 0% |
| Seasonality periods | DICTIONARY | Seasonality over the year | TD | Extracted from real dataset |
| Outlier rate | FLOAT | Outlier rate | TDUR | 10% |
| Outlier cost | FLOAT | Procedures extra cost | TDUR | 10% |

Table 3.4 outlines characteristics related to ticket generation, including the number of tickets, trend over time, priority levels, creation datetime, seasonality, clients, escalation, and outlier.

The number of tickets comprises the number of training and testing incidents, which helps generate training (resolved tickets) and testing (unresolved tickets) datasets for other applications (for example, test the tool discussed in section 4.3). The generation of the latter dataset type helps evaluate other systems, as tested in section 4.4. If users want to generate only a training dataset, the test number should be 0. The growth rate indicates the trend in ticket generation over time (negative values reflect a declining tendency, positive values show an increasing tendency, and a value of 0 suggests no changes in tendency). The priority levels define the existing severity categories of the tickets. The initial and end dates specify the datetime range for ticket generation. The clients indicate the number of entities reporting incidents, and the escalation likelihood represents the probability of tickets being immediately escalated to more expert teams (analyzed in section 3.2.2.4). The seasonality is extracted from the real dataset prior to generation and is preprocessed to address missing data and other potential issues. The resulting distribution is then influenced by the defined growth rate, and variations based on the time of day and day of the week (studied in Table 3.5). Smoothing is also applied in this process to ensure a more realistic pattern. The outlier rate and cost represent the frequency of outliers in the tickets and their impact on the procedure duration, affecting ticket resolution, respectively (discussed in section 3.2.2.2).

Table 3.5: Family Settings. T: Tickets Generation; TD: Ticket Distribution; TSIM: Ticket Similarity

| Settings | Type | Description | Impact | Example |
|---|---|---|---|---|
| Families | INT | Number of incident types (e.g. VPN) | T, TD | 12 |
| Generation | STRING | Default families (external file) or custom | T, TD | Default |
| Seasonality periods | DICTIONARY | Seasonality over the year | T, TD | Extracted from real dataset |
| Distribution type | STRING | Normal or Uniform Distribution | TD | Normal |

Family Settings

| Settings | Type | Description | Impact | Example |
|---|---|---|---|---|
| Day distribution | DICTIONARY | Likelihood to occur equally during the day, higher on daylight or nighttime | TD | Equal |
| Week distribution | DICTIONARY | Likelihood to occur equally during the week, higher on weekdays or weekends | TD | Equal |
| Min, Max subfamilies | INT | Minimum and maximum number of subfamilies (e.g. VPN_1) | T | 1 to 11 |
| Min, Max occurrences | INT | Minimum and maximum number of occurrences for escalation | TSIM, TT | 5 to 7 |
| Min, Max detection interval | INT | Minimum and maximum minutes interval sharing the subfamily and client | TSIM, TT | 59 to 90 minutes |
| New family and Subfamily | FLOAT | New family and subfamily likelihood | TD | both with 0% |

Table 3.5 illustrates multiple parameters for the configuration of the families, such as number of families, generation type, seasonality, number of subfamilies, number of permitted events sharing the subfamily and client within a detection interval, distribution type, distributions of the day and week, and probability of new family and subfamily.

As the name suggests, the number of families represents the number of incident types available for the tickets. The generation of these families can happen by default, where a configuration file provides standard families or custom, with the users configuring the remaining family parameters. The seasonality emulates families being more/less common in specific year periods and can only be personalized using a real dataset. The distribution type ensures that families have a uniform or normal behaviour over time. The day and week distributions further customize their likelihood throughout the day and week. For example, tickets may have a higher probability of occurring during the week instead of the weekend. They may also be likelier to happen during specific times of the day. The minimum and maximum number of subfamilies define the number of subfamilies for each family. The minimum and maximum occurrences and detection interval parameters, only pertinent when the ticket similarity is enabled, indicate the permitted number of incidents sharing the same subfamily and client within a specific period. This scenario, studied in section 3.2.2.4, emulates the occurrence of similar attacks within a short timeframe. The new family and subfamily rate details the likelihood of introducing an unknown family/subfamily into the system.

Table 3.6: Technique Settings. TDUR: Tickets Duration; TS: Ticket Scheduling

| Settings | Type | Description | Impact | Example |
|---|---|---|---|---|
| Techniques | INT | Number of techniques in each family | TDUR, TS | 7 |
| Real Techniques | DICTIONARY | Emulated real techniques duration | TDUR, TS | Extracted from real dataset |
| Min, Max subtechniques | INT | Minimum and maximum number of subtechniques | TDUR, TS | 3 to 4 |
| Min, Max subtechnique duration | INT | Maximum and minimum duration of each subtechnique | TDUR, TS | 2 to 5 |
| Min, Max subtechnique rate | INT | Maximum and minimum frequency of each subtechnique | TDUR, TS | 50% to 100% |

Table 3.6 presents the configurations regarding the generation of techniques, including the total number of techniques, the number of subtechniques within each technique, and the duration and frequency of each subtechnique. The minimum and maximum values define the range for selecting the number of subtechniques, their durations, and their frequencies. The subtechnique rate influences the generation of actions for subfamilies, while the subtechnique duration aids in estimating the duration of actions.

Table 3.7: Team and Operator Settings. TDUR: Tickets Duration; TT: Transferred Tickets; TS: Ticket Scheduling; TE: Team Treatment

| Settings | Type | Description | Impact | Example |
|---|---|---|---|---|
| Generation | STRING | Standard or custom generation | T, TE | Standard |
| Ticket assignment | STRING | Tickets are assigned to the lowest team, or a ticket percentage is allocated to each team | TS, TT, TE | Hierarchical |
| Name and Team | STRING | Operators name and team assigned | TE | Steve and L1 |
| Shift | INT | Operator shift | TE | 2 |
| Speed and Learning | FLOAT | Operator resolution speed and learning rate (within lower and upper boundaries) | TE, TS, TDUR | 1.2 and 1.4 |
| Min, Max learning limit | INT | Minimum and maximum incident counter for operator improvement | TDUR | 10 and 20 |
| Subfamily action probability | FLOAT | Likelihood to use the subfamily predefined action | TE | 90% |

Team and Operator Settings

| Settings | Type | Description | Impact | Example |
|---|---|---|---|---|
| Repeat action probability | FLOAT | Likelihood to use the previously action in the same subfamily | TE | 90% |

Table 3.7 describes the configurable parameters for SOC teams and their operators. These include using standard or custom teams, defining the ticket distribution for each team, and customizing the operator's features.

The difference between standard and custom team generation lies in their flexibility: standard teams consist of predefined teams: four teams ranked by expertise level (from lowest to highest), with the lowest-ranked team having 12 operators and the others having six operators. The custom option allows users to create operators with distinct profiles. The ticket assignment feature supports various ticket types, assigning them hierarchically or distributing them among teams based on a defined percentage. The remaining attributes focus on the operator's customization. The name, team, and shift refer to the operator's name, their assigned team, and their work shift. The speed represents the baseline performance speed of an operator and influences the action duration estimation. The learning rate simulates skill development or decline, representing the operator's improvement or degradation over time based on repeated use (or lack of use) of specific actions. The learning limit sets the maximum improvement or minimum deterioration to prevent infinite enhancement or excessive decline in performance. The last two probabilities translate the likelihood of using the action typically used to solve the subfamily and the possibility of using the same action to resolve a ticket from the same subfamily.

Regarding domain parameters, SNOOKER employs IP-related data to generate source and destination IPs alongside their respective ports. The process behind generating their addresses and ports is discussed in section 3.2.2.2 together with the ticket generation.

Figure 3.4 represents SNOOKER's main interface, where most customization parameters are illustrated.

#### 3.2.1.2 Input

SNOOKER generates synthetic datasets capable of emulating different behaviours using the following data sources:

- S21sec dataset – contains 3585 anonymized tickets solved by SOC operators between 2020 and 2021. Each entry includes details about the incident type and subtype, raised and fixed timestamps, a list of each procedure step timestamp, incident status, and other features beyond this project's scope. This dataset was preprocessed regarding feature selection, missing data through multiple imputation by chained equations (MICE), and other inconsistencies before being integrated into SNOOKER;

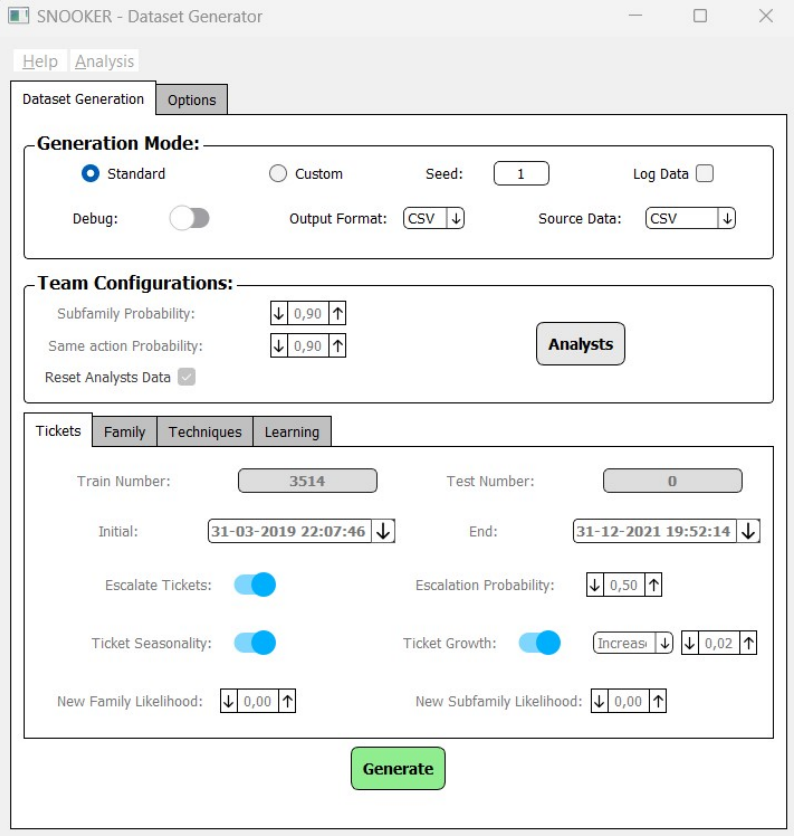



Figure 3.4: SNOOKER - Dataset Customization

- Countries – provides data about every existing country. For each country, it details the continents, capital, pytz timezones, and additional information irrelevant to this work. The dataset is formatted as a Python list and was retrieved from a GitHub repository[17];
- IP Geolocation – maps IPv4 address networks to their respective country. Each entry includes the network (in Classless Inter-Domain Routing format) and other attributes outside this project's scope. This JSON file was provided by Datahub[18];
- Custom Configuration file – this YAML (YAML Ain't Markup Language) file is used for initializing and testing dataset generation in SNOOKER. It provides configuration details for standard generation, including attributes regarding tickets (Table 3.4), families (Table 3.5), techniques (Table 3.6), SOC team and operators (3.7), and domain parameters. This file also contains predefined teams, operators, and families. Additionally, it specifies transformation functions (add, remove, change, maintain) used for generating actions.

As detailed in section 3.2.2.2, using the S21sec dataset allows the generated tickets to follow a similar daily distribution. The combination of country data and IP geolocation sources ensures that

[17]Dataset available at https://gist.github.com/mlisovyi/e8df5c907a8250e14cc1e5933ed53ffd
[18]Dataset available at https://datahub.io/core/geoip2-ipv4#data

the generated IPs of the tickets are aligned with their assigned locations. This analysis operates under the assumption that VPNs do not mask IPs.

### 3.2.2 Dataset Generation

This phase comprises the operations shown in the green shapes in Fig. 3.3. During this stage, each ticket is organized into three categories: initialization, domain, and treatment. The initialization phase creates the basic structure of the tickets with preliminary fields typically found in the client infrastructure. These fields can be the creation timestamp, basic metadata (type, priority), and other information. The domain step enriches the tickets through domain-specific knowledge. Depending on the selected domain, this could include attributes such as geographical location, IP data, and other contextual data. The treatment phase populates each open ticket with the assigned operator and the corresponding action required to address it. Table 3.8 illustrates the main features incorporated into the tickets in each phase.

Table 3.8: Ticket Data Model

| **Data Category** | **Features** | **Description** |
|---|---|---|
| Initialization | ID | Ticket identifier |
| | Priority | Ticket relevance |
| | Raised | Datetime of when the ticket was raised |
| | Family | Incident type (e.g. VPN) |
| | Subfamily | Incident subtype (e.g. VPN_1) |
| | Initial Escalation | Transfer (or not) to another team immediately |
| Domain | Client | Client or organization name; |
| | Location | Country where the ticket was detected; |
| | Destination and Source IPs and Ports | The IPs and Ports of destination and source entities. |
| Treatment | Allocated | Datetime of when the ticket is allocated |
| | Fixed | Datetime of when the ticket is fixed by the operator |
| | Team | Team assigned to the ticket |
| | Operator | Operator responsible for treating the ticket |
| | Action | Procedure employed by the operator in closing the ticket |
| | Subtechniques | Individual steps with a particular duration composing the available procedures |
| | Status | Closed or Transferred |

The following example illustrates these ticket segments:

1. Initialization: $\langle ID : 1,\ Priority : 3,\ Raised : 2021\text{-}07\text{-}27\ 20{:}51{:}25,\ Family:\ X,\ Subfamily : X\_1,\ Initial\ Escalation : False \rangle$
2. Treatment: $\langle Allocated : 2021\text{-}07\text{-}27\ 20{:}51{:}25,\ Fixed : 2021\text{-}07\text{-}27\ 21{:}14{:}25,\ Team : L1,\ Operator:\ A,\ Action : [(A,3),(2f,5),(6a,4),(2f,5)(w,6)],\ Status : Closed \rangle$
3. Domain: $\langle Client : Client\_1,\ Location : Serbia,\ Source\ IP:\ 198.51.100.25,\ Source\ Port:\ 56789,\ Destination\ IP:\ 203.0.113.45\ Destination\ Port:\ 8080 \rangle$

The dataset generation comprises multiple functions related to ticket generation, enrichment, assignment, scheduling, and escalation. Regarding the latter three tasks (discussed in section 2.4), SNOOKER utilizes a priority-based strategy for ticket assignment with the support of multilevel feedback queues (MLFQ), with prioritization following a severity-based hierarchy. For ticket scheduling, tickets may be addressed immediately or deferred for later solving, depending on their urgency. SNOOKER follows a predefined set of rules to escalate tickets to other teams for further investigation when necessary (detailed in section 3.2.2.4). These design choices were informed by consultations with cybersecurity experts, ensuring alignment with industry best practices.

#### 3.2.2.1 Technique Generation

Before exploring the ticket generation, it is essential to establish procedures for closing the tickets. In real-world scenarios, operators rely on general guidelines and specific workbooks to address incidents. For instance, in cybersecurity, an operator investigates why users cannot log in to their accounts. The operator checks if the user enters the correct username and ensures it matches the system records. In SNOOKER, these procedures are modeled using abstract techniques and subtechniques. Although operators do not directly employ the techniques, they can be categorized into subtechniques. Using the previous example, the technique would be troubleshooting login issues, while the subtechnique would be username verification.

The subtechniques of each action generated contain unique traits, ensuring a structured sequencing encompassing initiation, transferring, and conclusion operations. Transfer steps allow the simulation of cases wherein the assigned team encounters problems beyond their treatment capabilities, necessitating assistance from higher-tier teams. Each step contains a duration influenced by the presence of outliers (assigned randomly) or real datasets. In the latter option, SNOOKER can mimic the step duration from the authentic dataset.

Regarding the representation of the techniques, we chose to use alphanumeric characters (a-z, A-Z, 0-9), while the subtechniques assume multiple variations from them. The usage of alphanumeric characters stems from the need for a more straightforward and abstract representation of the problem. For subtechniques, we instantiate them with hexadecimal values ranging from 0 to 255, providing a reasonable, scalable approach. Although some domains are characterized by multiple procedures available, the range 0-255 provides sufficient granularity to distinguish the variations of the subtechniques while keeping the representation understandable. If a greater variety becomes necessary, the representation can be expanded by incorporating additional bits. To

conserve space and improve clarity, all techniques and subtechniques from Table 3.9 and subsequent examples have their hexadecimal component omitted.

Table 3.9: Techniques and subtechniques generated for a given family

| Operation Type | Techniques | Subtechniques |
|---|---|---|
| Initiation | 7 | 31: 2, d7: 5, e6: 8 |
| Intermediate | p | b1: 4, 2c: 2, 91: 3 |
| Intermediate | G | 39: 1, 52: 5, 20: 2 |
| Conclusion | L | 96: 3, f5: 6 |
| Transferring | D | 22: 4 |

As illustrated in Table 3.9, a particular family exhibits five distinct techniques denoted as 7, p, G, L, and D. Each technique comprises various subtechniques accompanied by their duration. For instance, technique 7, an initiation operation, can be further instantiated by three subtechniques (31, d7, and e6). The operator could employ an action sequence to close a ticket, such as [31, b1, 20, f5] with a duration of 14 minutes. The penultimate step is omitted in tickets involving transferring steps, and the conclusion step is substituted with a transfer operation. While the family and subfamily actions remain constant, the operator action is converted [31, b1, 22] with a duration of 10 minutes. Although these actions are executed sequentially, operators can add, remove, or change specific steps within this framework.

The real dataset also impacts the duration of both techniques and subtechniques. During real data, the time taken for each step taken to fix a family is recorded and later used to estimate the duration of the simulated procedures in SNOOKER. Simulated operators may execute steps with an entirely new duration, or one influenced by the real dataset. Other features like client and priority (where one indicates low impact and five shows high criticality) are randomly generated at this stage.

#### 3.2.2.2 Ticket Generation and Enrichment

Tickets are generated and populated with the initialization attributes analyzed in section 3.8. Each ticket is assigned a geographical location, client, priority, and a raised date, with the latter being randomly chosen within the user-defined datetime range.

Two factors influenced the decision to prioritize data distribution techniques over ML and DL: controllability and alignment with business logic operations. Data distribution provides direct and granular control over parameter generation using explicit rules, thresholds, and transformations. This approach can emulate business rules, ensuring the outcomes meet operational expectations. While ML and DL strategies offer a degree of controllability through hyperparameter tuning, feature engineering, and interpretability tools, they require specialized expertise and more attention in model design, validation, and monitoring. Moreover, implementing a validator, which could also function as a generator, would be necessary to ensure that data generated by ML and DL adheres to business logic. This functionality could introduce additional layers of complexity and increase the computational overhead to SNOOKER.

The generation of datetimes for each ticket involves three temporal distributions: time of day, day of the week, and day of the year. The latter distribution is influenced by the patterns observed in the real dataset, particularly its role in replicating authentic daily distributions of ticket events. When a real dataset is available, SNOOKER uses conditional probabilities derived from the annual frequency distribution of the tickets and their corresponding families over a year (section 3.1.1). These probability distributions work as weight inputs to ensure that the generated dataset aligns with the temporal distribution of the authentic one, preserving similarity in the distribution patterns of tickets and families throughout the year. Without such data, datetime values are generated randomly within a specified date range. The user can define custom temporal distributions (such as hours, weekdays, and months) either through the configuration file or the generator's interface, depending on the selected generation mode. Equations 3.1 and 3.2 detail the date and datetime generation based on the three temporal distributions discussed.

$$P_{date}(d) = \frac{P_M(m) \cdot P_W(w)}{Z} \tag{3.1}$$

$$P_{datetime}(D) = \frac{P_{date}(d) \cdot P_T(t)}{Z} \tag{3.2}$$

where $P_{date}$ refers to the probability of a date `d`, $P_M$ is the probability distribution of the real seasonality for a particular day of the year `m`, $P_W$ is the probability distribution for a particular day of the week `w`, and $P_T$ is the probability distribution for a specific time of the day `t`, and `Z` is the normalization factor (to ensure that the sum of all distributions is one).

This process could allow the user to have tickets with multiple temporal constraints. For instance, a user may prefer to maintain the daily distribution of the real data while introducing variability in the time of the day and day of the week.

The family and subfamily assignment for each ticket follows a similar approach to datetime generation, where three datetime components (day, month, and year) are considered. Equation 3.3 determines the probabilities associated with the datetime elements, providing a unified likelihood for each family.

$$P_{Fi} = (P_{Ti} \cdot P_{Wi} \cdot P_{Mi}) \tag{3.3}$$

where $P_{Fi}$ is the aggregated probability of each family `i`, $P_{Ti}$ is the probability of each family `i` during the day, $P_{Wi}$ is the probability of each family `i` during the week, and $P_{Mi}$ is the probability of each family `i` during the year (also retrieved from the real dataset).

With Eq. 3.4, a cumulative probability value is computed for all families concerning the temporal dynamics, such as time, weekday, and month of the ticket. Subsequently, a random number is generated within the range defined by the minimum and maximum values derived from the cumulative function. The assignment of the ticket to a specific family depends on its placement within the same function. Furthermore, the subfamily assignment involves selecting a random type from the available options in each family.

$$PC_k = \sum_{i=1}^{k} P_{Fi} \tag{3.4}$$

where $P_{Fi}$ is the aggregated probability of family `i`, and $PC_k$ is the family cumulative probability of `k` number of families.

During ticket enrichment, domain-specific data, in our case, cybersecurity details, including geographical location, destination, and source IP addresses and ports, are incorporated into the tickets. This process is accomplished with the support of the external files containing country and IP geolocation data, shown in section 3.2.1.2.

For each ticket, the destination and source information are generated using a multi-step randomization process. First, a geographical location is randomly selected and assigned to the ticket. Based on this country, a network is chosen using the IP geolocation dataset, and an IP address, either IPv4Address or IPv6Address, is randomly picked from the available pool within that network. A similar strategy is followed to determine the source IP, but using a different randomly selected country. Conversely, these IPs can be used to infer their associated geographical locations using the IP geolocation data referenced earlier. Finally, the port numbers are assigned at random and categorized into three groups: well-known ports (0-1023), registered ports (1024-49151), and ports available for unrestricted use by client programs (49152-65535) [Goralski, 2017].

#### 3.2.2.3 Team and Operator Assignment

This section details various SNOOKER operations, such as team assignment, operator-driven action generation, and operator allocation responsible for each ticket. As discussed in section 2.4, these operations are typical in ticketing systems. For confidentiality reasons, we do not disclose how S21sec handles these tasks.

The team assignment task allocates a team to each ticket through two strategies, including:

- Non-Hierarchical – tickets can be addressed by any team, regardless of their priority. Each team is assigned a percentage of tickets determined by the user;
- Hierarchical – high-level teams handle critical incidents, while low-level teams manage tickets not necessitating exhaustive investigation (followed by SNOOKER since S21sec adopts it).

The assignment of operators and the orchestration of their actions are crucial for the system workflow. Upon ticket entry, SNOOKER identifies all the operators available. If operators are free, the ticket proceeds to the next phase; otherwise, it is placed into the appropriate MLFQ based on its priority. This approach, combined with the dynamic adjustment of the priorities of the pending tickets based on the time spent in each queue, ensures a balanced and fair ticket treatment process. Tickets that remain in lower-priority queues for extended periods are promoted to higher-priority queues, ensuring timely resolution. When a ticket is closed, and there are tickets in these queues, SNOOKER retrieves the next ticket from the highest-priority queue for processing.

This methodology allows tickets to be handled immediately or scheduled for later treatment during the same or subsequent work shift, depending on the operator's availability.

Once available operators are identified, SNOOKER determines potential actions they can take to resolve the ticket. Operators may reuse a previously utilized action or create a novel one for the ticket resolution. This decision depends on the last two probability factors defined in Table 3.7. Suppose operators have handled the ticket's subfamily in the past, and the analyzing operator has experience with that subfamily. In that case, SNOOKER uses the "repeat action probability" to compute whether the operator should reuse the previously applied procedure or build a new action. On the other hand, if there are no records of the subfamily being treated by SNOOKER or the analyzed operator, the system utilizes the "subfamily action probability" to decide whether the operator should employ the standard action typically used for that subfamily or build a new one.

In the generation of actions, SNOOKER uses the subtechniques inherent to the ticket family to execute operations like addition, removal, substitution, or retention of specific components of the subfamily action. These operations are restricted to subtechniques with the intermediate operation type. As illustrated in Table 3.9, new operator actions can be instantiated, such as:

- Adding any subtechnique derived from the techniques p and G. It is important to note that adding any subtechnique from 7, L, and D is forbidden to prevent the disruption of the sequential action execution;
- Removing/changing one subtechnique, specifically b1 or 20. Subtechniques drawn from p and G (2c, 91, 39, 52) may be employed when a change operation is selected.

During this process, the operator may repeat the same step multiple times. For example, in real life, an operator may want to determine whether the connectivity is consistent or identify potential unreachable devices. In this case, the operator would perform the same action for every server's IP address to evaluate the response. The duration of an action is calculated as the sum of the products obtained by multiplying the temporal duration associated with each subtechnique and the operator's work speed.

The presence of outliers introduces an unexpected behaviour in the ticket treatment, as the duration of an action is increased based on the outlier cost defined during the customization task. For example, if a ticket is marked as an outlier and the action duration to treat it is 20 minutes, with an outlier cost of 10%, the adjusted duration becomes 22 minutes (20 + 20 x 0.1). If the computed resolution time exceeds the operator's designated shift duration, the ticket is placed for later treatment in the corresponding MLFQ.

With all the steps completed, SNOOKER allocates the first operator available and action for each ticket. To simulate experience gain or loss, the duration of each action step used by the operator is adjusted. It is reduced if the operator has performed it frequently and recently, or increased if it is rarely used or has not been used for a long time. Each step has a defined minimum and maximum duration to prevent unlimited improvement or degradation over time.

The resulting synthetic dataset with all tickets treated, generated based on the user-defined specifications, is then stored in an external file for further analysis and development.

#### 3.2.2.4 Ticket Escalation

Simulates cases where closed tickets are escalated and verified by higher-tier teams. The escalation scenarios include:

- Immediate escalation – according to the escalation probability detailed in Table 3.4, the tickets can be flagged for immediate escalation. This approach emulates incidents that require analysis and resolution exclusively by expert teams;
- Similar ticket escalation – tickets are escalated when the defined number of similar cases (subfamily and client) is detected within a set time frame. The limit of similar cases is determined by randomly selecting a value within the range of minimum and maximum occurrences described in Table 3.5. For each matching case, SNOOKER increments a counter, escalating the ticket once the maximum allowed occurrences is reached. Can be used to simulate the occurrence of potential suspicious activity like multiple login attempts;
- Levenshtein distance – escalation is triggered when the computed distance between the subfamily and the operator's action steps exceeds the action maximum disparity, specified in Table 3.3. For example, if the subfamily's action is [31, b1, 20, 90] and the operator decides [31, 91, 52, 90], Eq. 3.5 would compare the steps and yield a distance of 2 ("b1" and "20" are different from "91" and "52"). This scenario reflects cases where the operator's chosen action deviates significantly from standard actions, potentially indicating inaccuracies in assessments, outliers, or other abnormal behaviours.

$$lev(i,j) = \begin{cases} |i| & if\,|j| = 0, \\ |j| & if\,|i| = 0, \\ lev(tail(i), tail(j)) & if\,head(i) = head(j), \\ 1 + min \begin{cases} lev(tail(i), j)) \\ lev(i, tail(j)) \\ lev(tail(i), tail(j)) \end{cases} & otherwise \end{cases} \tag{3.5}$$

where $lev(i, j)$ denotes the Levenshtein distance between strings $i$ and $j$, $|i|$ and $|j|$ represent the lengths of these strings. The tail refers to the substring obtained by excluding the string's first character.

From an implementation standpoint, the escalated ticket assumes a transfer status, while a duplicate instance is concurrently closed by lower-tier teams. This mechanism facilitates collaboration between different teams involved in ticket analysis and resolution.

Table 3.10: Escalation Scenario of a ticket

| Raised | Fixed | Team | Subfamily Action | Operator Action | Status |
|---|---|---|---|---|---|
| 2021-07-27 20:51:25 | 2021-07-27 21:21:23 | L1 | [A, 2f, d8, X] | [A, 2f, 6a, 6a, 2f, w] | Transfer |
| 2021-07-27 21:21:23 | 2021-07-27 21:36:20 | L2 | [A, 33, 8e, X] | [A, 33, 8e, X] | Closed |

Table 3.10 illustrates a ticket escalated from L1 to L2 due to the computed distance between the subfamily and operator actions, measured with Eq. 3.5, exceeding the maximum disparity defined in Table 3.3.

## 3.3 Results

The performance of SNOOKER is evaluated through external and internal tests. Initially, a comparative analysis is conducted between a dataset generated by SNOOKER and an established S21sec dataset. This external test assesses SNOOKER's ability to replicate the incident distribution in the S21sec dataset. Subsequently, a series of internal experiments is conducted to investigate how different compositions of SOC teams impact SNOOKER's ticket treatment. The objective is to analyze the effects of varying numbers of operators and the distribution of work shifts on the efficiency of ticket treatment.

All experiments were conducted on a computer equipped with an Intel Core i9-10980HK (8 cores/16 threads, 2.4 to 3.0 GHz), 64 GB of RAM, and an SSD drive.

### 3.3.1 Comparison between the real and synthetic datasets

SNOOKER's capacity to replicate behaviours similar to those observed in the S21sec dataset was inspected in two key aspects: the distribution of generated families throughout one year and the emulated duration required to solve these incidents.

In this analysis, SNOOKER processes the input data described in section 3.2.1.2, with the properties of the configuration file based on the examples provided in Tables 3.3, 3.4, 3.5, and 3.6. While parameters such as the number of tickets and families to generate, their creation datetime range, seasonality, technique distribution, and team hierarchy were derived from the real dataset, some unavailable attributes were assigned using random selection, and others were generated based on conditioned choices informed by discussions with S21sec.

Figure 3.5 illustrates the distribution of anonymized families in the S21sec dataset from April 2020 to April 2021. This dataset comprises twelve families, each with a distinct distribution pattern across the analyzed timeframe.

The distribution of the real dataset is highly imbalanced over the year, with increased ticket seasonality observed during the middle of the year, particularly from March to September. This period exhibits more activity, potentially due to seasonal demands, operational cycles, or other external factors affecting the frequency of incidents. In contrast, the start and end of the year show lower incident activity, confirming the imbalanced nature of the ticket distributions. Furthermore, this analysis demonstrates a significant prevalence of incidents from family A, with J occurrences appearing more rarely.

Figure 3.6 represents the incident distribution of synthetic families over the same period as recorded in the real dataset. The generated dataset replicates the same number of incident types as the real dataset, ensuring a visually comparable analysis.

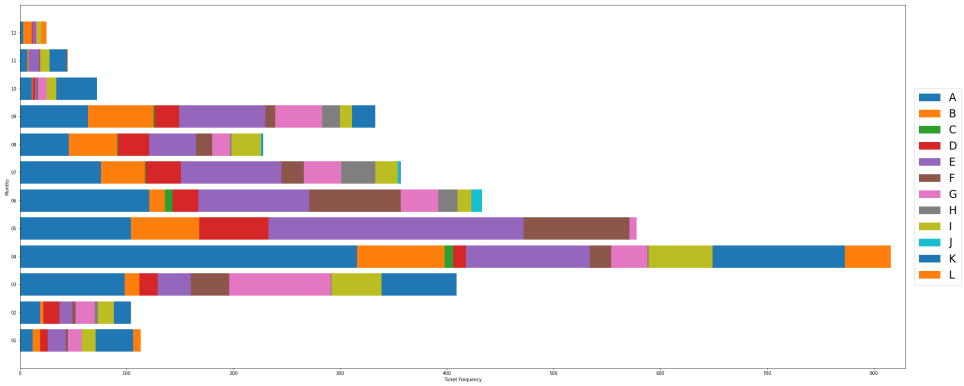



Figure 3.5: Real Dataset

Since both datasets share the same families and preserve the same sequential order in the figures shown, it is possible to observe that they exhibit analogous behaviour throughout the year. Both demonstrate higher ticket frequencies during months of higher seasonality (from March to September), with the family distributions following a similar pattern. To validate this observation, three statistical tests are performed: the K-S test, KLD, and Hellinger distance. The K-S test, conducted under the null hypothesis that two distributions are identical, yields a value of 0.015 and a corresponding p-value of 0.82. Given that the K-S statistic is close to 0 and the p-value exceeds the conventional significance threshold of 5%, the test confirms that the distributions of the real and generated datasets are similar. Further analysis using the KLD revealed a divergence of 0.001, indicating minimal information loss between the two distributions. Similarly, the Hellinger distance outputted a value of 0.02, further corroborating the close similarity between the datasets.

In addition to analyzing the temporal distribution of incidents, SNOOKER models the time real operators spend to resolve tickets. Specifically, it estimates the average resolution time for each real family and emulates this duration by proportionally distributing it across the techniques assigned to the generated family. The duration of each subtechnique is calculated based on the technique duration, adjusted by a delta percentage. For example, with a delta of 10%, a subtechnique with a

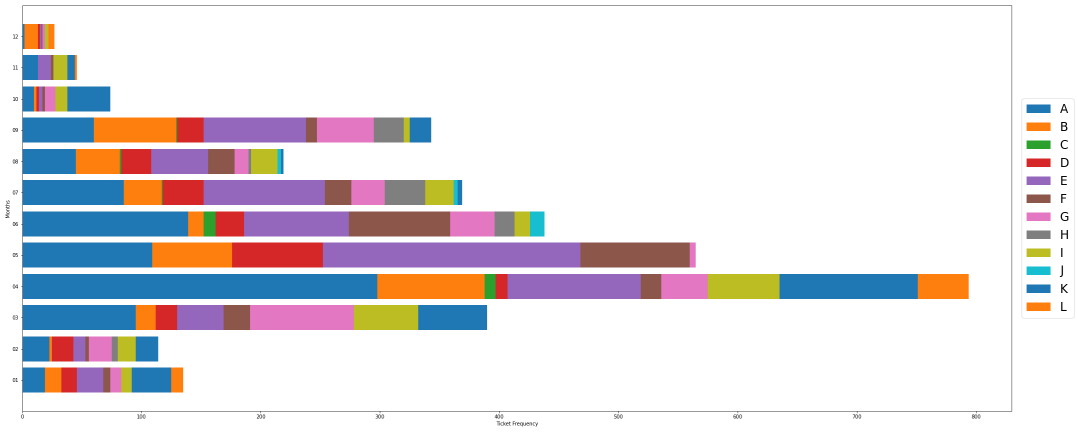



Figure 3.6: Generated Dataset

baseline duration of 20 minutes may take 18 or 22 minutes. Figure 3.7 illustrates the average time spent by operators in closing events of different families in both datasets.

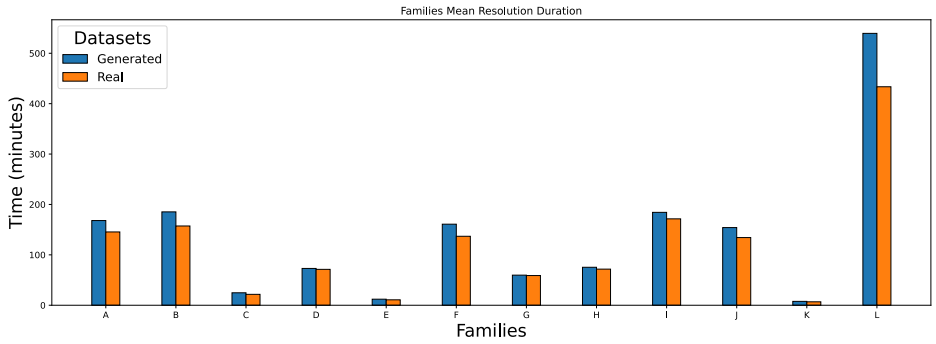



Figure 3.7: Comparison between resolution times of real and synthetic families

Except for family L, where the resolution time differs by approximately 100 minutes, the remaining families require roughly the same time for resolution. The minor differences observed can be attributed to SNOOKER's ability to model the evolving behaviour of operators over time. By simulating the improvement in the time operators take to apply correction procedures, SNOOKER can emulate the learning curve that operators experience with each treatment. The gap in family L is due to the delta applied and the real duration used. In this case, with a real duration of 400 minutes and a delta of 10%, a subtechnique could take up to 440 minutes (400 * 1.1) to resolve families of this nature. Together with the operator learning, this accounts for the observed difference.

It is important to note that the significant resolution times seen in Fig. 3.7 for the real dataset are accurate and may indicate potential errors during ticket treatment, such as forgetting to close a ticket. For example, some instances of family L required more than one day to be solved, which is unusual in cybersecurity operations, where time and speed are critical [Yoo, 2022].

### 3.3.2 Influence of Team Dimensionality

This section explores how variations in the number of operators and their distribution across the SOC team shifts can influence ticket scheduling and, consequently, ticket treatment in SNOOKER.

To investigate this aspect, we utilize the SNOOKER framework to generate and manage five datasets, each consisting of 20,000, 25,000, 30,000, 35,000, and 40,000 tickets. These datasets cover April 2020 to April 2021 and include 12 distinct families. Each team composition explored in this analysis is divided into three shifts and is based on two dimensions: number of operators (fewer [9], standard [12], and more [15]) and shift imbalance (low, balanced, medium, and high). These configurations are evaluated across three scenarios that use distinct incident distributions to assess how SOC teams would behave under different conditions:

- Incorporation of real distribution in conjunction with normal distributions for both time of day and day of the week (Table 3.12). Essentially, it reflects real-world patterns, such as higher incident volume of a certain type during business hours or specific days of the week;

- Only utilizing normal distributions for both time and day of the week (Table 3.13). Allows controlled variability while still adding realistic peak periods during certain times of the day and days of the week;
- Exclusively employing uniform distributions for both time and day of the week (Table 3.14). Allows the generation of incidents with no temporal dynamics, uniformly distributed across all hours and days of the week.

Given our objective to emulate real-world behavioural patterns, such as the tendency for certain ticket families to occur more frequently at specific times or on particular days of the week, we model the distribution of families using normal distributions. The determination of the number of operators within each imbalance level is obtained as follows:

$$O_{shift} = N/S \tag{3.6}$$

$$Low : S_0 = O_{shift} - 1; S_1 = O_{shift} + 1; S_2 = O_{shift} \tag{3.7}$$

$$Medium : S_0 = O_{shift} - 1; S_1 = O_{shift} + 2; S_2 = O_{shift} - 1 \tag{3.8}$$

$$High : S_0 = O_{shift} - 2; S_1 = O_{shift} + 3; S_2 = O_{shift} - 1 \tag{3.9}$$

where N is the total number of operators, S is the number of shifts, and $O_{shift}$ is the number of operators per existing shift (with shifts 0, 1, and 2 considered in the following analysis). It is important to note that N must be divisible by three, as three shifts are employed; however, this constraint should be adjusted if a different number of shifts is applied. As referred to in section 3.2.1, SNOOKER provides fine-grained control over the customization and generation operations.

Table 3.11 presents the scenarios SNOOKER considers based on the number of operators and imbalance level. Operator availability is categorized into three levels: fewer (9 operators), standard (12 operators), and more (15 operators). Using Equations 3.6, 3.7, 3.8, 3.9, operators are allocated across three shifts: shift 0 (00:00-07:59), shift 1 (08:00-15:59), and shift 2 (16:00-23:59).

Table 3.11: Operator Distribution per shift

| Scenario | Shift Imbalance | Operators Size | Shift 0 | Shift 1 | Shift 2 |
|---|---|---|---|---|---|
| 1 | | Fewer Operators | 3 | 3 | 3 |
| 2 | Balanced | Standard | 4 | 4 | 4 |
| 3 | | More Operators | 5 | 5 | 5 |
| 4 | | Fewer Operators | 2 | 4 | 3 |
| 5 | Low | Standard | 3 | 5 | 4 |
| 6 | | More Operators | 4 | 6 | 5 |
| 7 | | Fewer Operators | 2 | 5 | 2 |
| 8 | Medium | Standard | 3 | 6 | 3 |
| 9 | | More Operators | 4 | 7 | 4 |
| 10 | | Fewer Operators | 1 | 6 | 2 |
| 11 | High | Standard | 2 | 7 | 3 |

Operator Distribution per shift

| Scenario | Shift Imbalance | Operators Size | Shift 0 | Shift 1 | Shift 2 |
|---|---|---|---|---|---|
| 12 | | More Operators | 3 | 8 | 4 |

The performance evaluation of these scenarios is conducted using a What-IF methodology, an approach known for assessing sensitivity, risk identification, and performance optimization [Lyon and Popov, 2021]. This approach allows researchers to inspect how variations in specific inputs affect the outcome. In this context, it is applied to explore the scenarios described in Table 3.11, focusing on metrics including the percentage of scheduled tickets for later treatment, average wait time for initiating ticket resolution, and standard deviation, indicative of the dispersion observed in the wait times of tickets. Both ticket resolution and wait time are measured in minutes.

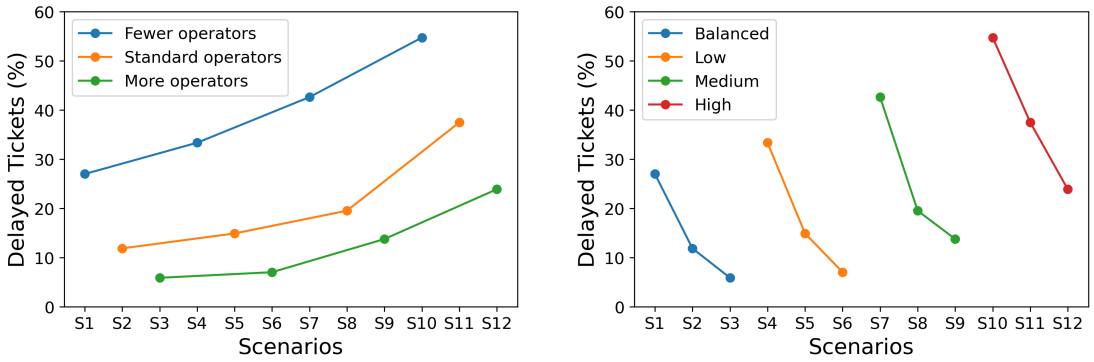
Each metric values in the Tables 3.12, 3.13, and 3.14 reflect the average outcome across ten runs using different seeds. Although only the means are shown in these tables, the complete results are provided in Appendix A. Using multiple seeds is important because SNOOKER's ticket generation and handling processes involve randomness. The seed controls how random values are generated, which affects the dataset and simulation results. Each seed functions as a starting point for the random generator number, leading to the generation of diverse datasets. This approach ensures that the results are not overly influenced by any single dataset configuration.

Table 3.12: Ticket Scheduling with S21sec Distribution

| Distribution Scenarios | 20000 | | | 25000 | | | 30000 | | | 35000 | | | 40000 | | | AVG | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 11 | 0.78 | 2.96 | 16 | 1.17 | 3.78 | 26.8 | 2.77 | 7.36 | 38.8 | 5.06 | 12.34 | 45.9 | 7.2 | 16.58 | **27** | **3.4** | **8.6** |
| Scenario 2 | 4.7 | 0.32 | 1.83 | 6.3 | 0.4 | 1.99 | 11 | 0.79 | 3.05 | 16.8 | 1.26 | 4.09 | 20.6 | 1.55 | 4.62 | **11.88** | **0.86** | **3.12** |
| Scenario 3 | 3.2 | 0.22 | 1.54 | 3.6 | 0.24 | 1.54 | 5.7 | 0.38 | 2.02 | 7.7 | 0.52 | 2.38 | 9.3 | 0.58 | 2.45 | **5.9** | **0.39** | **1.99** |
| Scenario 4 | 15.7 | 1.57 | 5.46 | 21.2 | 2.19 | 6.89 | 32.5 | 6.39 | 17.28 | 45.2 | 13.91 | 34.67 | 52.2 | 17.59 | 41.33 | **33.36** | **8.33** | **21.13** |
| Scenario 5 | 6.3 | 0.44 | 2.21 | 8.5 | 0.56 | 2.44 | 13.7 | 1.25 | 4.65 | 21.1 | 2.26 | 7.53 | 25 | 2.67 | 8.58 | **14.92** | **1.44** | **5.08** |
| Scenario 6 | 3.6 | 0.25 | 1.63 | 4 | 0.26 | 1.6 | 6.4 | 0.46 | 2.31 | 9.7 | 0.72 | 3.03 | 11.5 | 0.78 | 3.11 | **7.04** | **0.49** | **2.34** |
| Scenario 7 | 21.9 | 2.44 | 7.21 | 29.6 | 4.09 | 11.04 | 42.3 | 12.63 | 29.6 | 56.4 | 30.42 | 60.55 | 63 | 45.92 | 78.18 | **42.64** | **19.1** | **37.32** |
| Scenario 8 | 7.9 | 0.57 | 2.51 | 11.3 | 0.8 | 3.04 | 18.2 | 1.81 | 5.81 | 27.6 | 3.57 | 10.55 | 32.7 | 4.8 | 13.29 | **19.54** | **2.31** | **7.04** |
| Scenario 9 | 6 | 0.42 | 2.14 | 7.9 | 0.52 | 2.34 | 12.8 | 1.16 | 4.54 | 19.4 | 2.15 | 7.39 | 22.8 | 2.47 | 8.15 | **13.78** | **19.1** | **37.32** |
| Scenario 10 | 37.9 | 15.23 | 39.41 | 45.6 | 22.14 | 51.05 | 55.2 | 43.38 | 80.86 | 65.2 | 73.45 | 113.43 | 69.8 | 91.32 | 126.87 | **54.74** | **49.1** | **82.32** |
| Scenario 11 | 29 | 13.74 | 37.86 | 32.7 | 18.57 | 47.36 | 37.5 | 31.44 | 68.01 | 42.7 | 45.54 | 86.88 | 45.6 | 47.91 | 88.99 | **37.5** | **31.44** | **65.82** |
| Scenario 12 | 12.9 | 1.39 | 5.42 | 16.5 | 1.83 | 6.61 | 23.5 | 5.1 | 15.8 | 31.6 | 10.31 | 29.37 | 35 | 11.78 | 33.02 | **23.9** | **6.08** | **18.04** |

The scenarios are described in Table 3.11. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.

The results presented in Table 3.12 are derived from the scenarios described in Table 3.11 across five datasets, where time series data from a real dataset and normal distributions for the time of the day and day of the week are used in ticket generation. Across all examined scenarios and dataset sizes, an increase is observed in all metrics analyzed, characterized by progressive growth in pending tickets, average wait time, and standard deviation metrics. As expected, shifts with high imbalance and fewer operators lead to a higher percentage of scheduled tickets and longer average wait times. In comparison, shifts with low imbalance and more operators have less impact on these metrics. Figures 3.8a, 3.8b, 3.9a, and 3.9b illustrate the results for the metrics studied using the same number of operators and shifts, respectively, in these conditions.



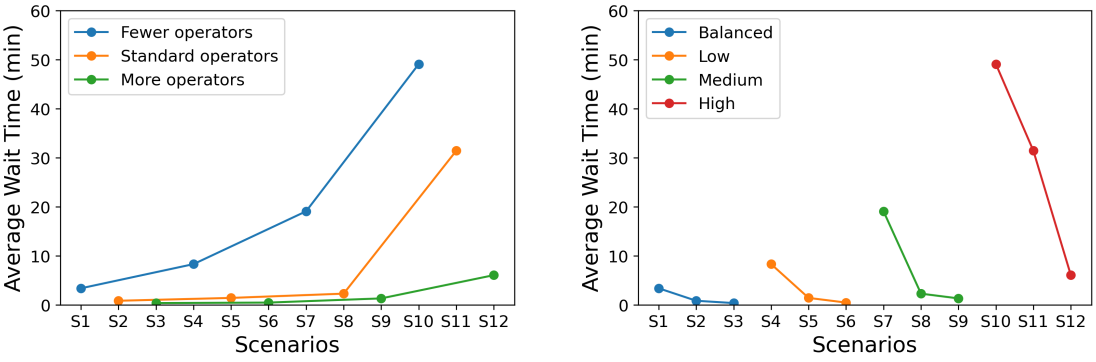

(a) Same number of operators

(b) Same shift balancement

Figure 3.8: Delayed tickets with all distributions

Table 3.13 evaluates the ticket treatment of a dataset generated based only on normal distributions for the day and day of the week. The analysis employs the same set of runs and seeds as the previous experiments. As expected, scenarios characterized by fewer operators and higher imbalance tend to result in more tickets scheduled for later treatment, leading to longer wait times. Contrary to the results obtained with real data, this test yielded a lower percentage of scheduled tickets and reduced average wait time. This observation can be attributed to the absence of seasonal fluctuations in the synthetic dataset. In real-world scenarios, ticket volume can increase significantly during specific periods due to seasonal patterns, overwhelming operators, and extended ticket wait durations. One interesting aspect observed in this analysis is that, except in scenarios 10, 11, and 12, the percentage of scheduled tickets is approximately half that observed in the corresponding scenarios in Table 3.12. This highlights the impact real dataset patterns can have on generating synthetic datasets. Figures 3.10a, 3.10b, 3.11a, and 3.11b represent the metric's outcomes obtained using the same number of operators and shifts, respectively, evaluated solely with normal distributions.

Table 3.14 explores the treatment of tickets generated using uniform distributions for the time and day of the week. The observed metric results are similar to those obtained in the prior




(a) Same number of operators

(b) Same shift balancement

Figure 3.9: Average wait time with all distributions

Table 3.13: Ticket Scheduling without S21sec Distribution

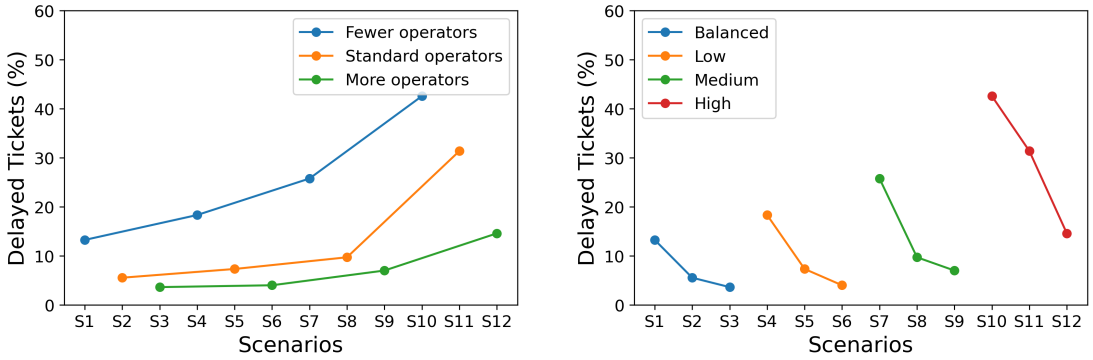
| Distribution Scenarios | 20000 | | | 25000 | | | 30000 | | | 35000 | | | 40000 | | | AVG | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 6.2 | 0.43 | 2.19 | 8.5 | 0.57 | 2.49 | 13.4 | 1.02 | 3.54 | 17.8 | 1.4 | 4.42 | 20.4 | 1.57 | 4.61 | **13.26** | **1** | **3.45** |
| Scenario 2 | 3.2 | 0.24 | 1.63 | 4 | 0.27 | 1.68 | 5.6 | 0.41 | 2.12 | 7.1 | 0.5 | 2.37 | 7.9 | 0.51 | 2.3 | **5.56** | **0.39** | **2.02** |
| Scenario 3 | 2.8 | 0.21 | 1.51 | 3.1 | 0.22 | 1.51 | 4.3 | 0.31 | 1.85 | 4 | 0.3 | 1.86 | 4 | 0.28 | 1.7 | **3.64** | **0.26** | **1.69** |
| Scenario 4 | 9.4 | 0.77 | 3.24 | 12.7 | 1.09 | 3.97 | 18.6 | 1.93 | 6.19 | 24 | 2.87 | 8.8 | 27 | 3.36 | 9.91 | **18.34** | **2** | **6.42** |
| Scenario 5 | 4.1 | 0.29 | 1.81 | 5.1 | 0.35 | 1.92 | 7.4 | 0.55 | 2.55 | 9.55 | 0.71 | 3 | 10.6 | 0.76 | 2.99 | **7.34** | **0.53** | **2.45** |
| Scenario 6 | 2.9 | 0.22 | 1.53 | 3.3 | 0.23 | 1.53 | 4 | 0.3 | 1.81 | 4.9 | 0.35 | 1.98 | 5 | 0.33 | 1.85 | **4.02** | **0.29** | **1.74** |
| Scenario 7 | 13.2 | 1.13 | 3.98 | 18.4 | 1.74 | 5.31 | 26.2 | 3.12 | 8.36 | 33.8 | 5.27 | 13.62 | 37.3 | 6.33 | 15.39 | **25.78** | **3.52** | **9.33** |
| Scenario 8 | 4.9 | 0.34 | 1.92 | 6.7 | 0.45 | 2.22 | 9.5 | 0.72 | 2.95 | 13 | 1.03 | 3.84 | 14.5 | 1.12 | 3.8 | **9.72** | **0.73** | **2.95** |
| Scenario 9 | 4.1 | 0.28 | 1.74 | 4.8 | 0.34 | 1.86 | 7 | 0.51 | 2.43 | 9 | 0.67 | 2.91 | 10-2 | 0.72 | 2.92 | **7.02** | **0.5** | **2.37** |
| Scenario 10 | 28.8 | 6.43 | 19.17 | 35.4 | 9.81 | 26.42 | 44.1 | 18.6 | 44.81 | 50.9 | 28.9 | 60.74 | 53.7 | 33.2 | 65.33 | **42.58** | **19.39** | **43.29** |
| Scenario 11 | 23.5 | 5.81 | 18.33 | 28 | 8.99 | 26.96 | 32.6 | 16.18 | 43.06 | 35.6 | 23.72 | 56.54 | 37.2 | 26.75 | 60.38 | **31.38** | **16.29** | **41.05** |
| Scenario 12 | 8.4 | 0.68 | 3.09 | 10.6 | 0.94 | 3.77 | 14.9 | 1.61 | 5.85 | 18.5 | 2.37 | 8.3 | 20.6 | 2.76 | 9.37 | **14.6** | **1.67** | **6.08** |

Same scenarios and metrics as in Table 3.12

experiment. This similarity suggests that SNOOKER's choice of normal or uniform distributions does not significantly affect ticket treatment and scheduling. This phenomenon may be explained by the nature of ticket generation, where the distribution spreads more or less evenly across periods, mitigating the interference with scheduling and treatment despite potential fluctuations in ticket volume within specific timeframes. Figures 3.12a, 3.12b, 3.13a, and 3.13b show how the metrics behave when the number of operators and shifts remains constant, respectively, considering only uniform distributions.

Across all these experiments, there is a consistent pattern: a higher percentage of scheduled tickets is associated with longer average wait times. This relationship can be attributed to the delay of more tickets for later treatment, which increases the likelihood of extended delays. When tickets are postponed, they can accumulate in their respective MLFQ, leading to longer wait times before an operator eventually treats them. Another insight from this analysis is that the growth of the three mentioned metrics follows a similar tendency across all the dataset sizes tested. This behaviour suggests that the dataset size does not notably affect ticket treatment and scheduling.

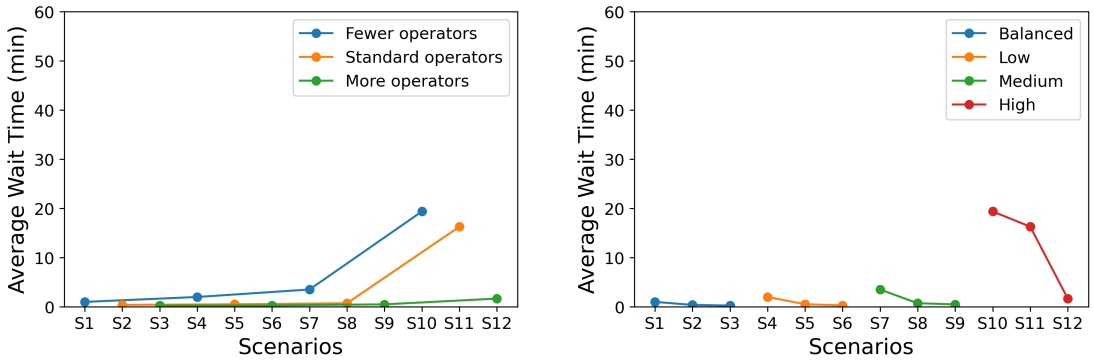
Figure 3.14 highlights the impact of varying team sizes and the distribution of operators across work shifts on ticket treatment, specifically regarding wait times using normal and real distributions.




(a) Same number of operators

(b) Same shift balancement

Figure 3.10: Delayed tickets with normal distributions




(a) Same number of operators

(b) Same shift balancement

Figure 3.11: Average wait time with normal distributions

As illustrated in Fig. 3.14, scenarios with fewer operators available (7 and 10) tend to exhibit higher wait times. This pattern is also observed in cases where there is a greater imbalance in the staff distribution across the shifts inspected (such as scenario 11). To ensure a more focused analysis, scenarios 1, 3, 4, 5, 6, 9, and 12 were excluded from this study. This decision was made to avoid overcomplicating the results and to highlight the scenarios with the most relevant tendencies for the test's purpose.

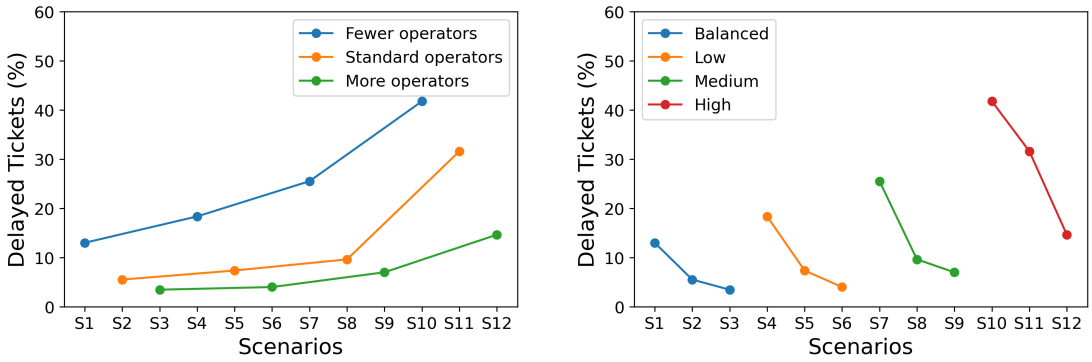
Figure 3.15 presents a similar analysis over the new dataset generated by SNOOKER but focused on average wait times from a weekly perspective.

As illustrated, in scenarios with fewer operators (7 and 10) or greater shift imbalance (11), tickets generated at the beginning and end of each day tend to experience longer wait times. This phenomenon occurs because tickets are continually delayed with fewer operators available in specific shifts, increasing their waiting periods before they are addressed. As described in section 4.3.4, an MLFQ strategy was implemented so that waiting tickets have their priority updated, preventing them from remaining in the queue indefinitely. Similar to Fig. 3.14, scenarios with fewer operators and more significant shift imbalance contribute to increased wait times.

Table 3.14: Ticket Scheduling with Uniform Distribution

| **Distribution** | **20000** | | | **25000** | | | **30000** | | | **35000** | | | **40000** | | | **AVG** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Scenarios** | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 6.2 | 0.43 | 2.28 | 8.3 | 0.58 | 2.52 | 13.3 | 1.01 | 3.51 | 17.4 | 1.36 | 4.25 | 19.9 | 1.49 | 4.4 | **13.02** | **0.97** | **3.39** |
| Scenario 2 | 3.3 | 0.25 | 1.67 | 4 | 0.28 | 1.74 | 5.6 | 0.41 | 2.13 | 7 | 0.48 | 2.32 | 7.8 | 0.5 | 2.29 | **5.54** | **0.38** | **2.03** |
| Scenario 3 | 2.9 | 0.21 | 1.51 | 2.9 | 0.22 | 1.52 | 3.6 | 0.28 | 1.78 | 3.9 | 0.29 | 1.83 | 4.1 | 0.28 | 1.71 | **3.48** | **0.26** | **1.67** |
| Scenario 4 | 9.4 | 0.78 | 3.27 | 12.8 | 1.09 | 3.99 | 19 | 2 | 6.41 | 23.8 | 2.87 | 9.05 | 26.9 | 3.24 | 9.67 | **18.38** | **2** | **6.48** |
| Scenario 5 | 4.1 | 0.3 | 1.83 | 5.2 | 0.36 | 1.96 | 7.4 | 0.56 | 2.6 | 9.5 | 0.72 | 3.07 | 10.7 | 0.74 | 2.99 | **7.38** | **0.54** | **2.49** |
| Scenario 6 | 3.1 | 0.22 | 1.55 | 3.2 | 0.23 | 1.56 | 4 | 0.3 | 1.84 | 4.8 | 0.34 | 1.98 | 5 | 0.33 | 1.85 | **4.02** | **0.28** | **1.76** |
| Scenario 7 | 13.6 | 1.19 | 4.14 | 18.1 | 1.69 | 5.13 | 26.6 | 3.25 | 8.68 | 32.9 | 4.71 | 12.24 | 36.5 | 5.5 | 13.69 | **25.54** | **3.27** | **8.78** |
| Scenario 8 | 5 | 0.35 | 1.96 | 6.5 | 0.45 | 2.19 | 9.8 | 0.74 | 3 | 12.7 | 0.98 | 3.66 | 14.2 | 1.04 | 3.66 | **9.64** | **0.71** | **2.89** |
| Scenario 9 | 4 | 0.29 | 1.76 | 5 | 0.34 | 1.93 | 7.2 | 0.53 | 2.51 | 9 | 0.68 | 2.99 | 9.9 | 0.71 | 2.93 | **7.02** | **0.51** | **2.42** |
| Scenario 10 | 26.4 | 6.56 | 19.63 | 35.7 | 10.25 | 27.21 | 44.4 | 19.45 | 46.85 | 49.8 | 27.97 | 61.19 | 52.6 | 31.55 | 65.57 | **41.78** | **19.16** | **44.09** |
| Scenario 11 | 24.1 | 6.03 | 19.42 | 28 | 9.02 | 26.97 | 32.9 | 17.03 | 44.96 | 36 | 23.67 | 57.46 | 37.1 | 26.54 | 61.56 | **31.62** | **16.46** | **42.07** |
| Scenario 12 | 8.4 | 0.69 | 3.1 | 10.5 | 0.93 | 3.74 | 15.3 | 1.7 | 6.12 | 18.7 | 2.44 | 8.63 | 20.4 | 2.68 | 9.13 | **14.66** | **1.69** | **6.14** |

Same scenarios and metrics as in Table 3.12




(a) Same number of operators

(b) Same shift balancement

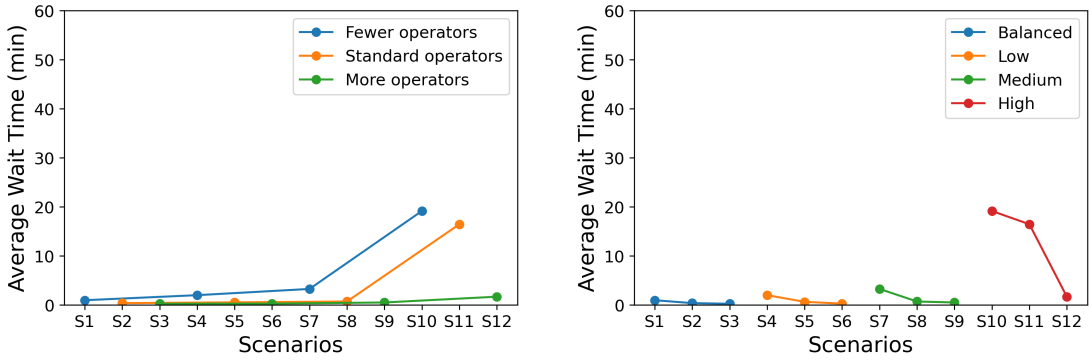
Figure 3.12: Delayed tickets with uniform distributions

This type of investigation can be helpful for projects where team composition and shift distribution are crucial, as it can guide strategic staffing decisions, including recruitment, dismissal, or reallocation of operators to different teams or shifts. Appendix A presents a complete view of the results obtained, with more details regarding each seed and run in each scenario.

## 3.4 Discussion

SNOOKER offers different benefits than the solutions outlined in section 3.1.2 while addressing the takeaways referred to in section 3.1.3. Its core features include generating time series data correlated with geographical information and accurately representing temporal patterns and time-related dependencies. SNOOKER effectively maintains correlations between features while handling numerical and categorical data, thus preserving inherent relationships within the dataset.

To evaluate SNOOKER's applicability in real-world scenarios, this tool was tested in two different sets of tests: a comparison between real data from S21sec and a synthetic dataset generated by SNOOKER, and an analysis of how team dimensionality impacts SNOOKER's performance.




(a) Same number of operators

(b) Same shift balancement

Figure 3.13: Average wait time with uniform distributions

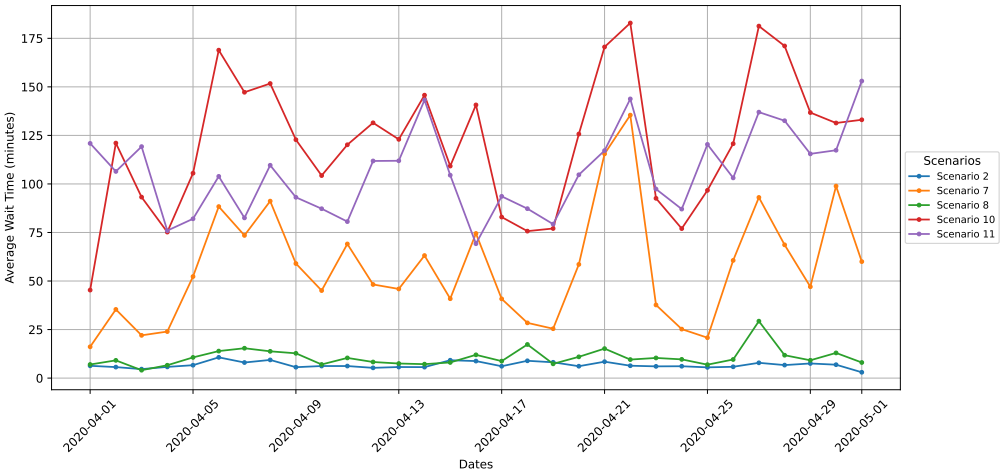



Figure 3.14: Average wait time over one month across scenarios 2, 7, 8, 10, and 11

Regarding the first part of **RQ2** - "Can we generate synthetic data that closely mimics real data regarding distribution and logical relationships?" SNOOKER's results demonstrate that it can simulate the temporal distribution of real data using meta-features such as their annual incident distributions. Additionally, this resemblance is further confirmed by statistical metrics, such as the K-S test, KLD, and Hellinger distance. In terms of the influence of team dimensionality, a higher percentage of delayed tickets is associated with scenarios characterized by fewer operators or shifts with a high imbalance level.

For the second part of **RQ2**, what truly distinguishes SNOOKER is its adherence to business rule logic, which pertains to different types of business tasks common in the context of ticket treatment operations such as ticket prioritization, assignment, scheduling, and other related tasks. SNOOKER mirrors real-world operation phenomena, ensuring the synthetic data remains consistent with real-world limitations and workflows, as discussed in section 3.2.2. It is important to note that all operational knowledge embedded in SNOOKER was provided by S21sec professionals, enhancing the realism of the generated data. Furthermore, this generator avoids privacy issues by relying solely on the meta-features while enabling comparisons between authentic and synthetic datasets regarding incident distribution. Although the discussed tools in section 3.1.2, like SDV, can replicate feature correlation, the proposed methodology introduces novel capabilities by generating time series data correlated with geographical information and adhering to business rule logic.

About the **RQ3** - "What is the impact of different team sizes on ticket handling, measured by the percentage of delayed tickets and average wait time?", we conclude that team size and operator distribution considerably impact their performance in terms of percentage of delayed tickets and average wait time. A smaller team and highly unbalanced shifts tend to increase these metrics' results substantially. The reduced number of operators can explain this when handling the tickets, which, when unevenly distributed, increases the delays, especially during shift transitions. However, the impact of team size and shift distribution is not the only factor impacting these metrics. It also depends on the ticket volume being processed and the selected timeframe. For example, when

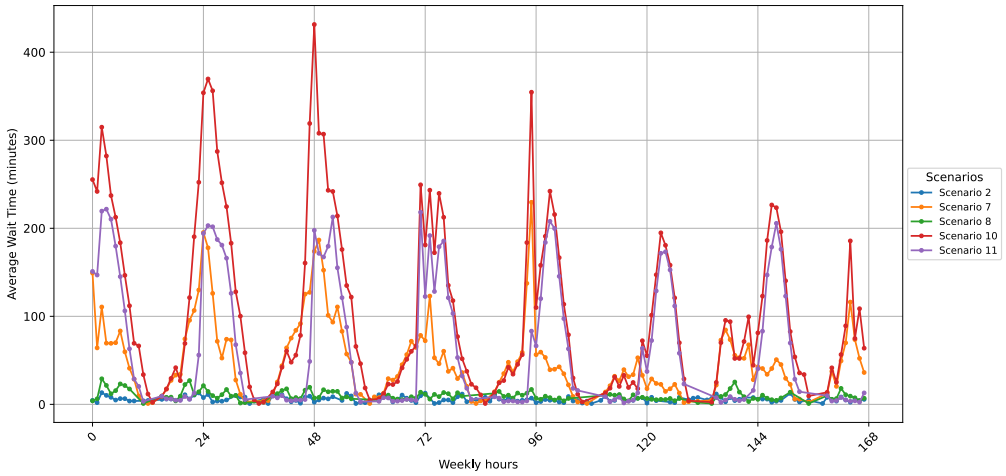



Figure 3.15: Weekly Average Wait time in the same scenarios

experimenting with datasets of various sizes and different timeframes (one year or one month), the ticket handling performance and the resulting wait time fluctuate.

Although this platform simulates cybersecurity data in the tickets, it can, with proper adaptation, be extended to other critical areas, such as healthcare, finance, and other domains. However, such extensions would require SNOOKER to include a wide variety of features, such as domain-relevant terminology, workflows, and temporal patterns, among other characteristics (most of the changes would happen within the dashed modules shown in Fig. 3.3). Section 6.3 addresses this possibility.

# Chapter 4

# CROSS: Context-aware Recommender System

In cybersecurity, each security ticket is unique and influenced by factors such as affected systems, user roles, and other components. The inherent challenges posed by ticket overload and the shortage of SOC professionals highlight the demand for more intelligent and adaptive systems. With their real-time contextual awareness, filtering mechanisms, and inherent adaptability, context-aware recommender system (CARS) solutions have the potential to mitigate information overload, enhance decision-making, accelerate incident response, and provide other benefits [Ferreira et al., 2023].

This chapter explores the concept of RS, outlining its characteristics and types, among other relevant topics in this field. It then presents the state-of-the-art considering different targets, including cybersecurity, context-aware, sequence-aware, and cold-start strategies. Following this, the chapter introduces CROSS (Context-aware RecOmmmender SyStem), an RS developed to support ticket treatment. For each open ticket, CROSS analyzes its context and recommends the fastest combination of operator and procedure to resolve it, prioritizing response speed over recommendation quality. This system tackles complexities associated with cold-start incident types and dynamically adjusts recommendation actions in response to an operator's rejection, among other tasks. In domains characterized by time-sensitive and critical alerts, such as cybersecurity, prompt decisions are vital for minimizing risks and protecting data.[1] Finally, the chapter evaluates and discusses the performance of CROSS in terms of scalability and ticket treatment.

## 4.1 Recommender Systems

RSs are decision support tools designed to provide personalized suggestions to users [Aggarwal, 2016]. These suggestions are derived from the analysis of user preferences or event properties. The acquisition of such data is facilitated either explicitly by eliciting their preferences or implicitly by monitoring the user's activity, such as songs heard, websites visited, and purchase history,

[1] Information available at https://www.forbes.com/sites/forbestechcouncil/2021/01/22/the-importance-of-time-and-speed-in-cybersecurity

among other factors. During the analysis stage, the RS discerns patterns in user behavior and actions, extracts beneficial knowledge, calculates the likelihood that a user expresses interest in a particular item, and then compares these results with the inventory of available items to deliver a custom recommendation [Ricci et al., 2011]. According to Isinkaye et al., such systems have been developed to address the problem of information overload by implementing effective filtering and prioritizing approaches, thereby providing users with helpful suggestions [Konstan and Riedl, 2012, Isinkaye et al., 2015]. Beyond their conventional application in predicting a particular item to users (e-commerce), RSs show versatility in various domains, such as cybersecurity, where they are employed to help experts deal with cyberattacks [Polatidis et al., 2017].

Users rely on recommendations to streamline their searches, making it easier to access content of interest and discover unexpected offers. Meanwhile, businesses aim to provide precise suggestions, aligning with their primary objective of boosting revenue and gaining competitive advantages.

Ricci et al. identified various reasons explaining why several service providers employ RSs in their products [Ricci et al., 2011]:

- Increase the number and diversity of items sold – by understanding the user preferences, companies can sell more items compared to scenarios where no recommendations are given. Furthermore, they can present items that might be challenging to discover, helping users access older or unpopular items;
- Enhance user satisfaction – delivering accurate recommendations in an appealing interface is essential for improving customer experience and satisfaction. This, in turn, increases system usage and the likelihood that users will accept the recommendations;
- Boost user fidelity – recognizing and retaining loyal and long-term users is a vital feature for the improvement of RSs. Systems that refine user recommendations based on previous interactions hold an advantage over the competition, as they can retain customers over an extended period;
- Understand the user preferences - allows an RS to provide the most accurate suggestions and improves system management of item stock.

Recognizing the importance of these goals and balancing user and company needs poses challenges that are yet achievable tasks. With the emergence of increasingly particular user requirements, RSs must adapt and evolve to meet these evolving needs.

RSs have been applied in multiple areas to optimize efficiency, increase sales, and enhance customer experience. Ko et al. identified seven main application fields: e-commerce, entertainment and streaming, social media, education, tourism, healthcare, and academic [Ko et al., 2022]. E-commerce platforms like Amazon employ RSs to engage users and drive sales based on past purchases [Linden et al., 2003]. In the streaming industry, Netflix generates video recommendations tailored to users' viewing history [Gomez-Uribe and Hunt, 2015]. Social media platforms utilize RSs to suggest personalized content based on behavioural patterns and preferences [Shokeen and Rana, 2020]. In education, RSs recommend learning materials to students in online environments [Esteban et al., 2020]. The tourism field also benefits from RSs since they suggest tourism

destinations, accommodations, and other activities related to this field [Abbasi-Moud et al., 2021]. In healthcare, RSs like the one devised by Chen et al. assist with disease diagnosis and treatment planning [Chen et al., 2018a]. In academia, RSs help researchers discover relevant information on past works, citations, conferences, and other aspects [Wang et al., 2018a].

### 4.1.1 Main Characteristics

RSs have various properties that can influence their application and user experience. Developers must consider potential trade-offs between the properties outlined below when designing these systems. For instance, increasing data diversity may reduce the accuracy of recommendations and predictions. RSs are dependent on various aspects [Shani and Gunawardana, 2011, Ricci et al., 2011, Mohamed et al., 2019, Roy and Dutta, 2022]:

- Prediction Accuracy – measures how well the system's predictions match the expected behaviour. This is one of the most widely discussed properties in RSs, as it is related to the core goal of predicting ratings, actions, or other events. Users prefer systems with more accurate predictions, as higher accuracy builds trust, confidence, and user engagement;
- Diversity – assesses the range and variability of the suggestions provided by the system, allowing the RS to focus on diverse solutions. In most cases, users exhibit a preference for recommendations characterized by a higher intra-list diversity driven by the fact it allows more extensive exploration of items in a shorter time, improving user satisfaction;
- Serendipity – measures the degree of surprise in successful recommendations, which can improve user satisfaction [Sridharan, 2014]. For instance, a customer visiting an online store to buy a laptop might receive a suggestion for noise-canceling headphones, even though the customer did not search for them. This unexpected suggestion could lead to an extra purchase, benefiting both the store and the customer;
- Privacy – the user's preferences must remain private and safe to prevent unauthorized access, misuse, or exploitation by third parties. RSs must collect, store, and process private data in compliance with privacy legislation to maintain user trust;
- Robustness – refers to the system's capacity to maintain performance under diverse scenarios, including fake information or system failures. Attackers exploit various attack types (grey sheep [Sharma and Gera, 2013] and shilling attack vectors [Zhou et al., 2018b]) to inject mistrusted data and manipulate the recommendations provided to users;
- Utility – concerns the value derived from a recommendation by either the system or the user. From the system's perspective, this value may translate into increased revenue, while for the user, it can result in a product they may buy;
- Adaptivity – refers to the ability of an RS to operate in dynamic environments while presenting relevant suggestions. Analyzing, comprehending, and adapting to emerging trends is crucial but not exclusive to RSs. For example, when changes occur in the item collection (price and characteristics), the platform must promptly adapt its recommendations. Despite implicit requests being essential for rendering a system more proactive, recognizing them is

challenging since it requires the system to predict what to recommend and when and how to push recommendations actively;

- Scalability – associated with the RS's capacity to handle large data volumes without sacrificing performance. RSs aim to deliver rapid and accurate recommendations, but increasing data volumes can strain computational resources and memory. Scalable systems maintain reliable suggestions even as the workload grows;
- Cold-Start – occurs when a new user or item is introduced to the system. Depending on the recommendation approach, it can be difficult to recommend items to a user with unknown interests or suggest items that have not been rated. Section 4.2.4 addresses several strategies presented by the research community to tackle this issue;
- Sparsity – intrinsic to the lack of information about transactions or feedback data, it poses a challenge in providing accurate suggestions, mainly when the number of rated items is smaller than the number of ratings to predict for a user.

Chapter 4 offers further details on how the proposed RS addresses some of these factors.

### 4.1.2 Types of Recommender Systems

With the vast amount of information available online, RSs play a key role in supporting decision-making and mitigating information overload by analyzing user behaviour and interactions. Over the past decades, numerous recommendation techniques have been developed with notable success, especially in e-commerce [MacKenzie et al., 2013].

Burke introduced a taxonomy consisting of five types of recommendation techniques [Burke, 2002]: content-based, collaborative filtering, demographic, knowledge-based, and hybrid. Building upon this work, Golbeck further enriched the field by introducing a community-based method, which integrates feedback and preferences from users' social networks to augment the recommendation accuracy [Golbeck, 2006]. Figure 4.1 illustrates a classification encompassing relevant groups, proficiently providing users with customized suggestions.

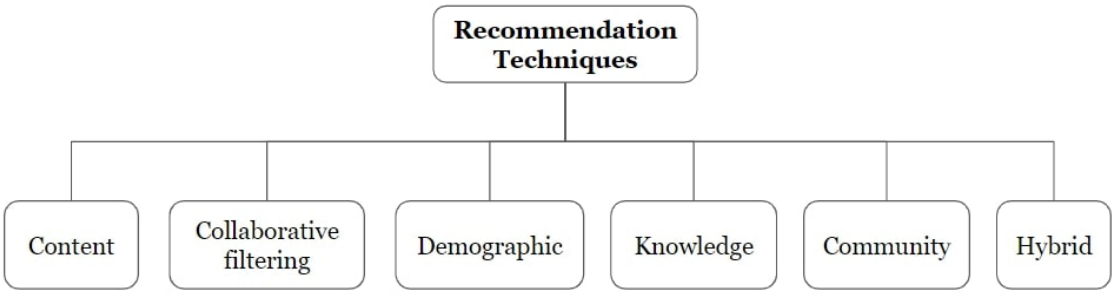



Figure 4.1: Recommendation Approaches

Traditional RS methodologies, which disregard the sequential order of user interactions and rely solely on user-item interaction data, can be extended by sequence-aware and context-aware

techniques [Adomavicius et al., 2011, Kumar et al., 2021]. Such approaches add more meaning and relevance to the recommendations by incorporating contextual information and temporal patterns in user behaviour (detailed in sections 4.1.2.7 and 4.1.2.8).

#### 4.1.2.1 Content-based Filtering

Content-based (CB) filtering methods builds recommendations based on the characteristics of items and the user's profile preferences. In this context, this approach employs the item's attributes to formulate suggestions. By analyzing the item's content and aligning it with the user's preferences, CB suggests items similar to those the user has liked in the past. The determination of item similarity is contingent upon the features inherent in each item. Usually, the system builds a CB user profile, represented as a weighted vector of event features. The weights assigned to these features signify their importance to the user. They can be computed from simple approaches (using the average values of the rated item vector) or ML methods, including Bayesian classifiers, cluster analysis, DTs, and NNs. Figure 4.2 illustrates the recommendation flow in CB systems.

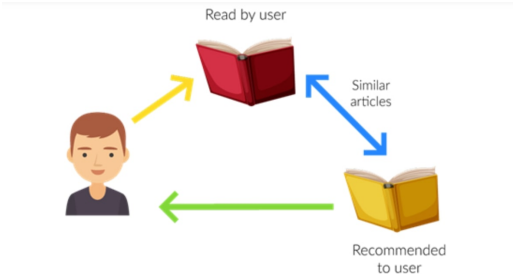



Figure 4.2: CB Scenario[2]

The user's preference regarding the red book is crucial in this example. As the red book shares identical features with the yellow book, there is an increased likelihood that users will find the yellow book intriguing. Consequently, based on this correlation, the yellow book is recommended to the user.

As illustrated in Fig. 4.3, this type of system is primarily characterized by the presence of the three modules:

- Content Analyzer – explores and classifies the items based on their description and features, employing several feature extraction tools. Its relevance is particularly relevant in scenarios where information lacks a structure;
- Profile Learner – collects data on the user preferences and formulates generalized user profiles by inferring preferences with ML techniques;
- Filtering Module – analyzes the similarity between the inferred user profiles and item attributes, culminating in generating recommendations.

[2]Figure available at https://recosenselabs.com/blog/content-recommendations-media-websites

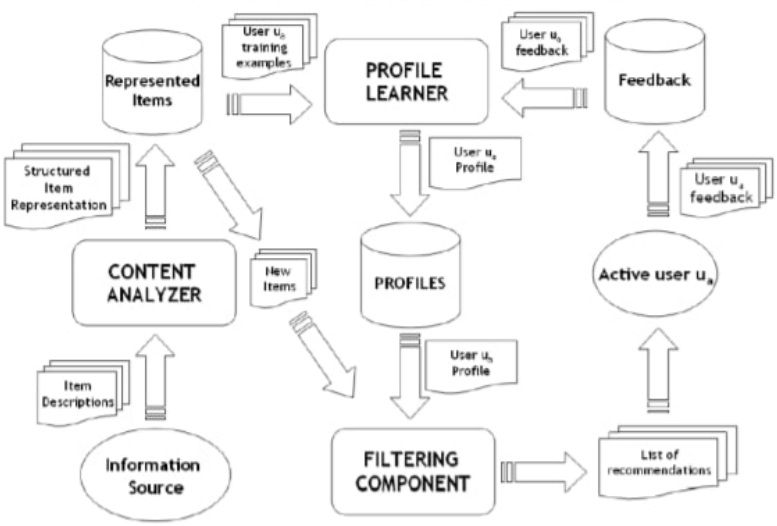



Figure 4.3: Content-based Architecture Example [Koene et al., 2015]

These systems exploit the content of data items, extracting textual features from multiple sources to generate recommendations based on similarity metrics. They comprise two approaches: information retrieval and filtering methods, and ML algorithms.

In information retrieval, bag-of-words techniques are used for constructing dictionaries or word models, where each bag contains a subset of words from the dictionary. A well-known method is the term frequency-inverse document frequency model. In this model, term frequency measures the frequency of a word in a document. In contrast, inverse document frequency reflects the inverse of the document frequency among the whole corpus of documents that contain the word [Salton and McGill, 1986]. This approach transforms users and items into weighted term vectors stored in a vector space model. This spatial representation exists in an n-dimensional space, enabling the computation of angles (such as Cosine similarity) to determine the similarity between vectors.

From an ML standpoint, the recommendation process can become more efficient, but also more complex and computationally demanding. ML-based approaches build models capable of categorizing new information items and exploiting data explicitly or implicitly labeled as interesting or uninteresting by the user. In these models, features represent attributes of users and items, while the response variable indicates the user's preference for a given item. ML algorithms create predictive models that estimate the likelihood of a new item capturing the user's interest. Bayesian classifiers, naive Bayes, DTs, NNs, and other probabilistic approaches constitute viable solutions to this challenge.

Although they generate recommendations solely based on the active user's ratings and unrated items, these systems have limitations, including limited content analysis, over-specialization, difficulty estimating preferences for new users, and challenges in capturing certain associations between users and items.

#### 4.1.2.2 Collaborative Filtering

Collaborative filtering (CF) suggests items to a user based on the preferences of other users with similar tastes, transaction histories, and behavioral patterns [Sharma et al., 2017]. It operates on the assumption that users who agreed in the past will likely manifest the same behaviour in the future, continuing to favor comparable items. User similarity can be quantified using algorithms, including K-NN and the Pearson Correlation, which measure the linear relationship between two variables. CF systems are characterized by their simplicity, ease of understanding, and ability to provide accurate recommendations without requiring content analysis or knowledge of the items. They may also enable diverse and serendipitous recommendations. Nonetheless, it struggles with sparsity, cold-start problems, and balancing the trade-off between scalability and prediction accuracy. Service providers like Netflix and Amazon employ CF-based RSs to enhance user retention. Figure 4.4 depicts the recommendation mechanism within this RS type.

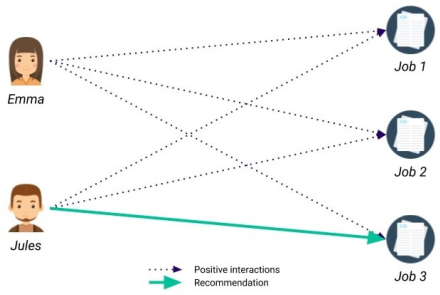



Figure 4.4: CF Scenario[3]

In this illustration, users Emma and Jules participated in jobs 1 and 2, and Emma was also involved in job 3. Under CF principles, we infer that Emma and Jules possess similar preferences. Consequently, we recommend job 3 to Jules.

CF techniques are organized into three groups based on their impact on the user database: memory-based, model-based, and hybrid-based. Memory-based or neighbourhood-based algorithms retain all user-item interactions within a database, typically organized in a matrix $N \times M$, where $N$ represents the number of users and $M$ the number of items. These methods compute the similarity between users or items to facilitate prediction tasks [Breese et al., 1998]. Within this category, methods can be further divided based on the focus of the analysis, resulting in subcategories such as user-based nearest neighbour and item-based nearest neighbour.

The user-based approach recommends items to users with similar interests [Sarwar et al., 2001]. This methodology assesses similarities between a particular user and other users, identifies the top similar users, and calculates a weighted average of their ratings, using these similarities as

[3]Figure available at https://www.welcometothejungle.com/en/articles/collaborative-filtering-job-recommendations

weights. Similarity is measured using metrics, such as the Pearson correlation-based metric, cosine-based similarity, or other adjusted cosine-based measures (discussed in section 4.1.3). Figure 4.5 exemplifies the implementation of user-based CF.

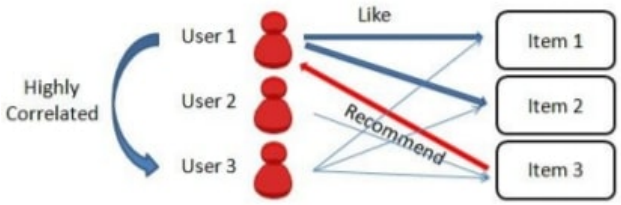



Figure 4.5: User-Based CF Scenario [Arekar et al., 2015]

In this example, the system inferred a similarity in interest between users 1 and 3, as the latter provided ratings for all items while user 1 only rated items 1 and 2. Consequently, item 3 was recommended for user 1.

Item-based techniques differ from user-based methods by focusing on item ratings instead of user preferences [Sarwar et al., 2001]. Two items are considered similar if they received comparable ratings from the same user. Suggestions are subsequently generated by computing the weighted average of scores for the most similar items from a particular individual. Figure 4.6 shows the recommendation process in an item-based example.

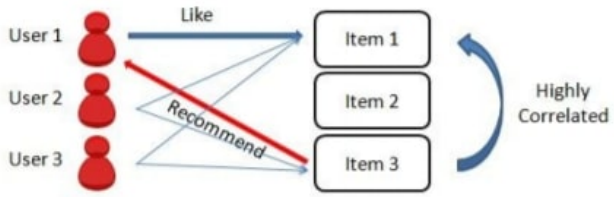



Figure 4.6: Item-Based CF Scenario [Arekar et al., 2015]

In this instance, users 2 and 3 rated items 1 and 3, leading the system to infer a similarity between these items. Given that user 1 liked item 1, and considering the perceived similarity between items 2 and 3, the system recommends item 3 to operator 1.

Model-based techniques are advanced tools developed to overcome the scalability and sparsity issues inherent in memory-based methods [Breese et al., 1998]. Unlike other approaches that depend on the user interactions with items, model-based approaches create predictive models from the user database. Within this classification, latent-factor approaches represent a subset characterized by models that operate in a reduced or reorganized feature space relative to the original user or item vector space. This dimensionality addresses challenges such as sparsity and space-related issues, computational speed, and noise reduction [Sarwar et al., 2000]. The choice of dimensionality reduction techniques depends on the nature of the data. For numerical data, methods such as matrix factorization (MF) [Bokde et al., 2015], singular value decomposition (SVD) [Luo

et al., 2015], regression, and PCA, among other algorithms, are commonly used. For categorical data, optimal compression involves using classification methods such as Bayesian networks [Breese et al., 1998] and clustering [Ungar et al., 1998].

MF is an ML technique used for latent variable decomposition and dimensionality reduction [Bokde et al., 2015]. This approach maps users and items into a shared latent factor space of dimensionality *f*, representing user-item interactions as inner products within the latent space. Recommendations are generated based on the degree of alignment between the user and the product in this latent space. For example, if a user consistently assigns high ratings to supernatural series, this pattern reflects a broader interest in the supernatural genre rather than specific opinions on individual series. Unlike specific series, latent features encapsulate higher-level attributes, with the paranormal category as one latent feature in this context. MF is crucial for assessing how well a user aligns with various latent features and gauging the compatibility of a series with these latent features. One of the key advantages of MF is its ability to identify similarities between users even when they have not rated identical series.

Combining memory-based and model-based techniques in a hybrid structure has proven effective in providing superior recommendations and addressing the limitations inherent in native CF algorithms [Kumar and Reddy, 2014]. Many systems employ hybrid methods to attain heightened accuracy and improved recommendation prediction. For instance, the Google News RS utilizes two model-based algorithms: MinHash clustering and probabilistic latent semantic indexing, and one memory-based method grounded in item-item visitation based [Das et al., 2007].

Overall, CF approaches are straightforward to implement and easily incorporate new data. They do not rely on the content of the recommended items, scale efficiently with co-rated items, and provide explainable results. However, they are limited, as they depend on human ratings, the performance declines with increased data sparsity, items cannot be recommended to new users, and scalability diminishes with large datasets.

#### 4.1.2.3 Demographic-based

Demographic-based RSs utilize users' demographic attributes, including gender, age, language, health status, mobility patterns, employment situation, geographical location, and other demographic data, to generate personalized recommendations. Highly applicable to content-aggregation websites and e-commerce solutions, demographic systems show stereotypical attributes by assuming that all users within a demographic group share similar tastes and preferences [Callvik and Liu, 2017]. Considered a more straightforward yet niche option than previous algorithms, this method class exclusively considers user-related aspects, necessitating thorough market research within the specified region. Accompanied by surveys, this methodology gathers and uses data for categorization in the recommendation task.

It is noteworthy that demographic RSs often yield suboptimal results and raise security and privacy concerns. However, they enhance the efficacy of other RSs when integrated as a component within hybrid or ensemble models and are immune to cold-start problems.

#### 4.1.2.4 Knowledge-based

Knowledge-based RSs provide suggestions based on user-specified requirements rather than relying on the user's purchase history [Aggarwal, 2016]. These systems query users systematically about their interests and combine that information with domain knowledge to formulate valued suggestions. A similarity function estimates user needs and presents recommendations that align with the user's utility. They have applications in industry [Klöber-Koch et al., 2017], education [Tarus et al., 2018] and healthcare [Song et al., 2021]. The knowledge base can be a simple database, a domain ontology, or a case-based knowledge structure [Liao, 2005]. Understanding the user profile is essential for delivering tailored recommendations, including attributes such as tastes, interests, needs, and other properties [Bouraga et al., 2014].

Based on interface type, this RS type can be divided into two main categories: constraint-based and case-based systems. Both strategies share identical architectures for knowledge representation, involving acquiring user requirements, resolving conflicting preferences, and explaining suggestions. However, they differ in how they generate solutions. A third category, utility-based systems, can be viewed as a third group [Aggarwal, 2016].

Constraint-based RSs use rule-based methodologies to deliver suggestions to users by considering explicitly defined constraints on item attributes, such as filters or incompatibility constraints [Felfernig and Burke, 2008]. Such systems rely on domain-specific rules to match user requirements with item features, ensuring recommendations align with the specified constraints. When no suitable items are found, the system can make minimal adjustments to the user's requirements, enabling the presentation of feasible solutions. For example, Felfernig et al. developed a financial services recommendation system with mechanisms to repair unfeasible user requirements [Felfernig et al., 2007].

Case-based RSs (CBRSs) recommend products based on their similarity to those explicitly indicated by users [Lorenzi and Ricci, 2005]. Each item is treated as a case characterized by multiple attributes, and similarity metrics are employed to retrieve items akin to those outlined by users. Unlike constraint-based RSs, CBRSs do not impose strict limits, such as minimum or maximum values. Furthermore, these systems incorporate critiquing methods [Chen and Pu, 2012], which help users refine their requirements. Whether the system is conversational, search-based, or navigation-based, it identifies products that meet the user's requirements, presents feedback through critiques, and allows users to adjust their preferences for more precise recommendations.

CBRSs are occasionally mistakenly compared to utility-based tools because both focus on recommending items based on utility. However, they differ in how they define utility. Utility-based RSs prioritize the weighting of user preferences and incorporate value trade-offs as an essential aspect of the decision-making process. In contrast, CBRSs do not incorporate such weighting in their approach or prioritize value trade-offs.

A key element explored in CBRSs is case-based reasoning, a problem-solving approach that integrates ML to tackle novel challenges by reusing past experiences stored in example cases [Ketler, 1993]. Each case represents a specific problem, including its interpretation and solution,

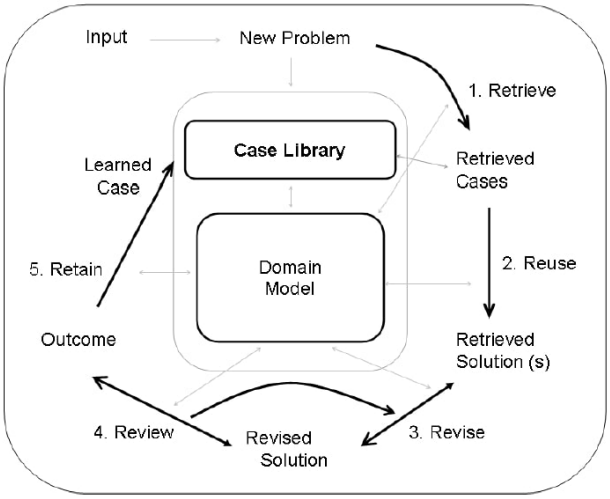



Figure 4.7: Case-based Reasoning Cycle [Ketler, 1993]

and is archived in a case base. This repository enables the system to learn from previous cases and generate informed decisions for future problems. As depicted in Fig. 4.7, the case-based reasoning process involves four steps: retrieve, reuse, revise, and retain [Aamodt and Plaza, 1994]. When a new problem arises, the system begins by retrieving the most similar case(s) from its repository. In the reuse phase, the RS applies the knowledge and solutions from past cases to the current problem. During the revision step, the effectiveness of the proposed solution is evaluated, and any necessary adjustments are made. Finally, in the retain stage, the refined solution is stored as a new case, enriching the system's knowledge base for future use.

CBRSs offer several benefits, such as immunity to cold-start or sparsity issues and a clear understanding of how specific items meet user needs without relying on user ratings. However, they also face challenges such as knowledge acquisition bottlenecks, difficulty identifying niche user preferences, inflexibility, rigid modification processes, and limited learning capabilities.

#### 4.1.2.5 Community-based

Community-based approaches, also known as social RSs, use the opinions of other users, particularly those within social networks. It follows the assumption that recommendations emanating from social networks are more efficient and trustworthy than those based solely on similarity measures [Sinha and Swearingen, 2001]. By analyzing various social networks, such as Facebook and Twitter, these systems generate recommendations for a user based on the preferences of their friends and social groups. Research offers diverse perspectives on the efficiency and performance of these systems. Some studies suggest that their accuracy is comparable to standard CF approaches, except in cases where user ratings for a specific item vary greatly [Massa and Avesani, 2004]. On the other hand, other analyses highlight that community-based methodologies often outperform similarity-based metrics [Guy et al., 2009].

While many solutions focus on a single dimension of interactions within social networks, this limitation can be addressed through principal modularity maximization [Li and Tang, 2016]. This method improves community partitioning in real-world networks by considering the degree distribution of nodes. The core idea of integrating network information from multiple dimensions is to unveil cross-dimension group structures. This approach begins with modularity analysis to extract structural features from each network dimension, and then it uses these features to discover the community structure among nodes.

One of the key strengths of these methods is their ability to take advantage of a broader range of information compared to other approaches, which can enhance recommendation performance. However, a significant limitation is their reliance on data from social networks, which may sometimes be inaccurate or false.

#### 4.1.2.6 Hybrid-based

Hybrid RSs combine the strengths of various recommendation approaches into a single framework, effectively addressing some of the limitations outlined earlier. By integrating different algorithms, such as CB with CF, hybrid RSs can provide more accurate recommendations [Schafer et al., 2007]. For instance, by combining techniques A and B, each with distinct strengths and weaknesses, a hybrid system can mitigate the shortcomings of both methods while capitalizing on their respective advantages. Burke identifies various hybridization techniques for the creation of a hybrid system, such as [Burke, 2002]:

- Weighted [Mobasher et al., 2004] – the rating of a recommended item is calculated through the result's aggregation from all the recommendation techniques used;
- Switching [Smyth and Cotter, 2000] - the system dynamically switches between different recommendations according to the user preference or certain criteria;
- Mixed [Billsus and Pazzani, 2000] – instead of providing a single recommendation per item, it merges the recommendations of different recommender approaches. It is beneficial when a large number of recommendations are required simultaneously;
- Feature Combination [Zanker and Jessenitschnig, 2009] – gathers features from different knowledge sources and combines them to provide a unified recommendation algorithm;
- Feature Augmentation [O'Sullivan et al., 2004] – employs the rating of an item produced by one technique as an input feature of another recommendation method;
- Cascade [Burke, 2002] – produces an unrefined ranking of candidates using one recommendation technique and then refines the recommendation from among the candidate set using another approach. The resulting recommendation list prioritizes items with high ratings appearing first, followed by items with lower scores;
- Meta-level [Pazzani, 1999] – the model generated by one recommendation tool serves as input for the following method. Instead of using a learned model to generate features for input to a second algorithm (feature augmentation), the entire model becomes the input.

Employing a hybrid RS is a practical approach for building and optimizing a system. This method exploits the strengths of various recommendation techniques to address issues such as sparsity, cold-start, and other problems. For example, CF encounters challenges when confronted with new items lacking ratings upon introduction into the system. Applying the principles of content-based approaches offers a viable solution to this issue, as predictions for new items are derived from their features instead of relying on ratings. However, selecting the appropriate hybrid approach requires understanding the application scenario and the desired system features. Furthermore, its implementation can be costly and complex due to integrating distinct tools.

As discussed earlier, most RS categories, such as CB, CF, and hybrid-based, utilize ML algorithms to generate user recommendations. In contrast, the other RS groups rely on more straightforward methodologies for generating pertinent suggestions, which can be enhanced with supplementary data mining and ML techniques.

Various well-known companies utilize hybrid recommendation platforms, including Amazon, Netflix, and YouTube.

Amazon[4] utilizes a hybrid RS item-to-item CF with market basket analysis, also known as affinity analysis, a data mining subset used to identify correlated product pairs [Linden et al., 2003]. As discussed in section 4.1.2, CF offers simplicity, scalability, and explainable suggestions. Linden et al. elucidated that item-to-item CF approaches ascertain matches between user-purchased and rated items, consolidating akin items into potential suggestions [Linden et al., 2003]. This process involves formulating an item-to-item matrix, where pairwise similarities (via cosine measure) are calculated to discover the best matches for each item. Although CF may perform sub-optimally in terms of memory usage and processing time in scenarios devoid of identical customers, Amazon overcomes this challenge by using its offline computation capabilities, effectively addressing concerns related to dimensionality and scalability. Since the early 2000s, Amazon's algorithms have significantly evolved, with the temporal aspect being one area of improvement. In dynamic scenarios, where product catalogs shift constantly, time is critical in ensuring recommendation quality. For instance, buying a shirt seven months after acquiring another shirt yields different implications for recommendations compared to simultaneous purchases.

Figure 4.8 provides an illustrative representation of the sequential stages involving Amazon's recommendation process, since the creation of the similarity table (item-to-item CF) to the formulation of personalized recommendations. Step 108 emerges as the pivotal task where the RS undertakes affinity analysis within this schematic. Fundamentally, this step discerns pairs of products with large overlaps in their customer base, empowering the system to suggest items to customers who may have exhibited partial interest in certain items.

Smith and Linden extensively explore Amazon [Smith and Linden, 2017], exploring its historical development, recommendation techniques exploited, and future research lines, such as the ability to shop with intelligent conversational systems.

[4]Website available at https://www.amazon.com/

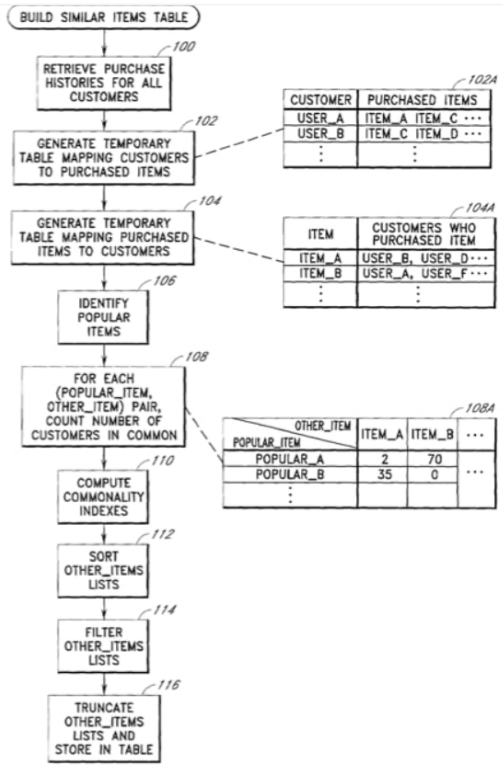



Figure 4.8: Amazon Recommendation Flowchart [Altitude Labs, 2017]

Netflix[5] employs knowledge-based and utility-based approaches, augmented by CF, to determine the most relevant content for each viewer. As stated by Gomez-Uribe and Hunt, the Netflix RS integrates various algorithms, including the personalized video ranker, top-n video ranker, trending ranker, continue watching ranker, video-video similarity, page generation, and evidence selection [Gomez-Uribe and Hunt, 2015]. The recommendations generated by Netflix consider a variety of inputs, including user interactions, such as viewing history and ratings assigned to titles; preferences of similar users; title metadata, encompassing genre, categories, actors, release year, and classification, among others; and contextual factors, such as time of day and device type. Figure 4.9 exemplifies multiple recommendations displayed within the Netflix interface. These suggestions include new, trending releases and series based on the user's past viewing history.

Recognizing the influence of recommendations on revenue and user engagement, Netflix initially prioritized optimizing its services over global expansion. To fine-tune their RS, Netflix employs A/B testing, detailed in section 4.1.3. These tests help evaluate customer retention across different audience segments, including existing and new members. By randomly assigning users to various experimental groups, or cells, Netflix can observe distinct user behavior and pose targeted inquiries to these groups. Each new experience is considered an improvement if it leads to increased engagement with Netflix products and a higher retention rate.

[5]Website available at https://www.netflix.com/pt/

[6]Figure available at https://medium.com/netflix-techblog/interleaving-in-online-experiments-at-netflix-a04ee392ec55

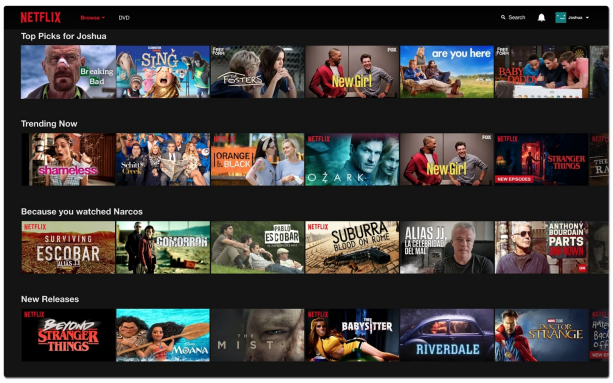



Figure 4.9: Netflix Recommendations[6]

YouTube[7] employs a hybrid RS, integrating knowledge-based, CF, demographic, and NN approaches. DL is a crucial component in YouTube [Covington et al., 2016], supported by Google's AI research, particularly Google Brain [Dean et al., 2012], which has evolved into TensorFlow [Abadi et al., 2016]. As illustrated in Fig. 4.10, YouTube's RS generates suggestions through two interconnected networks:.

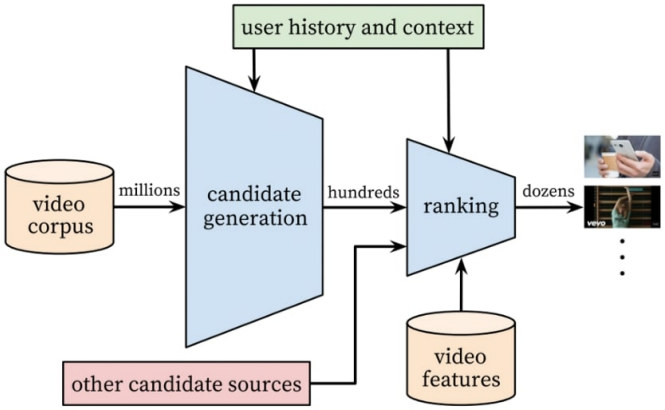



Figure 4.10: Neural Network YouTube Algorithm [Covington et al., 2016]

The candidate generation network first aggregates user activity data, including videos watched, search query tokens, interests, and demographic information. With CF techniques, this network recommends a small subset of videos and a vast corpus, ensuring precision and relevance. Next, the ranking network evaluates these recommendations by analyzing various video features using an objective function and logistic regression. The video with the highest score is recommended to the user. Both stages are refined through extensive offline and online testing. Offline metrics, such as precision, recall, and others, are used alongside online methods like A/B testing (explained in

[7]Website available at https://www.youtube.com/

section 4.1.3) to optimize performance. Covington et al. investigated YouTube's internal works, and the evaluation process followed [Covington et al., 2016].

Despite being one of the most powerful recommendation engines, YouTube is not immune to drawbacks that affect the diversity and relevance of its suggested content [Shiligin, 2019]. A common issue is the tendency for recommendations to closely mirror previously watched videos, which can limit novelty and variety. The platform's recommendation approach, focused on matching user preferences, may inadvertently restrict content diversity. Moreover, features like the "Not Interesting" button fail to contribute to the recommendation process.

Besides these companies, several other organizations utilize RSs to achieve their goals, such as Facebook[8] or Spotify[9]. Sharma and Singh showcase more cases of companies with recommendation techniques [Sharma and Singh, 2016].

#### 4.1.2.7 Context-aware RSs

Context-aware RSs (CARSs) use specific contextual data to enhance the accuracy and relevance of suggestions by considering factors that influence user interests, such as social status, time of the day, and other aspects. Context is "any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves.". For example, the user's music interests may vary depending on the time of the day and weather conditions. Contextual information can be classified in terms of the level of knowledge RSs have about this data and their adaptability over time. Regarding the first aspect, context can be categorized into three classes:

1. Fully observable – the contextual data, their structure, and values are known explicitly;
2. Partially observable – only a partial number of context factors is known explicitly. This group can be further divided into different levels of partial observation;
3. Unobservable – no information about the context is available to the RS. Utilizes only latent knowledge in the recommendation task.

In terms of continuous improvement, the context can be static when its structure remains static over time or dynamic when it presents a sense of evolution. Once the pertinent contextual information is identified, they can be integrated into RSs with the support of the following contextual approaches [Adomavicius et al., 2011]:

1. Pre-Filtering – context aspects are applied in selecting the relevant set of data that the RS utilizes. It allows the reduction of the candidate set size, potentially improving the speed of the recommendation process;
2. Post-Filtering – ratings are predicted with the RS, and the context filters the results. Similarly to the previous group, it can use various traditional recommendation methods;

[8]Website available at https://www.facebook.com/
[9]Website available at https://www.spotify.com/pt/

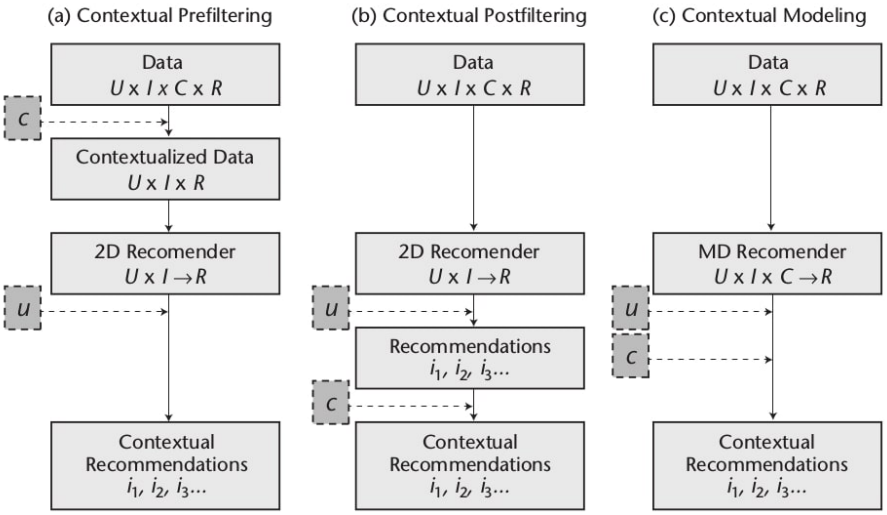



Figure 4.11: CARSs Techniques [Adomavicius et al., 2011]

3. Modeling – the context is incorporated in the recommendation model with methods such as SVM, Markov models, and matrix factorization.

Figure 4.11 illustrates the theoretical foundation of including the context factor at different stages of the recommendation process. In the diagram presented, "U" represents the user, "I" denotes the items, "C" signifies the context, and "R" indicates the rating.

#### 4.1.2.8 Sequence-aware RS

Traditional techniques focus on modeling user-item interactions to produce recommendations. However, contemporary user dynamic behaviour exhibits dynamic patterns, promoting an interest in sequential interactions observed over time. Sequence-aware RSs, sometimes referred to as sequential RSs, address this concern by analyzing short- and long-term preferences to recommend items/actions aligned with a given sequence of actions [Quadrana et al., 2018]. While the terms "sequence-aware" and "sequential" are sometimes used interchangeably, some argue that sequence-aware represents a specialized subset of CARSs, as they incorporate short-term preferences as a form of context data [Quadrana et al., 2018]. These frameworks find application in several domains, such as music recommendation, e-commerce, learning platforms, advertisement, and other fields. Figure 4.12 presents a schematic overview of sequence-aware RSs, delineating the input-output flow.

These frameworks typically process timestamped ordered lists of past user actions to perform various operations, such as analogous item search, complementary search, list continuation search, logically-next item search, trend detection, and repeated recommendations. The output entails ordered lists of actions/items tailored to user preferences. Various algorithms have been explored on sequence-aware RSs, including:

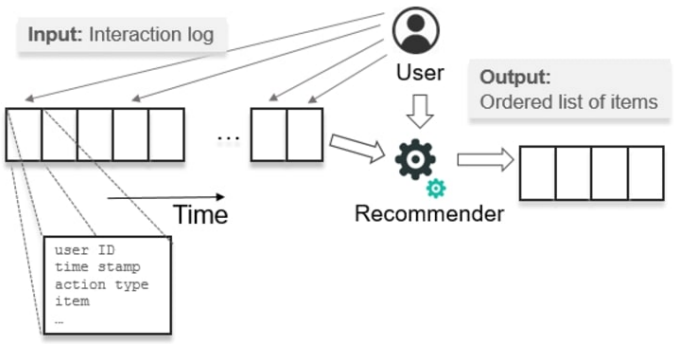



Figure 4.12: Sequence-Aware RS Cycle [Quadrana et al., 2018]

- Frequent pattern learning [Sohrabi and Roshani, 2017] – identifies recurring sequences of user interactions or item transactions within large databases. These recurrent events are categorized as either unordered patterns or sequential patterns (SP) and are analyzed using techniques like association rule mining, sequential pattern mining (SPM), and continuous sequential pattern mining approaches;
- Sequence modeling [Hidasi et al., 2016] – involves developing predictive models that consider past observational data to forecast future events or actions. This is achieved through methods like Markov models and RNN variants (LSTM, Gated Recurrent Unit, and others);
- Distributed item representations [Feng et al., 2015] – facilitates the representation of items in a distributed and continuous vector space, preserving sequential relationships between items. It operates similarly to latent factor approaches such as Latent Dirichlet Allocation (LDA);
- Supervised learning with sliding windows [Wang et al., 2015a] – segments sequential data into sliding windows of size $W$, each representing a subset of user-item interactions. These windows are treated as input-output pairs for training predictive models, with features extracted from the window used to predict the next item or user behavior;
- Matrix completion [Candes and Recht, 2012] – although the sequence-aware recommendation problem differs from standard matrix-completion, MF techniques can still be applied in this context. Sequential data is represented within a user-item interaction matrix, and MF methods are employed to infer missing entries in it;
- Hybrid approaches [He et al., 2016] – combine sequence learning techniques with MF methods to mitigate issues, such as data sparsity and other inherent problems in the sequence-aware recommendation, and improve recommendation efficiency and robustness;
- Others [Xu et al., 2016] – graph-based and discrete optimization (with weak and strict ordering constraints) algorithms are some examples employed within this area.

Sequence-aware RSs, while adept at capturing temporal dynamics and sequential patterns in user interactions over time to enhance recommendation accuracy, encounter several inherent challenges. Among these limitations is the effective handling of long user-item interaction sequences with flexible order, noise, and heterogeneous relations [Wang et al., 2019b].

In the field of research, various groups actively explore RSs, including DART (Data Analytics Research Team, USA), MIT (Massachusetts Institute of Technology, USA), and DSRS (Decision Support and Recommender Systems, UK), among other research teams. Key conferences where advancements in RSs are discussed include RecSys (ACM Conference on Recommender Systems), IUI (Intelligent User Interfaces), and groups and forums.

### 4.1.3 Evaluation

System evaluation is critical and depends on its design. Effectively comprehending the recommendations' quality and adequately assessing RS's predictions constitute primary challenges within this area [Silveira et al., 2019]. According to Shani and Gunawardana, three levels of experiments can be conducted for comparing various RS [Shani and Gunawardana, 2011]:

- User studies - a small group of individuals is recruited to engage with an RS. While they perform several tasks, their experience is recorded and analyzed using quantitative and qualitative information. The retrieval of this type of data contributes to the enhancement of the RS. User studies identify issues that offline experiments might have overlooked, such as aspects related to the quality of the user interface or ease of performing specific tasks. Although they provide useful results, user studies regarding time (scenarios are repeated multiple times) and compensation (especially if the participants are not volunteers) are resource-intensive. Pu et al. outlined three critical interactions between the user and the system concerning this topic [Pu et al., 2012]: the initial preference elicitation process, the preference refinement process, and the presentation of the system's recommendation results;
- Offline evaluation - the most straightforward method for experimentation since it doesn't require genuine user involvement and system deployment. In this setting, algorithms can be compared and tested with a dataset at a low cost. The primary objective is to filter and discard approaches that fail to provide robust predictions. This testing phase aids in understanding user behaviours when interacting with an RS, narrowing down the recommendation techniques explored by the following evaluation methods. Despite their utility in the early stages of development, they can only address a limited subset of questions, particularly those related to prediction power;
- Online evaluation - also known as A/B testing, is the closest method to reality and is considered highly reliable. In an online experiment, the system generates recommendations and presents them to a real user (unaware that it is being analyzed), with the evaluation based on the items the user consumes or likes. This real-time technique measures the changes in user behavior towards different RSs, that is, which features influence user interest and how one system can be considered superior. Despite being the most insightful approach, implementing it on live systems can be time-consuming, hard to implement, and may even harm retailer's reputation if suboptimal recommendations are presented to users. Peska and Vojtas presented an interesting comparison between offline and online methods in small e-commerce scenarios [Peska and Vojtas, 2018].

As observed, the most effective approach to evaluating an RS involves mixing these methods at different stages of development. Initially, offline techniques are employed to grasp the problem and discard unsuitable strategies. Then, user studies and online methodologies are applied to enhance and supplement the evaluation task.

One of the most critical stages in RSs is identifying similar items (events, tickets, rules, actions, among others). This process allows users to access meaningful suggestions and mitigate the impact of information overload. Numerous methods explained previously employ similarity measures to assess the degree of similarity. For example, in CF, user feedback is used as ratings to discern similarities and differences among user profiles and generate personalized recommendations. Some of the exposed RS approaches can use the following similarity measures [Sondur et al., 2016]:

- Euclidean distance (Eq. 4.1) – computes the distance between two points in a multidimensional space. This metric calculates the user similarity in the RS domain based on the number of commonly rated items. This involves computing the rating score $x_i$ and $y_i$ (varies between 0 and 1) of an item given by two different users for the same item. The number of dimensions is given by $n$.

$$d(x,y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2} \tag{4.1}$$

- Manhattan distance (Eq. 4.2) – also referred to as Taxicab Geometry, City Block Distance, or L1 Norm, measures the sum of absolute differences of their Cartesian coordinates. It has a high application in regression analysis and frequency distribution scenarios;

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{4.2}$$

- Chebyshev distance (Eq. 4.3) – also known as maximum or L∞ metric, where the distance between two vectors is the greatest of their difference alongside any coordinate dimension;

$$d(x_1,x_2) = max_i(|x_{2,i} - x_{1,i}|) \tag{4.3}$$

- Minkowski distance (Eq. 4.4) – calculates the distance between two vectors within an n-dimensional space. The outcome of this distance computation depends upon the order parameter ($p$). Specifically, for $p = 2$, this metric corresponds to the Euclidean distance, $p = 1$, to the Manhattan distance, and $p = \infty$, to Chebyshev;

$$d(x,y) = \sqrt[p]{\sum_{i=1}^{n}(x_i - y_i)^p} \tag{4.4}$$

- Mahalanobis distance (Eq. 4.5) – calculates the distance between a point x and a distribution y. Considering the covariance structure of this distribution, this distance metric computes the correlation between the variables, allowing the discovery of more similar events/procedures/items. It has many applications, such as multivariate anomaly detection, classification on highly imbalanced datasets, and one-class classification problems. The $W$ parameter

represents the covariance matrix;

$$d(x,y) = \sqrt{(x-y)W^{-1}(x-y)^T} \tag{4.5}$$

- Hamming distance (Eq. 4.6) – computes the number of positions at which corresponding symbols diverge between two strings with the same length. Equation 4.7 increases the distance if elements are distinct;

$$H(x,y) = \sum_{i=1}^{n} d(x_i,y_i) \tag{4.6}$$

$$\delta(x_i,y_i) = \begin{Bmatrix} 0 & if x_i = y_i \\ 1 & if x_i \neq y_i \end{Bmatrix} \tag{4.7}$$

- Canberra distance – measures the dissimilarity between two points in a vector space. Although it presents robustness against outliers, it is sensitive to values near 0;

$$d(x,y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{4.8}$$

- Pearson correlation coefficient (Eq. 4.9) – is a metric that assesses the linear correlation between two variables X and Y. Depending on the correlation level, this measure can present values from -1 to +1. A Pearson correlation coefficient of 1 indicates that the instances analyzed are perfectly correlated, while a score of -1 means that the objects are uncorrelated. Essentially, items with a Pearson correlation coefficient equal or close to 1 are prioritized by an RS instead of items showing lower coefficients;

$$p(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.9}$$

- Spearman Rank Correlation (Eq. 4.10) – or Spearman's $\rho$, is a non-parametric measure that verifies the relationship between two variables using monotonic functions. These functions are known for either never increasing or decreasing as their independent variable increases;

$$r_s = 1 - \frac{6\sum_i d_i^2}{n(n^2-1)} \tag{4.10}$$

- Cosine similarity (Eq. 4.11) – is primarily applied in text analysis to check document similarity. It involves measuring the cosine of the angle between two vectors (A and B) projected in an inner product space. If the two vectors are aligned in the same direction, they are considered correlated; otherwise, they are uncorrelated. Gomaa and Fahmy conducted a survey comprising several text similarity measures (string-based, corpus-based, and knowledge-based) [Gomaa and Fahmy, 2013];

$$cosine(A,B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} \cdot \sqrt{\sum_{i=1}^{n} {B_i}^2}} \tag{4.11}$$

- Kullback-Leibler Divergence (Eq. 4.12) – or KLD score or relative entropy, quantifies the difference between two probability distributions (P and Q). It can be used to measure the difference between predicted and actual user preferences;

$$D_{KL}(P||Q) = \sum_{x \epsilon X} P(x) log\left(\frac{P(x)}{Q(x)}\right) \tag{4.12}$$

- Hellinger Distance (Eq. 4.13) – another metric evaluating the dissimilarity between two probability distributions (P and Q). It ranges from 0 to 1, with 0 indicating identical distributions and 1 completely disjoint distributions;

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i}^{n}(\sqrt{p_i} - \sqrt{q_i})^2} \tag{4.13}$$

- Jaccard coefficient (Eq. 4.14) – quantifies the similarity by calculating the intersection divided by the union of the objects. The resulting values lie within the range of 0 to 1. For example, if users A and B watch the same movies but hold different opinions, they are still considered similar (the Jaccard coefficient is 1);

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{4.14}$$

- Adjusted Jaccard coefficient – also denoted as Tanimoto coefficient extends the previous metric to other data types besides binary.

Besides these similarity measures, numerous other variations have been proposed to compute the similarity between two events, such as the constrained Pearson correlation, the adjusted Cosine similarity, and Pip similarity, among others [Agarwal and Chauhan, 2017]. Schwarz et al. conducted an interesting analysis comparing the performance of the Cosine similarity measure and the Pearson correlation coefficient regarding their effectiveness in predicting user-item ratings within RSs [Schwarz et al., 2017]. Alamuri et al. categorized many existing measures into two main classes: context-free and context-sensitive measures for categorical data [Alamuri et al., 2014]. Regarding continuous data, Shirkhorshidi et al. detailed these metrics' advantages, disadvantages, and applications [Shirkhorshidi et al., 2015].

Table 4.1: Similarity Measures Information

| Metric | Data Type | Limitations |
|---|---|---|
| Minkowski | Numeric | Sensitive to outliers and lack of feature standardization. Hard to compute appropriate $\rho$; |
| Euclidean | Numeric | Sensitive to outliers and scaling. Can't handle sparse data; |
| Manhattan | Numeric | Limited to grid-based data. Sensitive to outliers and scaling; |
| Chebyshev | Numeric | Sensitive to outliers. Inaccurate with sparse data; |
| Mahalanobis | Numeric | Requires a complete covariance matrix. Captures only linear relationships between variables. Sensitive to outliers; |
| Hamming | Binary and categorical | Can only compare vectors with the same length. No knowledge about the difference magnitude; |
| Canberra | Numeric | Sensitive to values around zero and outliers; |

| | | |
|---|---|---|
| Pearson correlation coefficient | Numeric and Interval or Ratio | Sensitive to outliers and range of observations; |
| Spearman rank correlation | Ordinal and ranked | Ignores the magnitude of differences between the values of variables. Converting data to ranks leads to information loss; |
| KLD | Probability distributions | Asymmetric |
| Coasine | Numeric and interval or ratio | Ignore the difference in magnitude between vectors. Assumes vectors are normalized. Sensitive to sparse data; |
| Adjusted cosine | Numeric and interval or ratio | Sensitive to sparse data. Requires more computation than the standard version; |
| Jaccard coefficient | Binary | May be unsuitable for clustering tasks. Sensitive to sparse data and does not consider item frequency; |
| Tanimoto | Binary and non-binary | Same drawbacks as the previous metric. |

### 4.1.4 Explainability

Explainable AI (XAI), also known as interpretable and transparent AI, builds trust among users by providing hints and explanations elucidating the reasons behind the outcomes [Adadi and Berrada, 2018]. Figure 4.13 illustrates the main principles targeted by explainability.

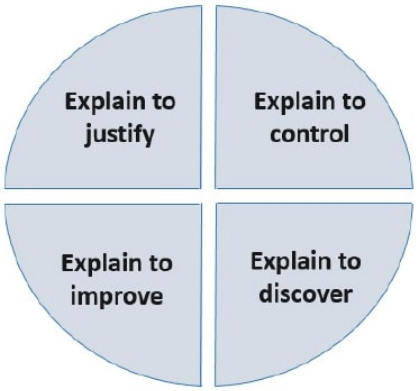



Figure 4.13: XAI Fundamentals [Adadi and Berrada, 2018]

XAI serves as a suitable repository of justifications, offering insights into decision-making processes, particularly in unexpected instances. It provides enhanced control by describing the system's inner workings, potentially mitigating the likelihood of failures and errors. Moreover,

XAI allows the discovery of novel facts by identifying threats and opportunities and analyzing system weaknesses. Despite its potential, XAI is challenged by the trade-off between accuracy and explainability, and the lack of standardized metrics and methods for evaluation [Minh et al., 2022].

Nevertheless, XAI assumes a pivotal role, not only as a catalyst for a more trustworthy AI but also in helping address complex challenges across various domains such as cybersecurity [Capuano et al., 2022], education [Khosravi et al., 2022], and insurance [Owens et al., 2022].

The introduction of explainability features into RSs can enhance their effectiveness by increasing the transparency of decision-making. Zhang and Chen provided an in-depth analysis of explainable recommendations and techniques for achieving such transparency [Zhang and Chen, 2018].

## 4.2 Related Work

This section presents an overview of various types of solutions focused on recommender systems. It also summarizes their applications in cybersecurity, highlights available methods in context-aware and sequence-aware approaches, and debates strategies for addressing the cold-start problem.

In 2011, Shani and Gunawardana offered an in-depth analysis of possible experimental configurations employed in comparing distinct RSs [Shani and Gunawardana, 2011]. The authors scrutinized three types of experiments: offline, online, and user studies. Moreover, they identified several properties beneficial to evaluating RSs, including scalability, privacy, and utility.

A year later, Lü et al. provided a comprehensive overview of the concept, techniques, challenges, applications, and evaluation metrics pertinent to RSs, from the perspective of physician experts instead of computer scientists [Lü et al., 2012]. This study is distinguished by its adoption of a paradigm wherein complex networks and physics principles are harnessed in the recommendation mechanism. Park et al. proposed a classification schema for categorizing the domains and approaches investigated in 210 research articles on RSs [Park et al., 2012]. The domains covered include areas such as books, documents, images, movies, music, shopping, and TV programs. The methodologies are organized into eight groups: association rule, clustering, DT, K-NN, link analysis, NN, regression, and heuristic-based.

In 2013, Bobadilla et al. detailed the evolutionary path followed by RSs, partitioning it into three levels: traditional web, social web or web 2.0, and IoT web-based or web 3.0 [Bobadilla et al., 2013]. The initial level emphasizes the employment of filtering techniques to improve recommendation accuracy; the second stage augments the outcomes by incorporating social data information; and the third phase entails applying IoT data to gather more complex knowledge.

After two years, Lu et al. clustered RSs applications into eight categories and summarized recommendation methods utilized in each group [Lu et al., 2015]. The domains explored include government, business, e-commerce, library, learning, tourism, resource services, and group activities. The reported RS in each field is analyzed in terms of recommendation approach, software, domain, and platform.

In 2016, Aggarwal et al. researched diverse service providers utilizing RSs, such as Amazon, Netflix, Google News, Facebook, and GroupLens [Aggarwal, 2016]. In addition to elucidating the

inner workings of the RS methods available, the authors gave a brief overview of RSs employing supplementary data, including temporal, location-based, and social information. Beel et al. conducted a comprehensive review comprising 200 research articles published between 1998 and 2013 [Beel et al., 2016]. Their findings show that 55% of the papers inspected employ content-based filtering, while 18% opt for CF approaches, and 16% embrace graph-based solutions. Moreover, the authors suggest possible motives for the ambiguity surrounding the most suitable recommendation method. Sharma and Singh introduced a taxonomy mapping various domains with existing RSs and alongside their associated methodologies [Sharma and Singh, 2016]. Their framework also details assessment metrics while addressing the inherent limitations of each RS algorithm.

Two years later, Karimi et al. explored news recommendation, encompassing an analysis of its limitations, existing methodologies, and evaluation concerns [Karimi et al., 2018]. Their findings identify scalability as a paramount challenge within this field while highlighting the prevalence of information retrieval and click-through rates as commonly employed metrics for evaluation.

In 2019, Mohamed et al. investigated the RS-related work and proposed different classification frameworks based on the application domain, approach, simulation platform, and evaluation measures [Mohamed et al., 2019]. Their analysis delineates each method's advantages and disadvantages alongside research projects tackling them.

In 2020, Wu et al. elaborated a systematic taxonomy for GNN RSs according to the data type and recommendation task entailed [Wu et al., 2020]. The proposed classification schema categorizes GNN-based solutions into five main groups: including user-item CF, sequential recommendation, social recommendation, knowledge graph-based recommendation, and other group aggregating tasks that do not fit into the previous classes (POI recommendation, group recommendation, bundle recommendation, multimedia recommendation, and click-through rate prediction).

One year later, Wang et al. researched the landscape of session-based recommender systems, focusing on existing literature, their session data properties, and associated challenges [Wang et al., 2021b]. They categorized existing projects into three sub-areas: next interaction recommendation, next partial-session recommendation, and next session recommendation. Furthermore, the authors formulated the problem statement underpinning these systems, orchestrating the related work based on session topology, model employed, and area. Gao et al. argued that the benefits of applying GNNs to RSs stem from their high-order connectivity, the structural property of data, and the supervision signal [Gao et al., 2023]. Various challenges involving the introduction of GNNs in RSs, including graph construction, network design, model optimization, and computation efficiency, are discussed. Moreover, the authors organize the related work based on four criteria: stage, scenario, objective, and application.

Much of the existing literature has focused on developing and refining RSs across diverse applications. CB methods have received the most attention, followed closely by CF and hybrid.

In 2010, Ghazanfar and Prugel-Bennett designed a scalable hybrid RS integrating CF, CB, and demographic properties to predict item ratings already appraised by the users and to evaluate the effectiveness of the predictions obtained even in cold-start scenarios [Ghazanfar and Prugel-Bennett, 2010]. Their approach calculates item similarity using adjusted Cosine similarity for rating data and

vector similarity for demographic and feature vectors. A boosting function is then applied to predict an item for a particular user. The model was evaluated using MovieLens[10] and FilmTrust[11] datasets, with an 80/20 training-test split, against seven different techniques [Breese et al., 1998, Pazzani, 1999, Sarwar et al., 2001]. The experiments indicate that the proposed solution outperforms the remaining strategies with the lowest MAE value. Wang et al. developed a hybrid user model CF that combines the strengths of both memory and model-based approaches to reduce system complexity, lower computational overhead, and improve scalability and performance [Wang et al., 2010]. Given the challenges of scalability and recommendation accuracy inherent in CF, the authors combine the accuracy of the memory-based method with the scalability of the model-based technique to address these problems efficiently. Figure 4.14 outlines their model's main operations, consisting of two stages: offline model construction and online recommendation.

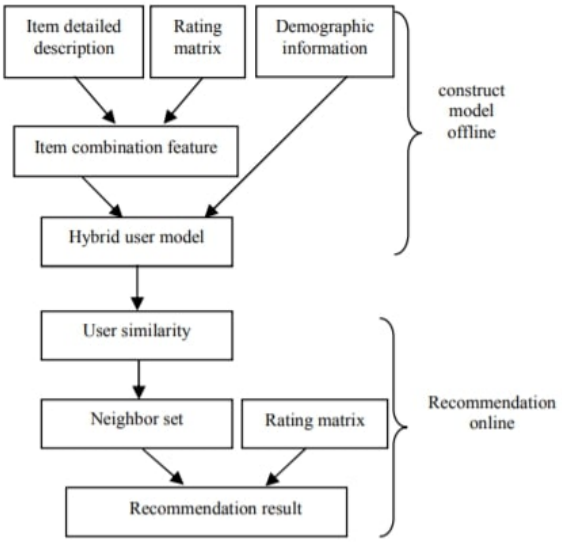



Figure 4.14: Hybrid User Model [Wang et al., 2010]

In the offline phase, a user model is built using item combination features from the rating matrix and demographic information. This model enhances scalability and recommendation quality by grouping users with similar tastes. The online step generates recommendations based on the previously constructed model, employing memory-based and model-based CF techniques. The model-based approach identifies similar users (neighbours), while the memory-based method combines the rating matrix with the neighbor set to produce recommendations. Genetic algorithms are employed in both stages to optimize the learning of feature weights and improve similarity calculations, enhancing overall accuracy [Wang et al., 2010]. The authors tested the model on the MovieLens data dataset with an 80/20 training-test split and compared its performance against traditional CF [Adomavicius and Tuzhilin, 2005] and content-based filtering combined with CF [Kim et al., 2006] using the MAE metric and neighbours between 5 and 50. The proposed

[10]Dataset available at https://grouplens.org/datasets/movielens/
[11]Dataset available at https://www.librec.net/datasets.html

methodology consistently outperforms the remaining alternatives, manifesting lower MAE values across most cases, except in instances where the neighbor's number falls within 15 and 20.

In 2012, Mican et al. introduced an RS for social networks that delivers personalized content by utilizing explicit and implicit user interactions [Mican et al., 2012]. This system is divided into two modules: data collection and recommendation tasks. The first component gathers user interactions with other users and content, such as comments, ratings, and clicks. This data is then used to compute trust scores and determine and prioritize recommendations. In the recommendation stage, personalized suggestions are presented based on these trust scores and favorably rated content. The system was evaluated on a Romanian cultural website[12], where 511 users out of a total population of 6,723 generated 16,620 interactions. After analysis, 1,388 user connections were identified, with approximately 6.23% being explicit and 18.62% implicitly inferred using the proposed method. The authors argued that this method, which combines user behaviour and trust scores, offers a viable alternative to conventional CF and CB methods. Furthermore, incorporating favorably rated texts alongside those published by friends mitigates serendipitous concerns.

Four years later, Dev and Mohan proposed a joint-based RS capable of countering drawbacks, such as scalability, cold-start, and novelty [Dev and Mohan, 2016]. Their system computes the similarity between user preference sets extracted from user-item reviews. The active user's preferences are then compared to those of others with similar tastes, and a score is assigned to each item based on matching user ratings, generating recommendations. The system is implemented using the MapReduce[13] framework and the Hadoop[14] distributed computing platform. Its performance was evaluated against a user-based algorithm using the Pearson correlation coefficient [Adomavicius and Tuzhilin, 2005] and keyword-aware service recommendation method [Meng et al., 2014]. Various metrics, including MAE, normalized MAE (NMAE), mean average precision (MAP), and discounted cumulative gain (DCG), were used to assess the accuracy and the quality of top-k recommendations [Shani and Gunawardana, 2011]. Its scalability was measured with the Speedup metric by adjusting the number of computational nodes [Yang et al., 2013]. The results showed that the presented method outperformed the remaining techniques in terms of accuracy, recommendation quality, and scalability.

### 4.2.1 Cybersecurity

Regarding the literature review of RSs applied in cybersecurity, it is organized based on surveys, cyberattack prediction, visual analysis, vulnerabilities related to cyberattacks and anomalies, and companies' suggestions.

In 2016, Gadepally et al. investigated the utilization of RSs in the Department of Defense (DoD) and the intelligence community (IC) applications [Gadepally et al., 2016]. They highlighted a fundamental distinction between RSs' deployment in these sectors versus commercial solutions.

[12]Website available at http://intelepciune.ro./
[13]Documentation available at https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[14]Documentation available at https://hadoop.apache.org/

While commercial success is quantified by tangible actions, such as product sales, in the DoD, success is measured by whether an action will lead to a greater likelihood of mission accomplishment. The authors also emphasized the necessity to clarify ethical and technical questions about user trust, privacy, and security preservation, user environment adaptation, multi-level metrics, system extension, and collaborations between the academy and industry. At the time of publication, their lab was working on seven projects combining RS with cybersecurity.

Five years later, Pawlicka et al. published an exhaustive analysis of the advantages, disadvantages, applications, and research work available within this domain [Pawlicka et al., 2021]. The authors also addressed pertinent research questions regarding the state-of-the-art of RSs in cybersecurity, discussing various approaches.

In 2010, Soldo et al. devised a multi-level model to predict future blacklists, framing the problem as an implicit RS inspired by the Netflix model [Soldo et al., 2010]. Their approach used ML to forecast malicious activities, combining time series analysis with two methods: victim neighborhood (K-NN) and joint attacker-victim neighborhood. The former captures similarities among victims targeted by the same attackers, while the latter uses co-clustering to group the intruders and victims concurrently involved in the malicious activity. The model was evaluated using one month of real data[15], with the primary metric being the hit count, which measured the accuracy of predicting attackers on the blacklist. The proposed model outperformed state-of-the-art techniques, including the local and global worst-offender lists. Furthermore, tests showed the model's robustness and accuracy in handling false positives (pollution) and malicious contributors (poison events).

Four years later, Lyons formulated a hybrid system capable of anticipating cyberattacks and building a prioritized list of cyber-defense actions [Lyons, 2014]. Her research investigates using an RS with a CF to predict attacks, combined with a KB technique to formulate defensive strategies. This system integrates several components, such as a virtual network, a host server machine, an IDS, and the RS. Performance was measured using the RMSE, factoring in the recommendation speed and other properties. The system's performance was affected by various factors like the configuration of client machines and the RS machine, IDS alerts, and the RSs' initial state before executing the algorithm. Despite lower prediction accuracy, the tool demonstrated proficiency in generating actionable defense actions to mitigate threats.

Three years later, Polatidis et al. examined the predictive capabilities of RSs using a multi-level CF approach [Polatidis et al., 2017]. Their algorithm modeled cyber attack paths through graph-based modeling, combining attack path discovery and attack prediction to anticipate future threats. The attack path discovery stage analyzes attacker features, such as geographical location, capability, entry, and target selection. The attack prediction phase then uses this information, applying RS methods to forecast and classify potential cyberattacks. The system was tested using IT infrastructure data from the maritime supply chain and validated within a cybersecurity maritime supply chain risk management system. Expert opinions from five professionals further supported the conclusion that this approach effectively predicts incoming intrusions.

[15]Information available at `Dshield.org`

In 2010, Rasmussen et al. explored methods to enhance real-time monitoring and incident response for cybersecurity operators handling alerts from IDSs [Rasmussen et al., 2010]. Their solution utilized a graph-based visualization approach to represent correlated IDS outputs and generated defense recommendations based on ML approaches applied to historical analyst behavior. This led to the development of a prototype cybersecurity framework called Network Intrusion Management (NIMBLE), which aimed to enhance decision-making by employing learned expertise. The evaluation involved eighteen cybersecurity experts using alert data from operational monitoring systems and assessed metrics such as accuracy, response time, confidence, and ratings. Although NIMBLE did not fully replicate analysts' performance, it improved accuracy by providing defensible and displayable recommendations.

Three years later, Nunnally et al. implemented NAVSEC, an RS designed to streamline navigation within 3D network security visualization frameworks [Nunnally et al., 2013]. NAVSEC enhances user efficiency in detecting, identifying, and inspecting network attacks by presenting visual and interactive recommendations. The system comprises four main components: active user, expert user community, interaction database, and recommender module. The active user represents someone engaging with the visualization tool, while the specialist user community provides understanding based on their network security expertise. The interaction database records user interactions, which the recommender structure analyzes to suggest optimal actions for the active user. NAVSEC was evaluated using stealthy port scanning attacks, comparing the volume of interactions with and without the system enabled. The similarity between active users' interactions and expert analysts' behaviours was also assessed. Results showed that NAVSEC effectively assisted users in navigating the platform and identifying potential threats.

In 2021, Brisse et al. developed KRAKEN, a KB-based RS designed to guide human operators through exploratory paths within log data during incident analysis [Brisse et al., 2021]. The system aims to reduce the time analysts spend on labor-intensive data collection tasks by integrating knowledge from APT repositories, like MITRE ATT&CK[16] into a visual framework called ZeroKit. KRAKEN was evaluated using a subset of the transparent computing exercise 3 dataset[17] from the Defense Advanced Research Projects Agency (DARPA). Seven cybersecurity experts used KRAKEN to analyze various APTs from this dataset, receiving feedback that confirmed its useful suggestions and enhanced the efficiency of incident analysis.

In 2015, Zhou et al. investigated the recommendation of procedures for monitoring tickets while reducing their repetition [Zhou et al., 2015]. They applied a KNN approach for ticket similarity analysis, enhanced by incorporating event and procedure data through topic-level feature extraction using the LDA model. The authors also introduced a metric learning, which further refines the similarity measurement in scenarios where treatment procedure categories are known. The evaluation experiments were employed on real ticket datasets, utilizing four KNN variants and measuring performance with the average similarity and MAP metrics. The results showed that

[16]Documentation available at https://attack.mitre.org/

[17]Dataset available at https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md

including metric learning and resolution information improves efficiency in service management.

In 2016, Campiolo et al. proposed a model that recommends refined cybersecurity alerts for network administrators [Campiolo et al., 2016]. Such a hybrid approach combines CB and CF to extract and prioritize cybersecurity alerts from unstructured external data, improving data interpretability and focusing on critical alerts. The model was refined based on administrator feedback, leading to a detailed data model specification. The proposed model was tested in an offline experiment using the MovieLens dataset and a movie recommendation website, wherein precision and recall metrics were applied. The results show that the proposed model could augment administrator access to crucial information.

Three years later, Nembhard et al. introduced VulIntel (Vulnerability IntelliSensor), a text mining-based RS designed to detect insecure coding segments and suggest remediation actions [Nembhard et al., 2019]. VulIntel consists of two modules: a data analyzer and the RS itself. The former collects intel from sources, such as the National Vulnerability Database (NVD), Common Vulnerabilities and Exposures (CVE), and open-source projects, employing MapReduce on Apache Hadoop to extract detection features. The RS then assesses code safety using these features and IntelliSense technology to propose corrective measures for vulnerabilities similar to those detected. The system's usability was evaluated through A/B tests with fourteen participants, showing its potential to enhance secure coding practices. Additionally, VulIntel's scalability was tested by processing ten random Google code projects for SQL injection vulnerabilities. Results confirmed that VulIntel scales well, demonstrating its robustness in handling larger datasets. Sula developed ProtecDDoS, an RS aimed at mitigating DDoS attacks through offsite network protection services [Sula, 2019]. The goal was to improve decision-making in companies and provide information regarding DDoS protection options. ProtecDDoS adopts a hybrid scheme that includes parameters such as attack type protection, service type classification (reactive or proactive), and coverage protection region. The author used various metrics to evaluate the system's performance and similarity, including Cosine similarity, Euclidean distance, Manhattan distance, Minkowski distance, and Pearson correlation. Additionally, a blockchain-based marketplace was integrated to augment ProtecDDoS, providing greater transparency, security, and traceability. The results showed that ProtectDDoS offers accurate recommendations while maintaining low blockchain costs.

Two years later, McDonnell et al. implemented a cyber bidirectional encoder representation from transformers (CyberBERT), a deep session-based RS designed to discern the intentions of anonymous users through dynamic state models [McDonnell et al., 2021]. CyberBERT utilizes bidirectional transformers to enhance user representation and excels in tasks such as malware recognition and next-click prediction. Its efficiency was tested within two scenarios. In the first case, the Windows PE Malware Application Programming Interface (API) dataset was employed to analyze CyberBERT's performance in malware classification against benchmark models, such as LSTM and Transformer [Vaswani et al., 2023]. In the second scenario, CyberBERT's next-click prediction capabilities were assessed using the YOOCHOOSE dataset, comparing it with item K-NN and alternative techniques. Their findings demonstrate CyberBERT's superiority over other state-of-the-art approaches in both tested scenarios. Ayala et al. developed a hybrid RS to support

cybersecurity experts in managing anomalies and vulnerabilities while minimizing the response time and workload [Ayala et al., 2021]. This tool integrates CF and KB methodologies to prioritize threats based on their criticality. With the collaboration of domain experts and security entities, such as Symantec, Open Web Application Security Project, and the National Institute of Standards and Technology (NIST), the CF model identifies the most severe vulnerabilities and anomalies, while the KB element alleviates data scarcity. The system was evaluated using the technology acceptance model and feedback from six academic and industry experts. Empirical findings demonstrated the effectiveness of the RS in reducing the time invested by professionals during incident treatment.

In 2016, Abuhussein et al. examined the integration of RSs within the cloud computing domain to address security and privacy risks [Abuhussein et al., 2016]. They proposed a cloud services security recommender (CSSR), a conceptual framework designed to help businesses understand and manage these risks. CSSR identifies pivotal attributes from the stakeholder perspective to mitigate vulnerabilities within the cloud infrastructure and delivers recommendations for optimal security solutions based on parameters such as attack vector, operational impact, defense, information impact, and target taxonomy. The evaluation of the CSSR performance utilized real-world datasets, including code repositories such as Code Spaces and Git, and was validated with ten specialized students. While CSSR was praised for its user-friendly interface, feedback indicated that the tool's complexity regarding accountability was a concern.

In 2019, Franco et al. introduced MENTOR, a tool to recommend optimal cybersecurity protection services based on user-specific needs, such as geographical location, deployment timeline, and cost constraints [Franco et al., 2019]. MENTOR comprises four main structures: an extractor, a classifier, a retriever, and a recommendation engine. The extractor acquires data about the targeted infrastructure and the characteristics of cyberattacks. The extractor and classifier analyze and correlate the acquired information with the cyberattack type, aiming to identify practical solutions. The retriever then compiles a list of possible protection services, while the recommendation engine selects the most suitable protection service considering the customer profile and analyzed parameters. MENTOR's framework was evaluated using four similarity measures, including Euclidean distance, Manhattan distance, Cosine similarity, and Pearson correlation, on a dataset composed of 10,000 randomly generated protection services. Results showed MENTOR's capacity to recommend protection services while prioritizing cost and geographical location, choosing budget-friendly solutions instead of high-performance ones.

Two years later, Huff et al. implemented an RS aimed at aligning an organization's hardware and software with the typical software product enumerator (CPE) standard used by vulnerability sources like the national vulnerability database (NVD) [Huff et al., 2021]. The RS utilizes NLP, fuzzy matching (FM), and ML algorithms to simplify the identification of vulnerabilities associated with software products, reducing the workload of human operators. In the first stage, NLP transforms software names into word vectors. Next, the FM measures the Cosine similarity between CPE and these word vectors. Lastly, ML refines and ranks the FM results, presenting analysts with prioritized CPEs based on severity (highest, high, medium, low, lowest, and reject) and key attributes like vendor and product attributes. Based on these outcomes, the RS provides prioritized

CPE suggestions for focused attention by analysts. The performance was evaluated by comparing scenarios with and without the RS, using 50 software and hardware inventory packages. Without the RS, the average number of manual searches for software was 119, with only 8% leading to conclusive results. With the RS, the average number dropped to 2, with 40% conclusive results. For hardware, searches decreased from 34 to 2, with an enhanced conclusiveness rate from 28% to 48%. This analysis proves the RS's ability to provide precise results in identifying vulnerabilities compared to manual processes.

Table 4.2 presents various research projects investigating the application of RSs in the cybersecurity domain. The papers were categorized based on three points: surveys provide an overview of RS applications in cybersecurity, studies focused on approaches that anticipate attacks, identify threats, and provide defense measures, and approaches that support data interpretation and decision-making processes, goals that are shared by CROSS.

Table 4.2: Cybersecurity RSs related work summary

| Ref | Survey | Prediction | Cyberattacks/ Anomalies Vulnerabilities | Visual Analytics | Companies Suggestions |
|---|---|---|---|---|---|
| [Rasmussen et al., 2010] | - | - | ✓ | ✓ | - |
| [Soldo et al., 2010] | - | ✓ | - | - | - |
| [Nunnally et al., 2013] | - | - | ✓ | ✓ | - |
| [Lyons, 2014] | - | ✓ | ✓ | - | - |
| [Zhou et al., 2015] | - | - | ✓ | - | - |
| [Gadepally et al., 2016] | ✓ | - | - | - | - |
| [Campiolo et al., 2016] | - | - | ✓ | - | - |
| [Abuhussein et al., 2016] | - | - | - | - | ✓ |
| [Polatidis et al., 2017] | - | ✓ | - | - | - |
| [Nembhard et al., 2019] | - | - | ✓ | - | - |
| [Franco et al., 2019] | - | - | - | - | ✓ |
| [Sula, 2019] | - | - | ✓ | - | - |
| [Ayala et al., 2021] | - | - | ✓ | - | - |
| [Huff et al., 2021] | - | - | - | - | ✓ |
| [Brisse et al., 2021] | - | - | ✓ | ✓ | - |
| [Pawlicka et al., 2021] | ✓ | - | - | - | - |
| [McDonnell et al., 2021] | - | - | ✓ | - | - |

### 4.2.2 Context-aware

The related work in this area is organized based on distinct criteria, including surveys, privacy, social networks, sequence context, evaluation, scalability, sentiment analysis, and latent context.

In 2011, Adomavicius et al. investigated the concept of context and its role in improving RS's performance [Adomavicius et al., 2011]. Their analysis is focused on strategies for extracting contextual data, including explicit, implicit, and inference techniques, incorporating it into RSs through

pre-filtering, post-filtering, and contextual modeling algorithms. Furthermore, they elucidate a unified approach derived from these methods, exemplified through a detailed case study.

A year later, Verbert et al. proposed a taxonomy with different context dimensions within technology-enhanced learning (TEL) applications [Verbert et al., 2012]. The taxonomy encompasses computing, location, time, physical conditions, resources, user, and social relations. Their work also reviews TEL RSs, showing challenges and prospects in this field.

In 2015, Champiri et al. provided a comprehensive review of CARS solutions in academic digital libraries [Champiri et al., 2015]. Their analysis revealed a categorization of existing approaches based on the data types utilized, which predominantly fell into user, document, and environmental contexts. They found CF to be the most prevailing approach among various techniques. The authors also offer suggestions for developing effective CARSs within this domain. Abbas et al. surveyed CARSs utilizing computational intelligence (CI) approaches, assessing their efficiency in handling challenges, such as sparsity, cold-start, and scalability [Abbas et al., 2015]. The CI methodologies studied comprise fuzzy sets, artificial neural networks, evolutionary computing, swarm intelligence, and artificial immune systems. The evaluation was performed on these methods using MAE, RMSE, NMAE, coverage, precision, recall, and other metrics. The experiments demonstrate that the most effective CARS is based on combining multiple CI approaches.

Two years later, Ben Sassi et al. published an in-depth analysis of existing CARSs in mobile environments [Ben Sassi et al., 2017]. The related work is organized based on the nature of contextual data, including location, time, social information, activities, emotions, and multi-dimensional data. Each CARS implementation within these groups describes the application field, context granularity, recommendation approach, handling of cold-start cases, and evaluation employed.

In 2018, Villegas et al. surveyed CARSs to elucidate the relationship between contextual information and recommendation techniques [Villegas et al., 2018]. By examining numerous research papers, the authors discussed the techniques utilized across various stages of the recommendation task, the methods devised for integrating contextual data, and the phases at which such data is assimilated. Additionally, this review offers insights into the most prevalent context types in different domains alongside evaluation strategies, including properties, metrics, and protocols.

After one year, Raza and Ding addressed diverse recommendation techniques utilized by CARSs while also discussing strategies for mitigating the curse of dimensionality [Raza and Ding, 2019]. The most predominant recommendation approaches the research community embraces for integrating contextual factors include MF, Tensor Factorization (TF), LDA, Markov, and bandit algorithms. Furthermore, to alleviate the curse of dimensionality, the authors deliberated on techniques such as PCA, SVD, and MF alongside other methodologies.

In 2020, Kulkarni and Rodd provided an in-depth exploration of the context concept, including known classifications and techniques adept at incorporating this aspect in RSs [Kulkarni and Rodd, 2020]. Existing methodologies are categorized into two groups: bio-inspired computing and statistical computing. The former pertains to CI methods inspired by biological behaviours

observed in nature, while the latter entails the application of statistical tools for learning from data. The authors also analyze ideas to tackle challenges such as cold-start, sparsity, and scalability.

One year later, Lozano Murciego et al. conducted a literature review on deploying CARSs within the music domain [Lozano Murciego et al., 2021]. Their work encompasses analyzing diverse contextual types, evaluation protocols, and metrics alongside methodologies for integrating such data into the recommendation process. Moreover, they investigate various devices and software, including smartphones, sensors, and IoT devices, adept at capturing contextual data for music recommendation. Suhaim and Berri presented a comprehensive analysis exploring context factors in online social networks [Suhaim and Berri, 2021]. Their literature review discusses recommendation techniques employed in CARSs, such as CF, content-based, graph-based, and hybrid. Furthermore, the authors discern potential research avenues, including implicit context data and the introduction of IoT wearables to enrich contextual information acquisition. Rodríguez-Hernández and Ilarri also contributed with a vast survey focusing on AI-driven mobile CARSs from an information management standpoint [del Carmen Rodríguez-Hernández and Ilarri, 2021]. Their work encompasses knowledge about mobile computing, CARS features and techniques, and other relevant characteristics. The authors divide mobile context-aware recommendation methods into pull-based and push-based. The former entails users explicitly requesting suggestions, whereas the latter delivers implicit recommendations without a direct request. Their findings suggest a trend towards increased adoption of push-based solutions in the future, enabling mobile devices to operate in a more distributed manner. Nawara and Kashef investigated content-aware RSs deployed within IoT environments as their associated application domains [Nawara and Kashef, 2021]. Apart from exploring classical context-based techniques and existing limitations, such as cold-start, scalability, and sparsity, the authors address the context life cycle, comprising phases like contextual information acquisition, context modeling, context reasoning, and context dissemination. The most common challenges in this area include heterogeneity, scalability, interoperability, limitations related to context exploration and extraction, security concerns, and computation capabilities.

In 2022, Adomavicius et al. offer an updated view of approaches capable of modeling contextual information within RSs [Adomavicius et al., 2022]. In addition to classical methods and TF, the authors scrutinize the application of DL and reinforcement learning (RL) techniques for delivering contextualized suggestions.

Two years ago, Casillo et al. studied CARSs in the cultural heritage field [Casillo et al., 2023]. The authors debated various approaches and systems that facilitate the integration of context information and enhance the cultural heritage experience with improved recommendations.

In 2014, Panniello et al. conducted a benchmarking analysis with various pre-filtering, post-filtering, and contextual modeling techniques, employing performance and diversity metrics [Panniello et al., 2014]. Their experimental framework evaluates CARS methods across distinct recommendation task types (Find-all vs. Top-n), context granularities (coarse vs. fine granularity), and recommendation data types. Evaluation metrics include precision, recall, F-measure, Shannon's entropy, Tidemann & Hall's index, among others. The results indicate the absence of a singular dominant methodology across the scenarios and settings explored. The authors claim that CM is

the most reliable approach for achieving balanced performance in terms of accuracy and diversity.

In 2022, Chen et al. introduced SecRec, a privacy-preserving algorithm designed to ensure data privacy while maintaining the accuracy of context-aware recommendations [Chen et al., 2022]. This methodology supports offline users and operates with various security protocols. These security modules use standard additive secret-sharing structures with secure negative integer computation. The authors evaluated the SecRec protocols, focusing on runtime performance and computation overhead, measured in terms of message bytes. Additionally, they conducted an empirical analysis using a real-world dataset, comparing SecRec's performance against traditional methods that do not prioritize privacy. Experimental findings show that SecRec achieves recommendation accuracy comparable to non-privacy-preserving strategies. Moreover, its suitability for cloud environments is highlighted due to its parallel computing capabilities.

In 2013, Liu and Aberer devised SoCo, an RS combining social network data and contextual information to augment recommendation efficiency [Liu and Aberer, 2013]. This system employs random DTs to partition the rating matrix into distinct sub-matrices based on their contexts. Then, SoCo predicts user preferences for items by referencing similar rating sub-matrices. This prediction task is facilitated by a modified MT approach, enhanced with a social regularization term aimed at capturing the influence of friends sharing similar interests. The user similarity is quantified using a context-aware Pearson correlation coefficient. SoCo's performance is evaluated against state-of-the-art context-aware and social RSs [Sarwar et al., 2001, Ma et al., 2011, Zhong et al., 2012] with MAE and RMSE metrics. The results demonstrate SoCos's superiority, exhibiting a performance enhancement of 15.7% and 12.2% over the respective comparison groups.

Seven years later, Zhao et al. devised TrustTF, a novel bias tensor factorization model using user social trust data and implicit feedback to mitigate sparsity and enhance the recommendation performance [Zhao et al., 2020]. The social data is categorized into unilateral and mutual trust, enabling the association of distinct weight functions and varying degrees of relationship strength. TrustTF is evaluated against several baseline methods [Ma et al., 2008, Ma et al., 2009, Ma et al., 2011, Guo et al., 2015, Yang et al., 2017, Wu et al., 2017] through MAE and RMSE metrics. The results highlight the benefits of integrating implicit feedback and social data in the recommendation task, with TrustTF outperforming the other solutions.

Following another year, El Yebdri et al. implemented TCPoFA, a hybrid trust-based context-aware post-filtering methodology combining trust and context to enhance recommendation quality [El Yebdri et al., 2021]. TCPoFA consists of three main steps: preprocess contextual data through context compensation, compensate rating prediction with trust networks for user similarity estimation, and post-filtering recommendation. The proposed method's performance is compared against various state-of-the-art algorithms [Koren, 2010, Said et al., 2011, Zheng et al., 2012, Guo et al., 2015] using evaluation metrics like MAE and RMSE. Experimental findings indicate that TCPoFA outperforms these techniques, except TrustSVD. The authors also explored the impact of trust propagation, trust nearest neighbor, the threshold for confident users, and trust information.

In 2019, Iqbal et al. implemented KCR, a kernel-based CARS characterized by its accuracy, scalability, and flexibility [Iqbal et al., 2019]. This solution has two versions: user-based and

item-based, each exploring different user-related and item-related contexts using an array of kernels. KCR integrates various kernels tailored to capture certain aspects of user and item contexts (time, location, health, and other types). These kernels are combined with additive and multiplicative models, facilitating the computation of residual ratings crucial for the recommendation operation. The KCR is evaluated on different dimensions. Firstly, user- and item-based KCR are assessed with additive and multiplicative models regarding RMSE across sparse and less sparse datasets. Moreover, KCR is compared with other methodologies [Baltrunas et al., 2011, Ghazanfar et al., 2012, Zheng et al., 2013, Wu et al., 2017]. Polynomial and poly-gaussian approaches in rating and context kernels were also experimented with RMSE and F1-measure. The results show that KCR surpasses the remaining techniques in sparse and dense datasets.

During the same year, Aghdam explored a context-aware RS utilizing hierarchical hidden Markov models [Aghdam, 2019]. These models analyze the latent context of users to adapt to changes in their preferences dynamically. Users are represented as a hidden Markov process, with their current context being the hidden variable. This approach aims to maximize the likelihood of transitions between model states and derive a probability distribution for predicting the user's following context, thereby enriching the recommendation task. The effectiveness of this CARS system is evaluated against various baseline models, including a hidden Markov model, SPM, Item-based Markov modeling, user-based kNN, BPRMF, popular, and random, using precision, recall, and F-measure metrics. The overall results indicate that hierarchical hidden Markov models are more efficient than the remaining techniques, and the recommendations can be more diverse. Livne et al. proposed a novel method for modeling sequential context and introduced three neural CF (NCF) extensions tailored for the CARSs field [Livne et al., 2019]. The study capitalizes on the LSTM network to retrieve sequential latent context from sequences of contextual information. Moreover, three NCF variants are presented: the explicit neural context model (ENCM), which integrates explicit context data; the latent neural context model (LNCM), focused on the latent representation of short-term context data; and the sequential latent context model (SLCM), which uses sequential latent context derived by the LSTM. These methodologies are evaluated against various approaches [Baltrunas et al., 2011, Hazan et al., 2011, Unger et al., 2016, He et al., 2017, Civin, 2018] using RMSE, MAE, and HR@k. Experimental results demonstrate that SLCM outperforms these solutions, improving ranking quality by up to 16.3% in the HR@k metric and enhancing prediction accuracy by 9.47% for RMSE and 8.98% for MAE.

In 2020, Unger et al. investigated the incorporation of contextual information in DL frameworks and designed three DL-based CARSs that employ distinct latent context representations [Unger et al., 2020]. These strategies are the explicit context-aware model (ECAM), unstructured context-aware model (UCAM), and hierarchical context-aware model (HCAM), each characterized by the usage of explicit, unstructured, and structured latent contexts, respectively. To assess their efficiency, their performance is evaluated against diverse neural and non-neural CF approaches [Koren, 2008, Rendle et al., 2011, Baltrunas et al., 2011, Unger et al., 2016, Unger et al., 2016, He et al., 2017] through MAE and RMSE metrics. The experiments show that ECAM, UCAM, and HCAM outperform the alternatives for rating prediction, top-k recommendation generation, and

user feedback classification. Moreover, the application of structured latent context yields superior performance compared to other CARS models.

Last year, Sohafi-Bonab et al. presented DCARS, a deep CARS tool that enriches the recommendation task based on session latent context and users' short and long-term preferences [Sohafi-Bonab et al., 2023]. DCARS comprises three phases: user and item embeddings, session extraction context, short and long-term preferences modeling, and recommendation generation. The first level, DCARS, employs a non-negative matrix factorization (NMF) to represent item associations within each session in the latent context space. An attention mechanism is applied to identify the most relevant items within these sessions. In the second phase, a bi-directional LSTM and an attention module encode long-term preferences and discover the most crucial sessions. Meanwhile, gated recurrent units (GRUs) capture short-term interests. The last step integrates short- and long-term preferences with the context of the last session to generate recommendations. DCARS's performance is compared against various methodologies [Rendle et al., 2009, Rendle et al., 2010, Hidasi et al., 2015, Li et al., 2017b, Ying et al., 2018, Liu et al., 2018, Kang and McAuley, 2018] using the hit rate (HR) and MRR metrics. The results obtained showcase DCARS's superiority across the utilized datasets and metrics.

In 2019, Costa and Dolog proposed CoNCARS, a neural CARS system capable of learning temporal patterns in both user's and item's embeddings [Costa and Dolog, 2019]. This platform aggregates six layers: input, collective embedding, interaction, convolutional, fusion, and prediction. The first layer preprocesses user and item information, while the second structure embeds data regarding their interactions (user-item, item-user, user-time, and item-time vectors). The interaction layer models correlations between users and items in different timestamps. The convolutional layer utilizes CNNs to extract nonlinear interactions in the previous phase. The fusion layer merges the output provided by these CNNs, which the prediction layer utilizes to compute the user's preference score for a specific item at a given time. CoNCARS is benchmarked against other baseline methods [Rendle et al., 2009, Karatzoglou et al., 2010, Baltrunas et al., 2011, Kim et al., 2016, He et al., 2017, Costa and Dolog, 2018], using the HR and NDCG metrics. Experimental findings indicate that CoNCARS outperforms the mentioned approaches regarding the top-N item recommendation, considering implicit feedback.

Four years ago, Abinaya and Devi introduced a context-specific sentiment-based stacked autoencoder designed to learn user preferences by combining ratings and reviews, enhancing the top-N recommendation accuracy [Abinaya and Devi, 2021]. This tool comprises four modules: context generation, which is responsible for capturing user behaviour; item-splitting, which builds artificial context-based items; sentiment analysis, which discerns the sentiment scores of the reviews; and user preference prediction and top-N recommendation. This system is compared against some baseline models [Sedhain et al., 2015, Wu et al., 2016, Wang et al., 2019a] with precision, recall, MAP, and DCGN metrics. The experiments showcase CSSAE's superiority over the strategies mentioned and its efficiency in addressing the scalability challenge.

In 2022, Livne et al. presented a genetic-based feature selection method that identifies low-dimensional subsets of contextual data integrated within CARS [Livne et al., 2022]. Their GA

approach generates and stacks various feature subsets, each representing a deep context-aware model. Furthermore, this method can be fine-tuned to prioritize recommendation efficiency and user concerns like privacy and battery consumption. The proposed methodology is evaluated against a range of conventional ML techniques (LR, NB, SVM, RF, XGBoost, and Categorical Boosting), models devoid of contextual features (NeuMF), deep context-aware models with feature selection (ENCM, forward selection context model, and backward selection context model), deep context-aware models with dimensionality reduction (LNCM and SLCM), and deep context-aware models without dimensionality reduction or selection (NCM, deep factorization machines, and combination of feature importance and bilinear feature interaction) with a smartphone dataset. The evaluation metrics encompass AUC, log loss, MRR, HR, and computation time. The experiments show the superiority of the proposed method over the diverse scenarios explored.

Recently, Wei and Chow presented a unified graph recommendation platform integrating graph convolution networks (GCN) with RS techniques, highlighting their common points and differences in parameter configuration and message-passing styles [Wei and Chow, 2023]. Moreover, the authors introduced FGCR, a fused graph context-aware RS that fuses item-based models with a linear graph convolution network to improve the relationship between user, item, and context. FGCR employs a masked graph convolution approach to aggregate user-item interactions and contextual information effectively. The proposed method is evaluated regarding rating prediction [Guo et al., 2017, Wang et al., 2017, Wang et al., 2021a] and top-K recommendation performance [Rendle et al., 2009, He et al., 2020, Wang et al., 2020, Mao et al., 2021, Shen et al., 2021, Liu et al., 2022, Wu et al., 2022] with metrics such as AUC, recall, and NDCG. Experimental findings demonstrate the performance improvement achieved by the proposed solution compared to other methodologies. Additionally, applying an ablation study and user group analysis indicates that FGCR is effective for active users and modeling sparse context relationships.

In 2016, Srivastava et al. developed CSRS, an RS that integrates contextual and sequence-aware features into the recommendation process [Srivastava et al., 2016]. In addition to user data and preferences, CSRS incorporates the sequential bigram of items to enhance recommendation accuracy. Considering the user's context, past preferences, and the sequence of previously favored items, the proposed model computes the score for every next item. These items are ranked according to these values, optimizing recommendation relevance. This method is benchmarked against LDA and a query-based CARS [Hariri et al., 2013] using the HR metric across various datasets. Results show that CSRS outperforms these approaches in various scenarios.

Five years later, Kumar et al. implemented an RS implementing sequence and context-aware features for activity recommendation [Kumar et al., 2021]. With data about user activity, this platform offers suggestions for the next activity (ActivRec method [Kumar et al., 2014]), the following sequence of activities (with ActivRecSeq), and the context for the next activity. User activities are conceptualized as different timelines and inspected with a two-level edit distance, considering the order and features of activities executed. Evaluation of each recommendation level involves the application of diverse strategies [Pitkow and Pirolli, 1999, Rendle et al., 2010, Kumar et al., 2014, Hidasi et al., 2015, Grbovic et al., 2015, Quadrana et al., 2017] using MRR and recall across

various configurations. Regarding the recommendation of the next sequence of activities, the proposed approach is compared against AKOMSeq, PopSeqRec, OccurSeqRec, and DurationSeqRec using metrics such as agreement (a measure of the concordance between the suggested sequence and the actual sequence up to a specified length)) N-count matching. Furthermore, the performance is compared with recommendation-based post-filtering and context-aware post-filtering techniques using the MRR metric to recommend the next context. The experiments proved the introduced methodology's superiority over existing state-of-the-art strategies.

Table 4.3 provides a brief overview of several research projects explored previously. Their division also hinges on three aspects: survey papers analyzing the state of context-aware RSs, studies addressing common challenges such as privacy, scalability, and evaluation, and approaches that enhance recommendations through social networks, sequence context, sentiment analysis, and latent context factors.

Table 4.3: Context-aware related work summary

| Ref | Survey | Privacy | Scalability | Evaluation | Social Networks | Sequence context | Sentiment Analysis | Latent Context |
|---|---|---|---|---|---|---|---|---|
| [Adomavicius et al., 2011] | ✓ | - | - | - | - | - | - | - |
| [Verbert et al., 2012] | ✓ | - | - | ✓ | - | - | - | - |
| [Liu and Aberer, 2013] | - | - | - | ✓ | ✓ | - | - | - |
| [Panniello et al., 2014] | ✓ | - | - | - | - | - | - | - |
| [Champiri et al., 2015] | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | - |
| [Abbas et al., 2015] | ✓ | - | ✓ | - | - | - | - | - |
| [Srivastava et al., 2016] | - | - | - | - | - | ✓ | - | ✓ |
| [Ben Sassi et al., 2017] | ✓ | ✓ | - | ✓ | ✓ | - | - | - |
| [Villegas et al., 2018] | ✓ | - | - | ✓ | - | - | - | - |
| [Costa and Dolog, 2019] | - | - | - | - | - | - | - | ✓ |
| [Raza and Ding, 2019] | ✓ | - | - | ✓ | - | ✓ | - | ✓ |
| [Iqbal et al., 2019] | - | - | ✓ | - | ✓ | - | - | - |
| [Aghdam, 2019] | - | - | - | ✓ | - | ✓ | ✓ | ✓ |
| [Livne et al., 2019] | - | - | - | ✓ | - | ✓ | ✓ | ✓ |
| [Kulkarni and Rodd, 2020] | ✓ | - | ✓ | ✓ | - | - | - | - |
| [Unger et al., 2020] | - | - | - | - | - | - | - | ✓ |
| [Zhao et al., 2020] | - | - | - | - | ✓ | - | - | - |
| [Lozano Murciego et al., 2021] | ✓ | ✓ | - | ✓ | ✓ | - | - | - |
| [Suhaim and Berri, 2021] | ✓ | - | - | ✓ | ✓ | - | ✓ | ✓ |
| [El Yebdri et al., 2021] | - | - | - | ✓ | ✓ | - | - | - |
| [del Carmen Rodríguez-Hernández and Ilarri, 2021] | ✓ | ✓ | - | ✓ | ✓ | - | - | - |
| [Nawara and Kashef, 2021] | ✓ | - | - | ✓ | - | - | - | - |
| [Abinaya and Devi, 2021] | - | - | ✓ | - | - | - | ✓ | - |
| [Kumar et al., 2021] | - | - | - | ✓ | - | ✓ | - | - |
| [Livne et al., 2022] | - | ✓ | - | - | - | ✓ | - | ✓ |
| [Adomavicius et al., 2022] | ✓ | - | - | - | - | - | - | - |

Table 4.3: Context-aware related work summary

| Ref | Survey | Privacy | Scalability | Evaluation | Social Networks | Sequence context | Sentiment Analysis | Latent Context |
|---|---|---|---|---|---|---|---|---|
| [Chen et al., 2022] | ✓ | ✓ | - | - | - | - | - | - |
| [Sohafi-Bonab et al., 2023] | - | - | - | - | - | ✓ | - | ✓ |
| [Wei and Chow, 2023] | - | - | - | - | - | - | - | ✓ |
| [Casillo et al., 2023] | ✓ | - | ✓ | - | - | - | - | - |

Despite considerable research efforts directed toward surveys, evaluation techniques, and their metrics, critical helpdesk concepts such as privacy and scalability continue to present challenges.

### 4.2.3 Sequence-aware

The literature review under examination is structured according to its thematic focus, encompassing surveys, explainability, graph-POI recommendation, SPM, GNN, NN, or attention mechanisms.

In 2018, Quadrana et al. performed an extensive study into the role of sequences in RSs [Quadrana et al., 2018]. They categorized existing literature based on recommendation tasks, goals, and approaches while benchmarking different recommendation techniques. Their analysis, spanning numerous research papers, categorizes works by context adaptation type (session-aware, session-based, and last-n), order constraint (weak, inferred, and strong), and domain. Their findings indicate a prevalence of last-n interactions, especially within the e-commerce domain, often relying on implicit derived ordering constraints.

Two years later, Karlapalepu reviewed RSs integrating CF with SPM algorithms within the e-commerce domain [Karlapalepu, 2020]. The author introduces a taxonomy organizing the related work on SP-based e-commerce RSs according to their features and algorithms. His analysis reveals that incorporating SP into user-item matrices in RSs yields multiple benefits, including improved recommendation accuracy, reduced data sparsity, facilitation of novelty suggestions, and enhanced scalability. Fang et al. investigated the application of DL for sequential recommendation and introduced a taxonomy based on the recommendation task [Fang et al., 2020]. This classification organizes DL-based sequential recommendation systems into three groups, including experience-based, transaction-based, and interaction-based. Experience-based systems are characterized by users interacting with the same object multiple times, exhibiting different behaviour types. In contrast, transaction-based systems involve users displaying the same behaviour type for various items. Interaction-based systems represent a mixture of both previous tasks. Furthermore, the authors elucidate several techniques within these groups and identify aspects that impact DL-based models regarding input, data processing, model structure, and model training.

In 2018, Kang and McAuley implemented SASRec, a self-attention-based sequential recommendation model that captures long-term semantics similar to an RNN and predicts the next item based on the last-$n$ actions like Markov Chains [Kang and McAuley, 2018]. SASRec consists of

four components: an embedding layer, a self-attention layer, a point-wise feed-forward network, and a prediction layer. The embedding layer maps the training sequence into an item embedding matrix, while the self-attention layer generates matrices for the queries, keys, and values via linear projection. The point-wise feed-forward network adds non-linearity and interdimensional interactions to the model, and the prediction layer forecasts the next item. SASRec's performance is benchmarked against other approaches [Rendle et al., 2009, Rendle et al., 2010, Hidasi et al., 2015, He et al., 2017, Hidasi and Karatzoglou, 2017, Tang and Wang, 2018]. Evaluation based on the HR and with normalized DCG metrics demonstrates SASRec's superiority over these techniques across sparse and dense datasets.

One year later, Chu et al. presented a self-attention sequential-aware and knowledge-aware modeling designed for the next news recommendation [Chu et al., 2019]. This hybrid system utilizes self-attention mechanisms to capture intricate data patterns and integrates knowledge graphs to unveil deep user connections. Its performance is evaluated with other methodologies [Hidasi et al., 2015, Li et al., 2017b, Park et al., 2017, Tang and Wang, 2018, Zhang et al., 2018], using metrics like HR@10 and mean reciprocal rank (MRR). Experimental findings highlight the proposed solution's superior performance over alternative strategies. Chen et al. developed a long-term and short-term attention memory network that incorporates the user's interests and sequential session behaviours to predict the next item [Chen et al., 2019]. This system comprises three main components: an embedding layer for mapping users and items into dense spaces, a session-level attention memory network for computing current session embeddings, and a preference-level feed-forward neural memory network for calculating the attention on long and short-term preferences. Evaluation against various algorithms [Rendle et al., 2009, Rendle et al., 2010, Wang et al., 2015b, Yu et al., 2016, Ying et al., 2018], using metrics such as Rec@10, Rec@20, and MRR, demonstrates this solution's superior performance.

Also in 2019, Du et al. devised EventAction, a visual system designed to deliver and explain recommendations concerning temporal sequences in the context of student advising [Du et al., 2019a]. It processes input data about the current situation and desired output, formulating action plans based on the generated suggestions. EventAction is characterized by finding similar archived records with Euclidean distance, exploring potential outcomes, scrutinizing recommended procedures, and refining the action plans through visual techniques. These four stages were assessed across diverse scenarios involving a student review manager and three graduate students. The experiments reported EventAction's efficacy in executing the stages above and its capacity for coherent navigation.

Following two years, Lonjarret designed an explainable sequential-based RS for the Visiativ company [Lonjarret, 2021]. The system introduces recommendation embedding based on a frequent sequences metric, facilitating the construction of Markov chains with multiple orders to capture sequential dynamics and discern the most pertinent segment of user historical data for consideration in the recommendation task. Moreover, a subgroup discovery method employing different pattern languages is presented to enhance the explainability of the suggestions. Both features undergo evaluation across diverse scenarios and metrics, with the introduced method outperforming state-

of-the-art approaches [Rendle et al., 2010, He et al., 2017, Jannach and Ludewig, 2017, Kang and McAuley, 2018, Tang and Wang, 2018] based on metrics such as AUC, HR at position X (HR_X), and normalized discounted cumulative gain (NDCG) at position X (NDCG_X). Regarding explainability, an analysis of various algorithms [Rendle et al., 2010, Feng et al., 2015, He et al., 2017, Kang and McAuley, 2018] was conducted using AUC and HR metrics. The findings affirm the suggested method's capability to provide local explanations based on user preferences or the sequence of actions performed. Xian et al. proposed $EX^3$, an Extract-Expect-Explain framework addressing the need for explainable attribute-aware item-set recommendation [Xian et al., 2021]. This tool takes various inputs to generate interpretable recommendations, including a query item, a list of candidate items, and their features. The Extract phase employs an attention-based item embedding learning mechanism to reduce the candidate set. In contrast, the Expect phase utilizes an attribute-differentiating network to analyze the item utility relative to the query item. The Explain step uses a bipartite b-matching-based algorithm to construct explainable recommendations based on the filtered items and their utility scores. The evaluation analyses the top-$n$ recommendation performance with the NDCG@10, recall@10, precision metrics, and the attribute ranking with the average score and normalized score. In the first scenario, $EX^3$ is compared with a few approaches [Rendle et al., 2009, Shi et al., 2018, Chen et al., 2020], while in the second case, it is evaluated against random and greedy strategies. In both scenarios, $EX^3$ consistently demonstrates higher performance in providing explainable recommendations.

More recently, Ariza-Casabona et al. implemented a sequence-aware explainable RS that considers user-item review interactions to improve recommendation explainability [Ariza-Casabona et al., 2023]. This approach performs multitask optimization, including rating prediction, context prediction, explanation generation, and next-item prediction. This model uses three masking alternatives: causal, bidirectional, and sequence-adapted PETER. Its evaluation covers rating prediction, explainability, text quality, attention masking role, and the impact of recommendation tasks as regularizers. The presented RS is compared with three techniques [Dong et al., 2017, Li et al., 2017c, Li et al., 2021] with several metrics, including MAE, RMSE, and others. The results indicate that adding historical interaction user data further improves sequence-aware RS.

In 2018, Kerschbaumer proposed a novel two-stage approach for sequence-aware RSs with a convolutional neural network (CNN) [Kerschbaumer, 2018]. The initial level employs CNN to predict the next item a user will likely interact with, generating a list of the top-$n$ items. Subsequently, a ranking network refines this list's ordering by incorporating the item's content information and external data. The evaluation comprises a comparative analysis of CNN for sequence-aware recommendations across diverse hyperparameters and state-of-the-art techniques [Hochreiter and Schmidhuber, 1997, Rendle et al., 2009, Hidasi et al., 2015, Covington et al., 2016], along with assessing the proposed two-stage RS. The performance metrics include precision, average rank, and MRR. Results demonstrate CNN outperforms POOL and RNN methods in the first scenario and exhibits superior performance compared to POOL, particularly with reduced datasets in the second scenario. Smirnova also explored NNs by introducing an RNN model to enhance the next-item prediction by incorporating user browsing history and interactions with recommended

actions [Smirnova, 2018]. The proposed RNN combines action-conditional RNN principles with two fusion mechanisms: early and late fusion. These fusion strategies perform multiplicative integration of the hidden state and action at different positions with the RNN (early fusion at the state transition and late fusion at the output layer). The top-$n$ quality is evaluated by contrasting baseline RRN models (based on navigation and clicks) with action-condition RNNs employing early and late fusion with the precision metric. Findings suggest that an action-condition RNN with late fusion captures recommended item data more effectively than the remaining solutions.

One year later, Yan et al. presented a 2D CNN framework for sequential recommendation [Yan et al., 2019]. It comprises three core modules: the embedding lookup layer, the pairwise encoding structure, and the 2D convolution component. Within the embedding layer, items and users are embedded into matrices, while pairwise encoding builds a three-way tensor atop these embedding matrices to capture complex patterns. The 2D CNN is employed to learn sequential features from the tensor derived earlier. The efficacy is evaluated against other state-of-the-art methodologies [Rendle et al., 2009, Rendle et al., 2010, Hidasi et al., 2015, Tang and Wang, 2018]. Results demonstrate that the introduced method outperforms these approaches regarding MAP (mean average precision), precision, and recall attributes.

In 2021, Le et al. devised an approach to recommend the next items from large sequential databases, given a query item set and its associated attributes [Le et al., 2021]. Their strategy involves item extraction, item-set clustering, transition rule mining, and set-of-items recommendations. Initially, the method aggregates and categorizes similar items from sequential databases, followed by clustering based on the representative item-set vector obtained using density-based spatial clustering of applications with noise. Transition rules are inferred by vectorizing the item values and using pattern mining techniques. Recommendations are formulated by identifying items belonging to the closest item-set type concerning the input item-set. The items recommended are selected based on their frequency and uniqueness. This methodology is evaluated with medical staff in an electronic medical record system, utilizing precision, recall, and F-measures as metrics. The results suggest that the proposed method can recommend item sets with high precision and recall, although poor clustering can heavily degrade the recommendation accuracy.

In 2016, Xie et al. designed a novel graph-based embedding metric capable of learning the embeddings of point-of-interest (POI) within a low-dimensional Euclidean space [Xie et al., 2016]. The method reduces the prediction space while dynamically tracking user preferences. This metric adopts a time decay mechanism, wherein recently visited POIs by users are accorded higher weights. Additionally, the authors invented a graph-based embedding metric with the sequential influence of POIs by integrating sequential pattern analysis into the learning task of POI embeddings. Using the accuracy measure, both methodologies are evaluated with three methods [Rendle et al., 2009, Feng et al., 2015, Wang et al., 2016]. The experimental results demonstrate the superior performance of the proposed techniques.

Four years later, Lim et al. introduced a spatial-temporal-preference user dimensional graph attention network to enhance the next POI recommendation task [Lim et al., 2020]. This model employs local user preferences and self-attention to explore POIs within global spatial-temporal-

preference neighbourhoods. The recommendation process entails the analysis of the advantages conferred by utilizing local and global neighbourhoods in the presented model through the exploration and exploitation of the trade-offs. To evaluate its efficacy, the proposed model is compared with various algorithms [Elman, 1990, Koren et al., 2009, Kong and Wu, 2018, Sun et al., 2020, Zhao et al., 2022] with the accuracy and MAP metrics. The experiments indicate the proposed method outperforms the state-of-the-art methodologies on location-based social networks, terrorism, and transportation datasets. Chang et al. developed the GGLR, a graph-based geographical latent representation model that captures highly non-linear geographical influences between POIs [Chang et al., 2020]. This approach utilizes a graph autoencoder to train the geographical latent representation of two influences (ingoing and outgoing), increasing the geographical influence between two POIs with high visitation frequencies. Furthermore, the authors present a graph-based POI RS with GGLR as its core. The efficiency of the latter component is assessed against other available techniques [Liu et al., 2014, Lian et al., 2014, Li et al., 2015, Zhao et al., 2017, Wang et al., 2018b, Zhou et al., 2019a] with the support of precision, recall, MAP, and NDCG metrics. Experimental findings showcase the superiority of the proposed solution over the remaining solutions, with its performance bolstered by the integration of the GGLR model.

In 2020, Ma et al. introduced a memory-augmented graph NN (GNN), to capture short- and long-term user interests [Ma et al., 2020]. This framework uses GNNs to model items' short-term contextual data while employing a shared memory network to acquire long-range dependencies between items. The representations of short and long-term interests are fused through a gating mechanism. Furthermore, the discovery of co-occurrence patterns among related items is facilitated by a bilinear function. This graph model is evaluated against other baseline methods [Rendle et al., 2009, Hidasi et al., 2015, Hidasi and Karatzoglou, 2017, Kang and McAuley, 2018, Tang and Wang, 2018, Xu et al., 2019a] using the recall and NDCG measures. The experiments show improved performance compared to the remaining techniques. Wang and Cai devised a knowledge-based sequential RS centered on GNNs and memory networks [Wang and Cai, 2020]. This proposal integrates sequential preference with global preference to enhance the prediction of the next user action. In the first stage, this method constructs session graphs from interaction sequences via GNNs to capture complex transitions between neighbour items. Then, in the global preference stage, items are linked with existing KB entities, with this information being integrated into the GNN-based RS through key-value memory networks. The presented RS is benchmarked against many methods within the literature [Rendle et al., 2009, Rendle et al., 2010, Hidasi et al., 2015, Hidasi and Karatzoglou, 2017, Li et al., 2017b, Wu et al., 2018, Huang et al., 2018, Liu et al., 2018], employing recall and NDCG metrics. Experimental findings illustrate the superior performance of the proposed RS over other baseline methodologies.

Throughout 2021, Hsu and Li introduced RetaGNN, a relational temporal attentive GNN designed for holistic sequential recommendation with conventional, inductive, and transferable settings [Hsu and Li, 2021]. This architecture is structured in five layers: an embedding layer, an enclosing subgraphs extraction layer, a relational attentive GNN layer, a sequential self-attention layer, and a final embedding layer. The first module utilizes a feed-forward network to produce

embeddings for users, items, and attributes, while the second extracts high-order relationships between these entities using $h$-hop enclosing subgraphs. The third layer employs a relational attention mechanism and message passing between nodes to learn the representations of users and items. The fourth layer models sequential patterns of user preferences and item embeddings, whereas the last layer combines these representations to generate predictions. RetaGNN's performance is evaluated with other approaches [Kang and McAuley, 2018, Ma et al., 2019, Ma et al., 2020, Wang et al., 2019c, Zhang and Chen, 2019, Su et al., 2021, Peng et al., 2021] with the precision, recall, and NDCG metrics. The results indicate that RetaGNN outperforms the other state-of-the-art techniques across conventional, inductive, and transferable configurations. Chang et al. established a sequential recommendation GNN to accommodate dynamic user preferences while handling noisy and implicit preference signals [Chang et al., 2021]. This strategy comprises four main components: interest graph construction, interest-fusion graph convolutional layer, interest-extraction graph pooling layer, and the prediction layer. The initial module transforms loose item sequences into tight item-item graphs, distinguishing different interests. The subsequent element employs an attentive graph convolutional network to fuse the user's preferences, discerning strong behaviours from weaker ones. The third module utilizes user dynamic graph pooling to retain active core preferences crucial for predicting the next user behaviours. The last stage estimates the item most likely to be interacted with by the user. Its performance is evaluated against various approaches [Hidasi et al., 2015, He et al., 2017, Zhou et al., 2018a, Tang and Wang, 2018, Zhou et al., 2019b, Yu et al., 2019, He et al., 2020] through metrics such as AUC, group AUC, MRR, and NDCG. The conducted experiments reveal that the introduced approach outperforms these methodologies and models in long behavioural sequences effectively. Fan et al. presented TGSRec, a temporal graph sequential RS that integrates temporal collaborative signals to enhance the recommendation quality [Fan et al., 2021]. A temporal collaborative transformer and a graph information propagation mechanism characterize this framework. The latter layer exploits collaborative attention to model the collaborative signals from users and items and accounts for the item sequences' temporal dynamics. The second component hinges on a continuous bipartite graph where the sequential patterns and temporal collaborative signals are fused, enriching the recommendation task. TGSREC undergoes benchmarking with ten techniques [Rendle et al., 2009, Hidasi et al., 2015, Wu et al., 2018, Tang and Wang, 2018, Kang and McAuley, 2018, Nguyen et al., 2018, Sun et al., 2019, Li et al., 2020, He et al., 2020, Wu et al., 2020] using the evaluation metrics recall and MRR. Experimental findings indicate that TGSRec significantly outperforms the state-of-the-art algorithms in sequential recommendation with temporal collaborative signals.

Table 4.4 illustrates the main aspects addressed by each approach studied previously. The state-of-the-art is organized based on three areas: survey papers reviewing the current state of sequence-aware RSs, studies focusing on the explainability to make recommendations more transparent and interpretable for users, and techniques for generating sequence-aware suggestions, including graphs, SPM, GNN, NN, and attention.

Table 4.4: Sequence-aware related work summary

| Ref | Survey | Explainability | Graph-POI recommendation | SPM | GNN | NN | Attention |
|---|---|---|---|---|---|---|---|
| [Xie et al., 2016] | - | - | ✓ | - | - | - | - |
| [Quadrana et al., 2018] | ✓ | - | ✓ | ✓ | - | ✓ | - |
| [Kang and McAuley, 2018] | - | - | - | - | - | - | ✓ |
| [Kerschbaumer, 2018] | - | - | - | - | - | ✓ | - |
| [Smirnova, 2018] | - | - | - | - | - | ✓ | - |
| [Chu et al., 2019] | - | - | - | - | - | - | ✓ |
| [Yan et al., 2019] | - | - | - | - | - | ✓ | - |
| [Ma et al., 2020] | - | - | - | - | ✓ | - | ✓ |
| [Chen et al., 2019] | - | - | - | - | - | ✓ | ✓ |
| [Du et al., 2019a] | - | ✓ | - | - | - | - | - |
| [Karlapalepu, 2020] | ✓ | - | - | ✓ | - | - | - |
| [Chang et al., 2020] | - | - | ✓ | - | ✓ | - | - |
| [Lim et al., 2020] | - | - | ✓ | ✓ | - | - | ✓ |
| [Wang and Cai, 2020] | - | - | - | - | ✓ | - | ✓ |
| [Fang et al., 2020] | ✓ | ✓ | - | - | ✓ | ✓ | ✓ |
| [Lonjarret, 2021] | - | ✓ | - | - | - | - | - |
| [Xian et al., 2021] | - | ✓ | - | - | - | - | - |
| [Le et al., 2021] | - | - | - | ✓ | - | - | - |
| [Hsu and Li, 2021] | - | - | - | - | ✓ | - | ✓ |
| [Chang et al., 2021] | - | - | - | - | ✓ | - | ✓ |
| [Fan et al., 2021] | - | - | - | - | ✓ | - | ✓ |
| [Ariza-Casabona et al., 2023] | - | ✓ | - | - | - | - | - |

While the research community has disseminated some surveys in this area and proposed methodologies to enhance the explainability level, the predominant focus remains on concrete solutions for optimizing the next item/sequence recommendation task. Diverse attention mechanisms allow NNs and GNNs to effectively allocate attention to the most relevant elements within the input set. Moreover, graph-POI graph structures and sequential pattern mining are other problems explored, with the latter being less frequent in the literature.

The proliferation of various research projects exploring explainability within their recommendation process renders them particularly pertinent for helpdesk solutions. In such contexts, where attributes such as confidence and trust play vital roles in the analysis and resolution of tickets, the conclusions provided by these studies prove their applicability within the helpdesk domain.

### 4.2.4 Cold-start

As uncovered previously, one of the most common factors limiting the performance of RS structures and addressed in our proposal is the presence of cold-start scenarios. Nonetheless, the research

community invested efforts in developing countermeasures against cold-start over the years. The subsequent research articles are described chronologically, focusing on surveys, social networks, user profile expansion, context features, predictive modeling, NNs, and hybrid solutions.

In 2015, Berardi et al. investigated continuous cold-start and their impact on content and context-based e-commerce RSs [Bernardi et al., 2015]. This phenomenon arises when users or items experience infrequent updates, resulting in various entities remaining in a cool-down state. Despite the research community's myriad approaches, a gap exists in addressing their continuous effect. The authors view content-based and contextual recommendation strategies as the most effective coping tools for this problem.

Two years later, Gope and Jain offer a comprehensive survey delineating techniques for dealing with cold-start users in RSs [Gope and Jain, 2017]. These methods are stratified based on the nature of the information, dividing it into explicit, such as active learning and interviews, and implicit techniques, including social networks and adapted traditional filtering mechanisms. Additionally, the authors compare the available methods, advocating adopting hybrid solutions to cope with the limitations inherent in individual techniques.

In 2021, Sethi and Mehrotra investigated the cold-start phenomenon in various fields, elucidating the available techniques [Sethi and Mehrotra, 2021]. In addition to providing a concise overview of the cold-start concept concerning users and items, the authors delineate several domains where this challenge prominently manifests, including e-commerce, crowdsourcing, food journalism, location-based social networks, and news recommendations. Furthermore, they categorize approaches to treating cold-start into three groups: adaptive traditional methods, social network-based strategies, and DL-based solutions.

In 2010, Zhang et al. promoted a diffusion-based recommendation framework based on social tags to augment the data between users and items [Zhang et al., 2010]. Through analysis of tag frequencies, the method discerns user preferences across diverse topics, while tag-object associations delineate semantic correlations. The authors conduct comparative assessments of their approach against user-object-tag diffusion and user-tag-object diffusion, employing ranking score (RS), inter-diversity, and inner diversity metrics. Empirical evaluations reveal improved accuracy with the advocated method.

After five years, Zou et al. developed TrustRank, an algorithm that employs a user-trust network to handle cold-start [Zou et al., 2015]. These social linkages encapsulate friendships and trust relationships among diverse users under the assumption that individuals with established friendships exhibit similar preferences. To mitigate the limited information concerning these linkages, a PageRank methodology is employed to expand the user's network, encompassing more users with similar preferences. The efficiency of their approach and its variations are evaluated against various solutions [Shardanand and Maes, 1995, Golbeck et al., 2003, Deshpande and Karypis, 2004, Massa and Avesani, 2004, Massa and Bhattacharjee, 2004, Golbeck and Hendler, 2005], utilizing metrics, such as MAE, the standard deviation of the MAE (SDM), HR, and average reciprocal hit-rank (ARHR). Experiment findings reveal TrustRank's enhanced performance in cold-start scenarios compared to conventional user-based and item-based CF solutions.

In 2013, Guo introduced three methodologies capable of improving user preference modeling by incorporating user behaviours and social relationships [Guo, 2013]. The initial strategy, the Merge approach, entails the construction of rating profiles for active users through the combination of ratings from trusted neighbours. The second technique utilizes a novel Bayesian Similarity (BS) measure, accounting for both the direction and magnitude of rating profiles. Lastly, the utilization of prior ratings based on experiences garnered from virtual reality environments is proposed. These solutions are evaluated using different methods and metrics. The Merge approach is benchmarked against other state-of-the-art algorithms [Massa and Avesani, 2007, Chowdhury et al., 2009, Ray and Mahanti, 2010] with metrics, such as MAE, F1, and rating coverage. The performance of BS is assessed against the Pearson correlation coefficient, Cosine similarity, mean square distance, PIP [Ahn, 2008], and SM [Bobadilla et al., 2012a] using the MAE metric. Furthermore, the efficiency of the prior rating technique is validated through a user study involving the exploration of two different virtual reality scenarios. Results demonstrate the efficacy of the proposed methods in handling cold-start cases while surpassing other existing approaches.

In 2020, Herce-Zelaya et al. designed a behavioural social stream-based RS (BSSB-RS) that uses social networks' user profiles in conjunction with classification trees and RF for predictive tasks [Herce-Zelaya et al., 2020]. This framework harnesses Twitter data to generate user profiles utilized in prediction models to determine the sustainability of recommending specific item(s) to individual users. User segmentation based on behavioural patterns is accomplished by applying CT and RF methodologies, enhancing recommendation accuracy. The performance evaluation of the CT and RF methods integrated into the BSSB-RS tool is conducted using precision, RMSE, and F-measure metrics, with the latter showing better performance. Moreover, BSSB-RS is benchmarked with five state-of-the-art methods, including NHSM [Son, 2016], MJD [Bobadilla et al., 2012b], and others, with the current method achieving the lowest MAE value.

In 2010, Shaw et al. proposed the application of association rules to enrich the user's profile [Shaw et al., 2010]. This methodology builds a taxonomy-driven product recommender [Ziegler et al., 2004]), which mines association rules between topics that interest users, expanding their profiles. These association rules are derived with MinMax, ReliableBasis, MinMax with HRR, and ReliableBasis with HRR. Through evaluation metrics, such as precision, recall, and F1-measure, their approach demonstrates enhanced performance in RSs.

Four years later, Braunhofer et al. integrated user personality traits with active learning approaches in tourism CARSs [Braunhofer et al., 2014]. Employing an extended MF model, the authors incorporate the Five Factor Model [Mccrae and Costa, 1996] to delineate user personalities, encapsulating dimensions such as openness, conscientiousness, extroversion, agreeableness, and neuroticism. [Mccrae and Costa, 1996]. Furthermore, another augmented MF model is proposed to compose personalized user suggestions. Live user studies explore these techniques within an Android-based RS (South Tyrol Suggests) [Braunhofer et al., 2013]. The experiments conducted show the efficiency and utility of the implemented strategies.

In 2020, Natarajan et al. introduced an RS employing linked open data (RS-LOD) to overcome cold-start users/items in CF [Natarajan et al., 2020]. Based on the extensive knowledge encapsulated

within the LOD cloud, spanning diverse domains, RS-LOD incorporates semantic features about new entities, augmenting recommendation accuracy in this context. Empirical evaluation on a Netflix dataset reveals that the RS-LOD system exhibits lower MAE than other existing strategies.

In 2014, Zhang et al. introduced a context-aware semi-supervised co-training system to address cold-start in RSs [Zhang et al., 2014]. Using a context-aware factorization model to capture additional user and item data, this system builds recommendation models with a semi-supervised co-training technique. This approach incorporates unlabeled data and generates different sub-models from different contexts, which are aggregated into an ensemble method. The experimental setup includes two real-world datasets and various standard algorithms, including K-NN, factorization CF [Bell et al., 2007], and others, and utilizes the RMSE measure for evaluations. The results obtained show enhanced system performance and cold-start alleviation. Nguyen et al. devised A-LinUCB, an extended variant of Linear Upper Confidence Bound (LinUCB), to tackle cold-start problems framed as contextual-bandit issues [Nguyen et al., 2014]. This technique harnesses past user ratings as context for new users, exhibiting heightened efficiency in handling the context description matrix compared to the LinUCB approach. Experimental evaluations entail the benchmarking A-LinUCB against a spectrum of multi-armed bandit algorithms [Auer et al., 2002a, Auer et al., 2002a, Auer et al., 2002b, Agrawal and Goyal, 2013]. Results illustrate the superiority of the implemented strategy in terms of the cumulative regret online measure.

In 2011, Kim et al. surveyed error-reflected models aimed to enhance CF systems [Kim et al., 2011]. Their algorithm generates error-reflected models through explicit ratings, employed for predictive purposes in scenarios characterized by sparse data between users and items. The proposed method is evaluated against benchmark techniques, with the prediction accuracy to the cold-start cases (items and users) assessed using the MAE metric. Experimental findings indicate superior prediction accuracy compared to benchmark alternatives.

Throughout 2015, Pereira and Hruschka introduced a hybrid RS based on simultaneous co-clustering and learning (SCOAL) [Pereira and Hruschka, 2015]. The SCOAL module identifies clusters of users with similar preferences and builds prediction models for each cluster [Deodhar and Ghosh, 2010]. Considering these models, the hybrid RS searches for the most suitable cluster for a new user or item. Evaluation of this implementation entails the assessment of the normalized mean average in two cases: users with a limited number of ratings (partial cold-start) and new users devoid of any ratings (pure cold-start). In the former scenario, performance metrics, such as cluster with minimum error (CME), Pearson correlation, Cosine similarity, and PIP, are utilized. For pure cold-start cases, weighted prediction, dynamic classification, weighted prediction with naive Bayes, DT, and logistic regression, and dynamic classification with naive Bayes are employed. Results indicate that CME performs better for partial cold-start, while dynamic classification is recommended for pure cold-start.

Two years later, Volkovs et al. devised a content-based RS for offline and online tests during the ACM Recommender Systems Challenge'17 [Volkovs et al., 2017b]. Their model predicts the interaction probability between user-item pairs, translating this interactivity into binary positive or negative outcomes. The proposed framework integrates five types of input: user content, item

content, user-item content, user-item interaction, and user-item temporal. These inputs are fed into classifiers, including deep NN and gradient boosting machines, which generate predictions regarding the likelihood of positive or negative interactions between the users and items. Subsequently, various approaches are explored to prune candidate pairs, such as top-100, probability threshold, and alternative strategies. The proposed solution outperformed the remaining teams in terms of AUC and leaderboard score metrics in offline and online competition stages.

In 2010, Basiri et al. advanced with an ordered weighted averaging (OWA) technique for mitigating user cold-start [Basiri et al., 2010]. This proposal integrates the outputs of five recommendation classifiers: CF, content-based, CF with demographic-based, demographic-based, and CF with content-based. Using the precision metric for evaluation, the OWA approach outperforms alternative methods [Pazzani, 1999, Delgado and Ishii, 1999].

After four years, Lika et al. presented a mechanism integrating classification methods, similarity techniques, and prediction approaches [Lika et al., 2014]. Their system utilizes naive Bayes and C4.5 methods to classify users into specific groups while employing custom metrics to identify similar users based on demographic attributes. A comparative analysis tests the performance of these approaches with a random classification method, which randomly allocates users into predefined classes. Moreover, different weightings of demographic attributes are inspected. Based on the MAE and RMSE metrics, the proposed solution reveals better performance in scenarios characterized by a large user base.

In 2015, Idrissi et al. implemented a hybrid approach comprising user demographic data, semantic information of non-demographic data, and SVD-based collaborative recommendations [Idrissi et al., 2019]. This system retrieves and normalizes user demographic data, which is clustered using K-means to group users according to their demographic attributes. Additionally, non-demographic user data is vectorized to facilitate the identification of similar user profiles. Concurrently, an SVD-based CF method is deployed to overcome traditional CF limitations while uncovering latent similarities among users. Evaluation of the proposed method involves cluster validation to assess its efficacy in handling cold-start users, comparing its performance against matrix completion [Candes and Recht, 2012], and graph regularized alternating least squares [Rao et al., 2015] through the RMSE metric. Results demonstrate the effectiveness of the hybrid solution in mitigating cold-start and showcase improved performance compared to standard approaches.

In 2021, Fent et al. developed a Bayesian personalized ranking (RBPR) model integrating explicit and implicit feedback data to overcome cold-start while improving ranking performance [Feng et al., 2021]. This model merges a probabilistic MF (PMF) for explicit data and a Bayesian personalized ranking (BPR) for implicit feedback data. The BPR technique enables RBPR to extract implicit user and item features from implicit feedback data, while PMF facilitates the collection of explicit features from rating data. In terms of evaluation, the proposed solution is tested against other baseline algorithms [Salakhutdinov and Mnih, 2007, Rendle et al., 2009, Shi et al., 2012, Pan and Chen, 2013b, Pan and Chen, 2013a, Luo et al., 2015, Li et al., 2017a]. RBPR demonstrates superior performance in precision@3, precision@5, recall@3, recall@5, MAP, and MRR.

One year later, Kawai et al. proposed two LDA-based topic models to predict ratings for new

users/items [Kawai et al., 2022]. These models, namely multiple correspondence LDA (MCLDA) and multiple joint LDA (MJLDA), incorporate content-based filtering, allowing the analysis of item topics, item feature topics, and user trait topics. While MCLDA and MJLDA are conceptually similar, a subtle distinction lies in the independence of topics between auxiliary data about users and items and the topic of the item in the MJLDA procedure. These topic models are evaluated against various methodologies [Kawai et al., 2018] with diverse metrics encompassing precision, recall, F-1, MAE, and perplexity. The findings emphasize the superior performance of the proposed topic compared to the remaining approaches. Moreover, MCLDA and MJLDA exhibit closely aligned performance across the referred metrics.

In 2021, Wei et al. reformulated the cold-start item representation learning from the information-theoretic view and introduced CLCRec, a contrastive learning-based cold-start RS [Wei et al., 2021]. This framework for representation learning integrates two essential principles: mutual information between user and item embeddings (U-I) and mutual information between feature representation and collaborative embeddings (R-E). The authors establish a contrastive learning function integrated within the CLCRec system to optimize these principles. This tool comprises three core segments: contrastive pair organization, contrastive embedding network, and contrastive optimization. The contrastive pair organization segment categorizes pairs positively and negatively based on their similarity across U-I and R-E pairs. Subsequently, the contrastive embedding network generates contrastive embedding networks and computes the interaction between the users and items. Finally, the optimization structure refines model parameters with a contrastive learning algorithm. Based on the results obtained with the NDCG@10 and recall@10 metrics, CLCRec outperformed other state-of-the-art algorithms [Rendle et al., 2009, Geng et al., 2015, Volkovs et al., 2017a, Barkan et al., 2019, Zhu et al., 2020, He et al., 2020, Du et al., 2020].

Two years later, Zhang et al. conducted a study on the implicit embedding net based on expert models to address cold-start and warm-up users in the matching stage of RSs [Zhang et al., 2023]. This methodology comprises two distinct subnets, namely original and bias subnets, which integrate a gate network to harmonize the results generated by the two experts. Moreover, dynamic knowledge distillation is employed, featuring a teacher selector mechanism designed to facilitate experts in assimilating adequate data from cold-start users. Additionally, the bias subnet is instrumental in modeling user behavior bias. By combining both subnets, latent user behavioural patterns can be discerned. The efficacy of the proposed solution is evaluated against other state-of-the-art methods [Rendle et al., 2011, Huang et al., 2013, Covington et al., 2016, Li et al., 2019, Chai et al., 2022]. Comprehensive assessment utilizing metrics such as HR, NDCG, and DCG establishes the superiority of the cold and warm net over all competing models.

In 2020, Choi et al. investigated the utility of weak supervision in forecasting the average ratings within item-side cold-start problems. [Choi et al., 2020]. Their system entails the derivation of genre preferences as item features, the identification of representative users, and the computation of average preferences for a particular item through content-based filtering. This methodology is compared with other CF frameworks, including traditional CF and neural CF (NCF), with the MAE metric. The results reveal a superiority of the proposed method, demonstrating an enhanced

accuracy by 21% and 38% in contrast with CF and NCF techniques, respectively. Dong et al. devised MAMO, a memory-augmented meta-optimization model aimed at mitigating cold-start limitations inherent in RSs [Dong et al., 2020]. This tool integrates two distinct memory components tailored to facilitate optimal recommendation optimization: feature-based and task-based mechanisms. The former element defines a custom bias term for model initialization, accomplished through user profile and user embedding memory matrices. Meanwhile, the latter module aids the model in acquiring knowledge about similar user preferences across different items. Regarding evaluation, MAMO is compared to various techniques [Chen et al., 2018c, Lee et al., 2019, Bharadhwaj, 2019, Du et al., 2019b] with the MAE and NDCG measures. Notably, MAMO showcases improved performance in cold-start scenarios.

Table 4.5 provides an overview of research articles addressing the cold-start problem with diverse techniques. The literature review is organized into four main groups: survey papers addressing the cold-start problem in RSs, hybrid solutions combining different strategies, approaches that try to gather more information about the users and items such as from social networks, user profile enrichment, and context features, and techniques that mitigate cold-start patterns, including predictive modeling, and NNs.

Table 4.5: Cold-start related work summary

| Ref | Survey | Hybrid | Social Networks | User Profile Expansion | Context Feature | Predictive Modeling | NN |
|---|---|---|---|---|---|---|---|
| [Shaw et al., 2010] | - | - | - | ✓ | - | - | - |
| [Basiri et al., 2010] | - | ✓ | - | - | - | ✓ | |
| [Zhang et al., 2010] | - | - | ✓ | - | - | - | - |
| [Kim et al., 2011] | - | ✓ | - | - | - | ✓ | - |
| [Guo, 2013] | - | - | ✓ | - | - | - | - |
| [Zhang et al., 2014] | - | - | - | ✓ | ✓ | ✓ | - |
| [Nguyen et al., 2014] | - | - | - | - | ✓ | - | - |
| [Lika et al., 2014] | - | ✓ | - | - | - | ✓ | - |
| [Braunhofer et al., 2014] | - | - | - | ✓ | ✓ | ✓ | - |
| [Pereira and Hruschka, 2015] | - | ✓ | - | - | - | ✓ | - |
| [Zou et al., 2015] | - | - | ✓ | - | - | - | - |
| [Bernardi et al., 2015] | ✓ | ✓ | - | - | ✓ | - | - |
| [Gope and Jain, 2017] | ✓ | - | ✓ | ✓ | - | - | - |
| [Volkovs et al., 2017b] | - | - | - | - | - | ✓ | ✓ |
| [Idrissi et al., 2019] | - | ✓ | - | - | ✓ | - | |
| [Herce-Zelaya et al., 2020] | - | - | ✓ | ✓ | - | - | - |
| [Natarajan et al., 2020] | - | - | - | ✓ | - | - | - |
| [Choi et al., 2020] | - | ✓ | - | - | - | - | - |
| [Dong et al., 2020] | - | - | - | - | - | ✓ | ✓ |
| [Sethi and Mehrotra, 2021] | ✓ | - | ✓ | ✓ | - | ✓ | - |
| [Feng et al., 2021] | - | ✓ | - | - | - | - | - |
| [Wei et al., 2021] | - | - | - | - | - | ✓ | ✓ |

Table 4.5: Cold-start related work summary

| Ref | Survey | Hybrid | Social Networks | User Profile Expansion | Context Feature | Predictive Modeling | NN |
|---|---|---|---|---|---|---|---|
| [Kawai et al., 2022] | - | ✓ | - | - | - | - | - |
| [Zhang et al., 2023] | - | - | - | - | ✓ | ✓ | - |

Most of the exposed works utilize hybrid approaches, while alternative methodologies explore the integration of social network data and contextual information, augmentation of user profiles, and utilization of NNs and other predictive models to mitigate this challenge.

Topics such as sparsity and privacy are not explored here since the application of synthetic data generation (studied in chapter 3.1) can mitigate these issues. The last part of this work contemplates the potential scalability limitations inherent in this proposal.

### 4.2.5 Discussion

After a few decades, RSs have become essential tools for mitigating the challenges of consumer over-choice. With the escalating complexity of data and abundant online resources, users often struggle to make decisions among the overwhelming number of options. RSs employ predictive algorithms to anticipate user preferences and deliver recommendations for the most relevant items.

This chapter highlights six main types of recommendation techniques: collaborative filtering employs user data to find similar users for item recommendations; content-based analyzes item features to infer user interests; knowledge-based uses domain knowledge for tailored suggestions; demographic-based relies on user-specific information to provide recommendations; community-based suggests products based on friends' preferences; and hybrid models combine various methods to enhance accuracy while dealing with cold-start, scalability, serendipitous, and other issues. These methods can be further refined by considering the context of the application and the sequence of operations that lead to the next decision.

As shown in this thesis, RSs have made an impact in the field of cybersecurity. According to Gadepally et al., integrating recommendation features in this domain can reduce response times to cyber threats [Gadepally et al., 2016]. Cybersecurity operators often face overwhelming amounts of data, which may result in missing critical details or identifying threats only after an attack occurs. RSs address these challenges by assisting IT analysts in filtering and prioritizing information based on contextual factors such as severity and resources, thereby streamlining threat response. Nonetheless, there is an absence of tools capable of suggesting specific actions to security teams to resolve different incidents. While research on this topic is limited, existing studies emphasize the value of integrating RSs into the cyber domain [Lyons, 2014]. Currently, companies rely on

established standards, Information Technology Infrastructure Library[18] and ISO20.000[19] to align IT service delivery with business goals. Still, these certificates focus on providing recommendations and advice on management structure, but they cannot handle cyberattacks.

The related work reveals several research directions for advancing RSs in the cybersecurity area. One key improvement is developing systems that can recommend the most suitable operator to handle an incident while providing the appropriate procedure and detailed justification for the recommendation. Clear explanations about recommendations are critical to promote trust and transparency, especially in cybersecurity environments.

Moreover, a significant research gap lies in how cybersecurity RSs should respond when operators reject a recommendation. Current systems often assume user compliance; however, dynamic decision-making in this field means operators may decline recommendations for various reasons. Further work is needed to implement mechanisms that enable RSs to adapt to these rejections and improve their recommendations over time.

Another key area for improvement is related to cold-start, which arises when there is insufficient data to make informed recommendations, particularly for novel incidents. Although existing solutions have been developed to address information scarcity in vulnerability databases like CVE, NVD, and others [Gasmi et al., 2019], these methods focus on populating cybersecurity knowledge bases using external sources. There is a lack of research on handling incidents when no online data is available. Such incidents may involve zero-day vulnerabilities or emerging threats that have yet to be cataloged.

Lastly, the research community has explored some SOC evaluation metrics (section 2.3), such as time spent on each ticket and the average response time. However, additional relevant metrics could be inspected. For example, metrics that assess wait time operators experience in receiving recommendations, memory spent during the recommendation generation, and factors related to procedure adherence or/and the refusal of recommendations by operators.

## 4.3 Proposed Architecture for CROSS

CROSS is characterized by its ability to identify the fastest operator-action pair using pre-filtering techniques, handle cold-start cases through incident similarity analysis and crossover action generation, adjust suggestions based on operator refusals, and continuously adapt from each treatment procedure performed. Again, the term operators comprises all the roles discussed in section 2.1.

Like SNOOKER, CROSS is supported with an interface implemented with PyQT[20]. From this point forward, for clarity purposes, the incident type will also be referred to as family, and the incident sub-type as subfamily. Figure 4.15 illustrates the main operations in the CROSS workflow.

[18]Documentation available at https://www.ibm.com/topics/it-infrastructure-library

[19]Documentation available at https://pecb.com/en/education-and-certification-for-individuals/iso-iec-20000

[20]Documentation available at https://wiki.python.org/moin/PyQt

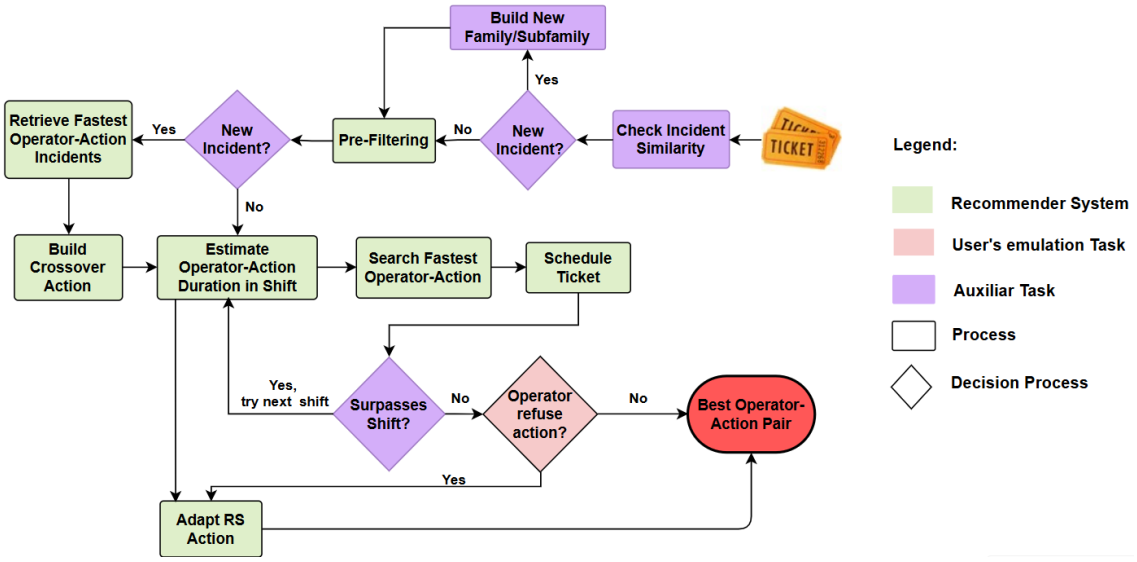



Figure 4.15: CROSS Ticket Treatment

For each incident, CROSS evaluates its novelty and follows a series of steps culminating in identifying the operator responsible for closing it and the corresponding action. The following sections provide more details regarding the tasks performed.

### 4.3.1 Input

As shown in Fig.1.1, SNOOKER was used to simulate training and testing data in the absence of real-world data to evaluate CROSS's ticket treatment. The training set emulates tickets previously processed by SOC operators. After processing the training set, CROSS assimilates various details regarding the families, subfamilies, and outliers, refining its knowledge about outlier data, family features matrix, family and subfamily actions, and other factors.

The testing dataset simulates tickets awaiting resolution by SOC teams. As discussed in the previous chapter, each ticket is characterized by numerous attributes, including but not limited to geographical location, raised and fixed date, and client, among other properties. Supplementary data regarding the teams, shifts, priority levels, and other features is extracted from a configuration file used by SNOOKER. It is important to note that CROSS does not have access to all the information available in SNOOKER. The goal is to maintain two distinct systems: SNOOKER acts as a real-life operator emulator serving as the ground truth for systems validation, and CROSS independently suggests the fastest pair for treating the testing tickets while having access only to data it would have in a real environment. By observing the ticket resolution process, CROSS improves its performance over time without relying on complete data access.

### 4.3.2 Cold-Start

Before delivering a recommendation, CROSS determines whether the incidents (family or subfamily) represent a cold-start scenario. If the incident is recognized, CROSS analyzes the fastest

operator-action combination. Conversely, in cases where the family remains unknown, the system searches similar types, generates a crossover action from them, and estimates the fastest operator to execute it.

The analysis of similar incidents in cold-start scenarios for unknown families is accomplished through a matrix cataloging all existing features of the families within the CROSS domain knowledge. This matrix, built during the analysis of the training dataset, inspects the features associated with both known and unknown incidents. Such a matrix is represented with dimensions $n \times m$, where $n$ represents the number of event types or families, and $m$ signifies the number of features associated with each event type. This approach follows the inverse principle of Hamming distance (discussed in section 4.1.3), where similarity is assessed based on list differences. This matrix employs a binary encoding scheme, where 1 indicates the existence of a specific feature, while 0 denotes its absence. This structure facilitates the comparison of known incident attributes with those of unknown events. Then, CROSS identifies those with the highest similar feature distribution. In case no family matches the family's registered features, a novel family is created and introduced in the system. The generation of new families is ensured by SNOOKER (detailed in section 3.2.2.2).

Table 4.6: Cold-Start example

| Family | Feature 1 | Feature 2 | Feature 3 | Feature 4 |
|---|---|---|---|---|
| A | 1 | 0 | 1 | 0 |
| B | 0 | 1 | 0 | 1 |
| C | 1 | 1 | 1 | 0 |
| Unknown 1 | 0 | 1 | 0 | 1 |
| Unknown 2 | 1 | 1 | 0 | 1 |

Considering the families A, B, C, and two unknown families in Table 4.6, CROSS classifies unknown 1 as existing since its feature vector (0, 1, 0, 1) matches that of family B (0, 1, 0, 1). In contrast, the other unknown family is categorized as new since there are no existing families with the same set of features (1, 1, 0, 1).

If the subfamily is unknown, the system counts the number of unique subfamilies within the current family, creates a new one, and assigns it to the ticket. The remaining information about the new subfamily is based on the data of similar subfamilies.

### 4.3.3 Pre-Filtering

Pre-filtering operations are performed on historical data, narrowing the focus to instances that share similarities with the current ticket. The contextual factors considered depend on the specific domain being explored. In the case of synthetic datasets related to cybersecurity, these contextual data include the team, client, subfamily, and timestamp associated with each step analyzed.

Figure 4.16 illustrates the pre-filtering steps followed by CROSS to identify the relevant historical data.

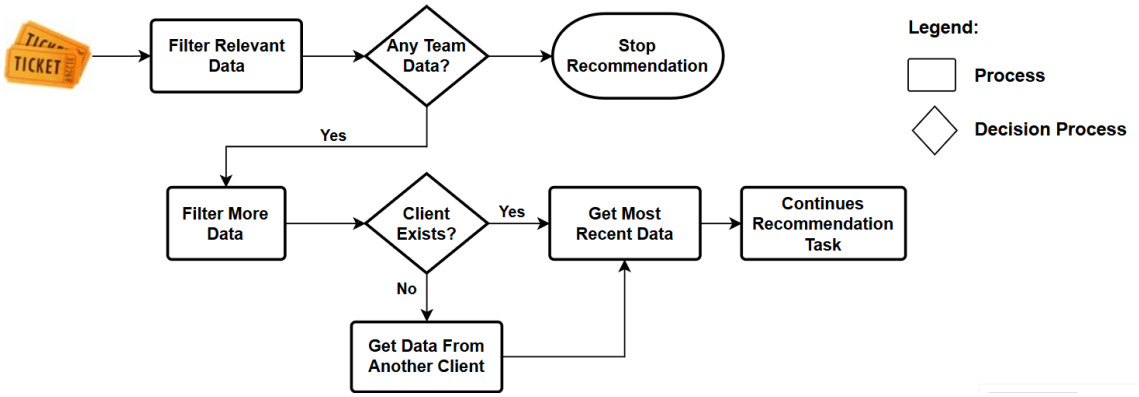



Figure 4.16: CROSS Pre-Filtering

CROSS begins by filtering the tickets solved by the ticket's assigned team. The filtering continues if relevant data is available; otherwise, the system refrains from providing suggestions. Next, a second filtering stage refines the selection by considering the ticket's subfamily and client context. If the client is new, CROSS retrieves data from another client within the subfamily (each subfamily may have multiple associated clients). One is selected at random, as the current version of CROSS does not differentiate individual clients within a subfamily.

Once available historical data is discovered, CROSS sorts it from the most recent to the least recent and computes the time difference between the filtered and current tickets. This methodology guarantees that recommendations provided to operators are based on the most up-to-date procedures.

As discussed in section 3.2.2.2, the actions employed by operators consist of a series of steps, some of which establish initialization, closure, and transfer functions. While the first and second steps initiate and close ticket resolution, the last allows ticket escalation to teams with more domain expertise. Each historical step has an associated duration, indicating the time taken to complete it. The duration of actions with transfer steps is contingent on subsequent actions executed by higher-level teams. For example, if the last step of the action (for instance, ["a", "b", "c"]) carried out by operator A in team Y is a transfer step, its total duration comprises the time spent by operator B in a more expert team.

### 4.3.4 Recommendation Task

Following the analysis and filtering of the recent procedures used by various operators in the ticket subfamily or similar event types, the RS recommends the fastest operator-action pairing. To enhance scalability, CROSS employs a strategy similar to a sliding window during this operation (tested in section 4.4.1). Furthermore, it improves memory efficiency by limiting the duration estimation of potential actions to the most recent $n$ actions, reducing the need to store or process older data.

Equation 4.15 translates the primary goal of the recommendation task based on the ticket being treated and its inherent context.

$$R : T_i \times C_i \rightarrow (u, p) \tag{4.15}$$

where $T_i$ represents the ticket features, $C_i$ the contextual factors, and $u$ and $p$ the operator and procedure assigned, respectively.

For known subfamilies, CROSS estimates the duration of each operator's actions based on data collected in the previous treatment phase. For each tuple $\langle Operator, Subfamily, Step \rangle$ of each action, the system verifies the data availability aligned with the ticket timestamp. If no records are detected for a given tuple, a prediction module is invoked to estimate its duration. Similar to the collection of the most recent $n$ actions, this forecasting structure, based on nonlinear least squares regression, employs the latest $n$ steps to predict the duration of each step (evaluated in section 4.4.1). This approach, formalized in Eq. 4.16, is an optimized linear regression version for handling data with nonlinear features, seeking the $\beta$ that would minimize the sum of squared residual errors.

$$S = \sum_{i=i}^{n} [y_i - f(x_i, \beta)]^2 \tag{4.16}$$

where $S$ represents the sum of squared residuals, $\mathrm{n}$ the number of data points, $y_1$ the observed value at $x_i$, f(x) the nonlinear function, and $\beta$ the coefficient to be estimated.

An additional strategy was initially considered before exclusively applying nonlinear least squares regression. This dual-approach rationale arose from the need for multiple historical observations with nonlinear least squares regression. When insufficient past timestamps for a given step were available, the predicted duration was instead calculated as the average of all available past durations. However, as demonstrated in the chapter 3.3, the nonlinear least squares regression consistently outperformed the averaging method across all tested n steps, leading to exclusive adoption of this strategy.

When background information about the tuple is unavailable, the duration is estimated by analyzing the steps employed in similar incident types. The strategy detailed previously is applied to all similar subfamilies until a successful estimation of the step duration for the targeted timestamp is achieved. If the step is not detected in similar events, this experiment is repeated for other operators who have used the same step.

Algorithm 1 describes how CROSS computes the relative speeds between different operators within the same team. This information is valuable for estimating the duration of a step for an operator, even if he has not previously executed that specific step. After initializing the relative speeds for the current team and subfamily, the system identifies other operators within the team who worked on the same subfamily. For each of these operators, it finds the steps they have in common with the current operator. Based on the total durations of the last occurrences of these shared steps, a relative speed is calculated for the current operator and other operators. In extreme cases where estimating the step duration proves impossible, the duration of a step is set to the average of all step durations taken by the corresponding operator within the ticket subfamily or similar subfamilies.

For novel incidents, CROSS begins its analysis of the fastest operator-action pairs by examining the similarity matrix of incident types related to the ticket subfamily, as detailed in section 4.3.2. After identifying the most similar families and subfamilies (those with the highest similar feature distribution in the similarity matrix), CROSS generates a crossover action based on their fastest

**Algorithm 1:** Update Operators Relative Speeds

```
Input: subfamilies_actions_steps, teams_data, team, subfamily,
       operator, operators_relative_speed
Output: Updated operators_relative_speed
1  if subfamily ∉ operators_relative_speed[team] then
2      initialize_relative_speed(team, subfamily)
3  foreach other_operator ∈ get_operators(team) do
4      if operator ≠ other_operator then
5          if has_data_for_both(subfamily, team, operator,
            other_operator) then
6              operator_steps ← get_operator_steps(subfamily, team,
                user);
7              other_operator_steps ← get_operator_steps(subfamily,
                team, other_operator);
8              common_steps ← get_common_steps(operator_steps,
                other_operator_steps);
9              if common_steps is not empty then
10                 operator_dur ← get_total_dur(operator, subfamily,
                    team, common_steps);
11                 other_operator_dur ← get_total_dur(other_operator,
                    subfamily, team, common_steps);
12                 relative_speed ← operator_dur / other_operator_dur;
13                 update_relative_speed(operator, other_operator,
                    relative_speed);
14                 update_relative_speed(other_operator, operator, 1 /
                    relative_speed);
```

actions. This process involves collecting all known step types (shown in Table 3.9) within similar subfamilies and determining the action size by randomly selecting a similar action size. Following the sequence order of these procedures, each step in the crossover action is assigned randomly.

Once the duration for all tuples is determined, whether for known or unknown subfamilies, CROSS identifies the fastest operator to close the ticket based on its pending ticket queue. Similar to SNOOKER, CROSS also employs an MLFQ approach to update the priority of tickets in each operator's queue, ensuring they do not wait indefinitely. A ticket's priority is updated if the total time spent in the corresponding queue exceeds the average time spent by all other tickets in the same priority queue. While a particular operator may resolve a ticket more quickly than others, they may be occupied with another ticket. Therefore, the decision-making process balances the operator's availability with their possible actions before finalizing a recommendation.

The ticket treatment may be initiated immediately or delayed in both scenarios, depending on the operator's pending ticket queue status. Furthermore, if the assigned operator belongs to a distinct shift, the ticket is scheduled accordingly to resolve it quickly.

Although the goal of CROSS is to recommend the fastest operator-action pair for ticket

resolution, this approach may raise concerns related to responsible AI. Prioritizing speed alone can lead to various issues, including unfair workload distribution and a reduction in the quality and appropriateness of recommendations. Unfair workload distribution where certain operators are favored is unlikely in a reasonable-volume scenarios with many open tickets and with no operators free, but they can be more prominent when the number of tickets is low and there is at least one operator available. As for the recommendations generated by CROSS, the quickest procedure may not always be the most effective or ethical (addressed in sections 6.3 and 6.1).

### 4.3.5 Recommendation Adaptation

In real-world environments, operators may not always follow automated recommendations, particularly when the suggestions deviate from their expected procedures. To address this behaviour during the ticket handling simulation, CROSS allows the recommended operator to accept or refuse the suggested procedure. This decision is influenced by their RecOmmendation Acceptance Rate (ROAR), a configurable feature that simulates the operator's likelihood of accepting the suggested procedure when no real-world operators are available. ROAR only applies when the recommended action differs from the one the operator would employ without CROSS (the suggested procedure is automatically accepted if it matches their intended choice). This parameter is an additional property specified by the user in the external configuration file, discussed in section 3.7.

When a recommendation is refused, CROSS activates an adaptive hint generator that recommends historical actions that partially match the initial hints. With each rejection, the hints are refined and expanded, gradually increasing the number of steps. The size of these hints begins at one, representing the number of consecutive steps the SOC operator would perform without CROSS. If no past actions meet the hint criteria, CROSS stops suggesting actions, and the ticket is treated with the operator's default (emulated) action.

In this process, the system begins by querying historical records from the CROSS knowledge base using the initial hint size and selecting the fastest action. If the suggested action is refused again by the operator or no suitable action is found, the hint size is incremented by one. CROSS then repeats the search using the updated hint size, recommending the quickest available solution. This iterative operation continues as long as the operator keeps rejecting the recommendations and the hint criteria do not match any known past action (a match would result in acceptance). The hints provided are cross-referenced with past actions from similar incident types, allowing a diverse and adaptable treatment. For example, consider the following scenario:

- An operator X has a ROAR of 20%;
- CROSS suggests the sequence ['Y', 'p4', 'b3', '7a', 'e0', 'j'];
- The operator intends to use the sequence ['Z', 'f3', 'e0', 'b3', '7a', 'j'] in the simulated case;
- Historical records include the following sequences ['Z','e2','b3','7a','e0','j'], ['Z','f3','b3', 'e0','7a','j'], ['Z','f3','1d','b3','7a','j'];

In this scenario, CROSS recommends the action sequence ['Y', 'p4', 'b3', '7a', 'e0', 'j'] to resolve a ticket. If the SOC operator refuses the suggestion, starting with step 'Z', CROSS searches

for procedures that begin with step 'Z'. CROSS then estimates the duration of each available option and suggests the fastest one (for example, ['Z','e2','b3','7a','e0','j']). If the operator declines again (by executing action 'f3'), CROSS narrows the search to actions starting with ['Z','f3'], selecting from the remaining two options. This iterative process continues until it is possible to adapt the suggestion using the subfamily action data and the actions from similar subfamilies. In rare cases, if no historical actions are available or do not meet the criteria, CROSS does not provide any recommendations.

### 4.3.6 Recommendation Visualization

CROSS possesses another functionality that facilitates the representation of past operator actions through tree models. This approach enhances recommendation interpretability and boosts operator confidence in the suggested methods by showing how procedural steps of the procedures transition from one to another. Although this approach can help improve helpdesk services by identifying the most frequent procedures, it may also result in very large graphs when several treatment procedures are available, potentially reducing their comprehension.

Figure 4.17 illustrates various sequences of actions undertaken by the operator assigned by CROSS to address a ticket with the subfamily K_1. In terms of representation:

- Nodes – there are five distinct node shapes representing different sequence roles. Diamond-shaped nodes denote initialization steps (step 'q'), round nodes signify intermediate operations (steps '30', 'ee', 'd9'), square nodes illustrate ticket closure ('4'), and triangular ones indicate actions culminating in a transfer operation (analyzed in section 3.2.2.1). The fifth node style uses a note shape to indicate the total duration taken by the action sequence from initialization to closure;

- Edges styles – all action sequences are illustrated with solid edges. However, dashed edges are specifically used to link closure to note nodes. Edges can have two colors: black for standard paths, and green for highlighted sequences. Green edges have different thicknesses, with thicker edges representing faster action sequences. All solid edges are labeled with the duration of transitioning between nodes (in minutes) and the historical percentage of that transition occurring.

Table 4.7: Top 5 fastest treatment procedures in the incident K_1

| **Action** | Duration (minutes) |
|---|---|
| ['q','30','4'] | 9.38 |
| ['q','30', 'ee', '4'] | 11.44 |
| ['q','ee','30','4'] | 11.44 |
| ['q','30','30','4'] | 13.15 |
| ['q','ee','d9','4'] | 14.18 |

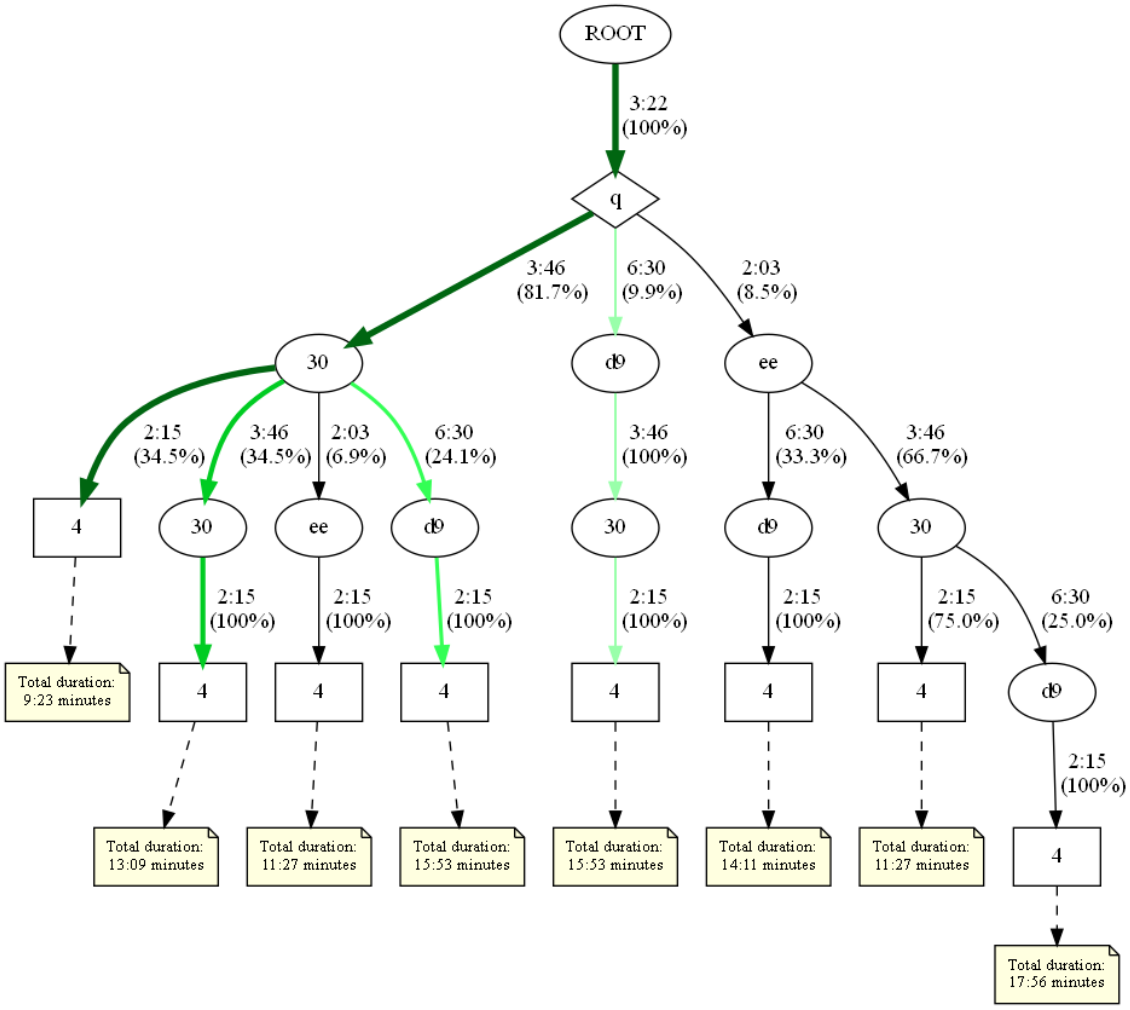



Figure 4.17: Actions analyzed in incident K_1

Table 4.7 presents the five fastest action sequences employed to treat a ticket from subfamily K_1. As observed in Fig. 4.17, the fastest actions analyzed by CROSS also correspond to those with the highest probabilities.

## 4.4 Results

The evaluation of the CROSS occurs at three levels: impact of hisTorical dAta volUme (TAU) on recommendation generation, CROSS's scalability, and a comparative analysis that uses artificial datasets with distinct ROARs (RecOmmendation Acceptance Rates) to simulate CROSS treatment.

Since the available real data lacked essential features for CROSS (client, team, and other features that help assess similarity) and no publicly available datasets exist for ticket treatment, all training and test datasets are generated using SNOOKER for every scenario. While the experiments use datasets with different sizes, they share the same features described in Tables 3.3, 3.4, 3.5, and

3.6. Additionally, a range of seeds between 2 and 6 (Appendix A.2) is employed to ensure robust results, with only the average outcomes presented.

### 4.4.1 Step Time Estimation

Before evaluating the impact of various TAU values on CROSS's performance across the mentioned metrics using a dataset of 100000 tickets over two years and a test set of 6000 tickets over ten months, the methodology for predicting the duration of a step at a given timestamp is first examined. By using a random TAU (for example, 70) to capture the most recent historical step data, we compare the performance of two prediction approaches: nonlinear least squares regression (detailed in section 4.3.4) and a simple averaging approach. The goal is to determine which method provides lower error between the expected and predicted duration. Initially, we hypothesized that there would be a TAU value at which one of the methods would outperform the other in terms of prediction accuracy. This assumption was based on the idea that the nonlinear least squares regression model might perform better when provided with sufficient data to capture trends, whereas the average approach might be more robust with limited information. However, such behaviour did not happen. This comparison shows that both approaches are insensitive to the choice of TAU, and that the selection of a specific TAU does not favor one technique over the other. Despite the small differences obtained in the results, Fig. 4.18 shows that nonlinear least squares regression always yielded shorter errors compared to those obtained in the average approach, regardless of the TAU experimented with two distinct datasets. Given the lower error rates obtained and the performance advantage of the regression method, alternative prediction strategies were not explored. Based on these findings, CROSS's prediction module was designed to utilize the nonlinear least squares regression exclusively.

CROSS utilizes distinct TAU values to identify the fastest historical actions and estimate step durations across different operators. These indicators focus on recency by retrieving the most recent $n$ actions or steps, where $n$ defines the number of past instances considered during analysis. We explore several TAU values and assess CROSS's performance using metrics, which extend those detailed in section 2.3. These include recommendation time (average and standard deviation) and readaptation time (average and standard deviation).

As the number of operators, incident types, and procedure variability increases, the recommendations provided by CROSS must be adjusted. Although the results in Fig. 4.18 indicate that a lower TAU tend to minimize prediction error in terms of step duration prediction, our primary goal at this stage is to assess whether CROSS can generate recommendations using numerous TAU indicators while ensuring the recommendation time remains within the 0.1 and 1-second range [Nielsen, 1994]. For this purpose, we evaluate if the results obtained with the following TAU values (by combining the average with standard deviation) fall within this range. These indicators range from 20 to 240, in increments of 20, with the maximum TAU representing the average number of procedures used to resolve the tickets in the training dataset.

As demonstrated in Table 4.8, CROSS shows promise in all tested TAU indicators, with even the largest value (240) potentially requiring 0.656 seconds (0.365 + 0.291), well below the defined

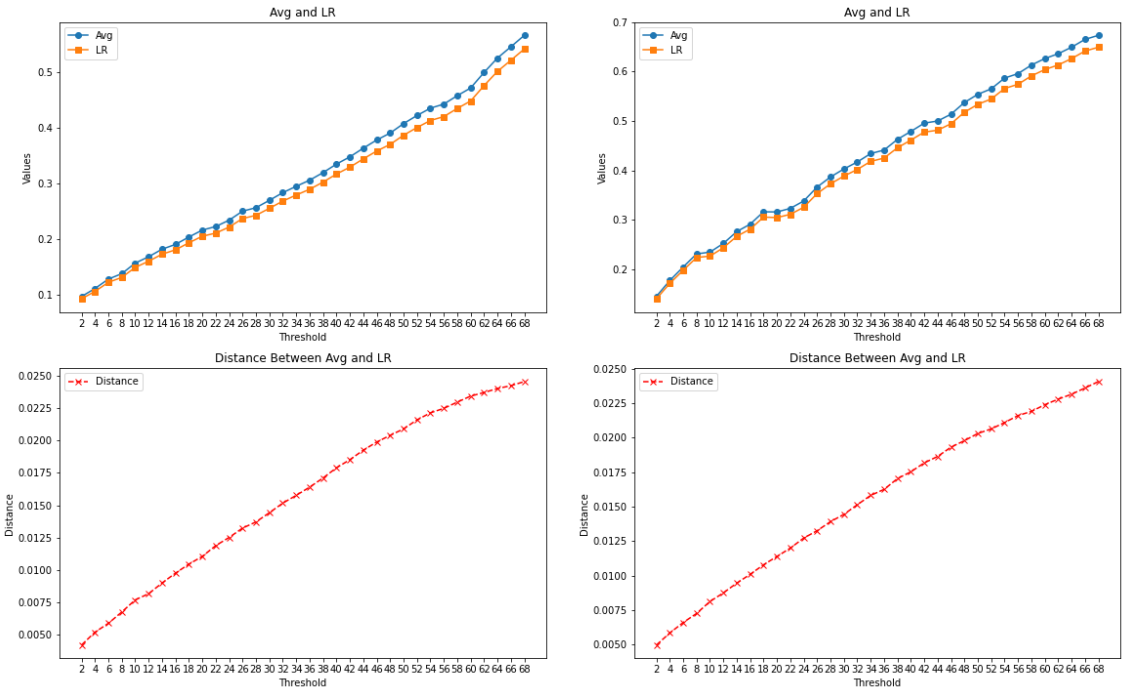



Figure 4.18: Error between the average duration expected and the average duration predicted with the two methods in two different datasets

criterion of 1 second. Additionally, the average recommendation time and its standard deviation increase with the number of procedures as more actions and, consequently, more steps are processed to determine the fastest operator-action combination. On the other hand, the average readaptation time remains constant, and its standard deviation fluctuates. Figure 4.19 illustrates the evolution of the average recommendation time with each TAU experimented.

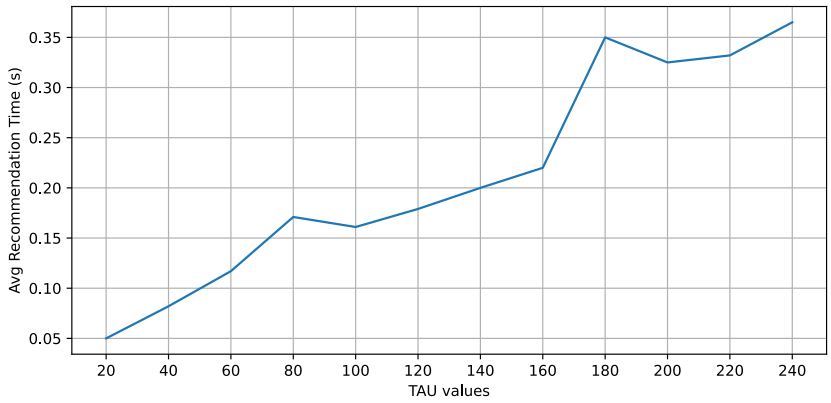



Figure 4.19: Average recommendation time with the TAU selected

As discussed earlier, the recommendation time increases as TAU grows. However, as observed in Fig. 4.19, this tendency is not always consistent. For example, the average recommendation

Table 4.8: RS Action/Step TAU Analysis

| **TAU** | AVG RT | RT SD | AVG RAT | RAT SD |
|---|---|---|---|---|
| 20 | 0.05 | 0.01 | 0.004 | 0.016 |
| 40 | 0.082 | 0.021 | 0.003 | 0.016 |
| 60 | 0.117 | 0.041 | 0.004 | 0.02 |
| 80 | 0.171 | 0.083 | 0.004 | 0.023 |
| 100 | 0.161 | 0.076 | 0.003 | 0.088 |
| 120 | 0.179 | 0.092 | 0.003 | 0.044 |
| 140 | 0.2 | 0.111 | 0.003 | 0.019 |
| 160 | 0.22 | 0.132 | 0.003 | 0.081 |
| 180 | 0.35 | 0.265 | 0.004 | 0.027 |
| 200 | 0.325 | 0.239 | 0.004 | 0.027 |
| 220 | 0.332 | 0.252 | 0.004 | 0.026 |
| 240 | 0.365 | 0.291 | 0.005 | 0.028 |

AVG - Average; RT - recommendation time; SD - standard deviation; RAT - Readaptation time.

time decreased from a TAU of 80 to 100 and from 180 to 200. We speculate that this behaviour occurs because adding more actions or steps does not always result in additional computation. When past actions contain similar steps, CROSS can utilize their predicted durations for a given timestamp, eliminating the need to repeat the prediction process. In the remaining intervals, the average recommendation time increases due to greater step variability, entailing more computation in the prediction task.

Regarding CROSS's scalability, additional datasets were generated by varying the number of operators and testing tickets to be resolved (the remaining features are the same as in the previous experiment). Table 4.9 summarizes the performance across seven scenarios and introduces a metric for peak memory usage. The TAU value 180 was chosen for the following experiments since it allows an average recommendation time of approximately half a second.

Table 4.9: Scalability Assessment

| **Scenario Selector** | AVG RT | AVG RT SD | AVG RAT | AVG RAT SD | MEM |
|---|---|---|---|---|---|
| 6000 tickets with 12 operators | 0.127 | 0.078 | 0.004 | 0.069 | 165 |
| 12000 tickets with 12 operators | 0.114 | 0.065 | 0.004 | 0.021 | 196.1 |
| 12000 tickets with 15 operators | 0.167 | 0.108 | 0.004 | 0.021 | 208.9 |
| 12000 tickets with 18 operators | 0.18 | 0.116 | 0.005 | 0.026 | 222.15 |
| 12000 tickets with 21 operators | 0.216 | 0.139 | 0.005 | 0.026 | 236.94 |
| 18000 tickets with 12 operators | 0.112 | 0.063 | 0.004 | 0.02 | 213.62 |
| 24000 tickets with 12 operators | 0.114 | 0.064 | 0.004 | 0.021 | 253.46 |

AVG - Average; RT - recommendation time; SD - standard deviation; RAT - Readaptation time; MEM - memory usage (MegaBytes). Except for the last metric, the remaining values are in seconds.

While memory usage increases with the number of testing tickets and operators, the same

does not hold for the average recommendation time and the other time-related metrics. These latter metrics fluctuate with each scenario but follow different patterns. For example, with a fixed number of operators (12), they remain stable except for the smallest testing size (6000 tickets). On the other hand, when varying the number of operators while using 12000 tickets, the average recommendation time and its standard deviation increase, whereas the average readaptation time and its standard deviation show a slight rise when shifting from 15 to 18 operators. Since more operators are involved, CROSS must estimate the same actions multiple times (proportional to the number of operators) using the methodology outlined in section 4.3.4, leading to a higher recommendation time. Figure 4.20, built with the results from Table 4.9, further corroborates this discussion by showing these features' influence on the average recommendation time and memory usage.

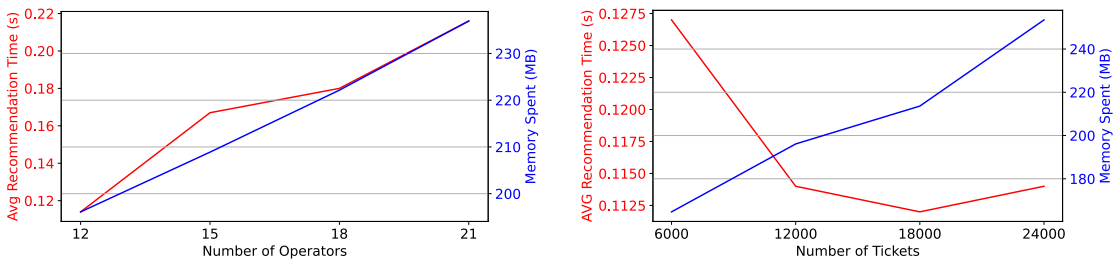



Figure 4.20: Impact of the number of operators and test tickets in the average recommendation time and memory usage

### 4.4.2 Ticket Treatment Benchmark

To assess the CROSS's impact on ticket treatment response, we perform a comparative analysis involving different ROAR values, including 100%, 80%, 60%, 40%, and 20%. The experimental process entails handling the first half of the test dataset without CROSS, using SNOOKER as the baseline emulator. The second half is then processed with CROSS enabled, applying the lowest TAU analyzed in Table 4.8. As described earlier, the training dataset contains 100000 tickets over two years, while the test dataset comprises 6000 tickets over ten months. A team of 12 operators, equally distributed across three shifts, is responsible for the ticket treatment. The remaining generation parameters are the same as the ones described in section 3.2.1.1.

This analysis evaluates various performance metrics, focusing on the following:

- Wait time for non-escalated tickets – measured by the average and standard deviation, it refers to the time from ticket creation to its allocation to an operator;
- Response time for non-escalated tickets – also computed with average and standard deviation, it represents the time an operator spends treating a ticket, from allocation to closure;
- Number of tickets accepted per refusal level – analyzes the number of tickets accepted by an operator after rejecting $n$ times.

Table 4.10: RS Impact Evaluation

| | ROAR | A | B | C | D | E |
|---|---|---|---|---|---|---|
| With CROSS | 100 | 6.16 | 53.47 | 9.98 | 2.97 | 0: 2916 |
| | 80 | 6.28 | 53.02 | 10.02 | 3.01 | 0: 2888, 1: 25, 2: 2 |
| | 60 | 6.24 | 52.82 | 10.15 | 3.18 | 0: 2796, 1: 82, 2: 25, 3: 9 |
| | 40 | 6.2 | 52.62 | 10.45 | 3.56 | 0: 2619, 1: 166, 2: 72, 3: 38, 4: 3 |
| | 20 | 6.12 | 52.17 | 11.3 | 4.36 | 0: 2207, 1: 296, 2: 214, 3: 139, 4: 20 |
| Without CROSS | — | 16.61 | 109.3 | 18.19 | 6.15 | — |

A - Average wait time of non-escalated tickets (seconds); B - Standard deviation wait time of non-escalated tickets; C - Average response time of non-escalated tickets (minutes); D - Standard deviation response time of non-escalated tickets; E - Accepted tickets per readaptation level.

Table 4.10 demonstrates that integrating CROSS contributed to a faster ticket treatment, reducing the waiting and response time compared to its absence. Moreover, lower ROAR values degrade the response time since the operators are more likely to reject recommendations containing the fastest response. On the other hand, lower ROAR values did not impact the wait time, as the results achieved are similar. Regarding readapted tickets, they are treated after $n$ acceptance considerations by the operator. For example, with a ROAR of 40, 2619 tickets were resolved immediately, while 166 were handled after one refusal, 72 after two, 38 after three, and 4 after four refusals.

CROSS's adaptability during ticket treatment is evaluated under two configurations. In the first scenario, operators always accept CROSS's suggestions (ROAR of 100). This setup measures the time difference between the actions an operator would have taken without using CROSS (emulated actions by SNOOKER) and those recommended by CROSS, assuming both sets of actions align with each other. In the second configuration, operators may reject CROSS's suggestions, using lower ROAR levels (80, 60, 40, and 20). In this case, the same time difference is calculated, but only when both sets of actions differ. Figure 4.21 shows the temporal evolution of these differences across both configurations.

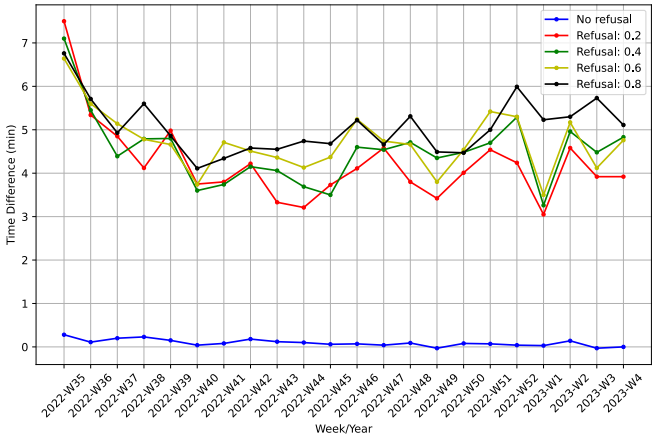



Figure 4.21: Evolution of the difference between emulated and recommended action durations

The plot illustrated in Fig. 4.21 comprises several weeks, from September 2022 to January 2023. In the scenario characterized by a ROAR of 100 or no refusal (blue line), the duration is constant throughout all the analyzed weeks, showing a stabilization around zero caused by the minor differences between the predicted and emulated durations. This stabilization also suggests an increasing alignment between the recommended actions and the actions chosen by SOC operators. The remaining cases present a time difference ranging between 46% and 63%.

Figure 4.22 showcases the progression of the RS acceptance level over the same timeframe with the referred ROAR values.

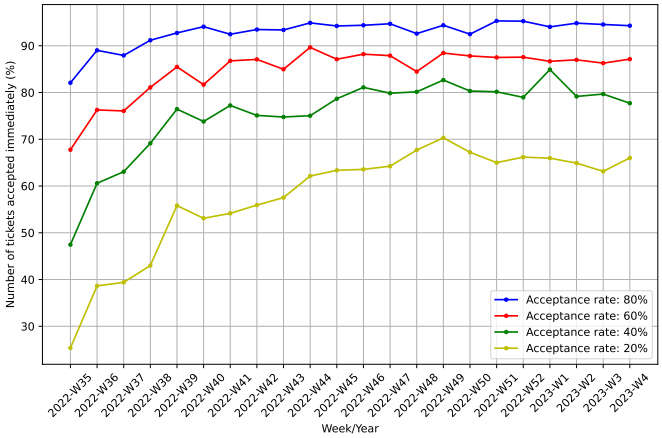



Figure 4.22: CROSS Progression

During the first four weeks, RS acceptance improves significantly, indicating that the recommended actions are more similar to those adopted by SOC operators. In the following weeks, the acceptance fluctuates, with periods registering a few drops and increases showing signs of stabilization.

## 4.5 Discussion

In contrast to the solutions presented in the related work and the considerations discussed in section 4.2.5, we have developed CROSS, an RS that performs ticket assignment and treatment through context and sequence-aware features. While it shares similarities with existing approaches, such as utilizing pre-filtering to search for the fastest SOC operator for addressing an open incident, this system also considers the quickest procedure for ticket resolution, along with visual explanations of the available steps. Additionally, it addresses the cold-start challenge by incorporating incident similarity (a concept explored several times in section 4.2.4) and action crossover (similar to how case-based reasoning tackles new problems [Lorenzi and Ricci, 2005]), enabling practical recommendations even when limited data is available. CROSS also takes into account operator feedback, adjusting its treatment procedure when they refuse the suggestion.

As the integration of CROSS into S21sec's operational systems is not currently feasible, and thus real-time operator feedback on the accuracy and utility of the recommendations cannot be captured, we carried out an internal and external evaluation of this system. The first phase performs a computational analysis assessing CROSS's scalability using different sizes of historical data, allowing us to understand its performance under different workload conditions. The second part focuses on evaluating CROSS's overall impact on ticket treatment, comparing the results obtained with and without this system. Furthermore, the influence of ROAR in CROSS is examined in terms of the temporal evolution and acceptance level. In these experiments, we employed various time-based performance metrics (average recommendation time and average wait time to treat a ticket), extending those studied in section 2.3. While these metrics do not serve as a direct substitute for the analysis of feedback, they support their potential applicability in real-life operational environments.

The results obtained from the internal evaluation directly address **RQ4** - "What is the influence of historical data and different team sizes on the computational performance of an RS and its scalability?", showing that both TAU indicators applied to historical data and the composition of teams significantly affect CROSS's performance. The analysis of CROSS with several TAU values highlights its proficiency in recommending operator-procedure pairs while adhering to the 1-second criterion. Regarding scalability, the experiments reveal that the number of operators has a greater impact than the number of testing tickets in terms of average recommendation time and memory usage. Chapter 6 explores potential improvements for CROSS's scalability, such as optimizing memory usage.

The external evaluation of CROSS provides insights related to **RQ5** and **RQ6**. In **RQ5** - "How does the operator's acceptance level impact the performance of a context-aware RS and its acceptance over time?", the results show that independently of the ROAR utilized, the RS shows signs of improvement in the initial ticket treatment. With a ROAR of 100%, the duration difference decreases and stabilizes near 0. In contrast, with lower ROARs, the time difference between the actions fluctuates, with the ROAR of 20% showing the most significant variations. As for **RQ6** - "How does the introduction of a context-aware RS impact the wait time and response time to tickets?", our findings confirm that implementing CROSS improves the response times compared to scenarios without it. For example, in the test performed, the recommendations provided by CROSS decreased the average treatment time by approximately 37.9% to 45.1% and the average wait time by 62.2% to 63.2% (with a ROAR of 100), allowing resources to be redirected to other tickets. About the influence of ROAR in this analysis, the average wait time remained relatively stable while the response time decreased by approximately 0.4% to 11.7%.

It is important to note that these results were obtained using synthetic data as input for CROSS, which is free from missing values. While this enables a controlled evaluation of the system, it does not account for challenges commonly found in real-world data, such as missing or incomplete information, which could negatively impact CROSS's performance.

# Chapter 5

# Workforce Management with AutoML

Workforce Management (WFM) comprises a set of practices that organizations employ to optimize personnel efficiency, ensure compliance, and enhance productivity [Hota and Ghosh, 2013]. Their implementation contributes to numerous areas, including time management and scheduling, forecasting, budgeting, payroll, data reporting, analytics, compliance, and risk mitigation, as well as recruitment and applicant tracking [Hennigan and Bottorff, 2024]. However, the dynamic market environment and rapid technological advancements should motivate businesses to improve their workforce standards to stay competitive [Senyucel, 2009].

DSSs play a vital role in workforce management by enabling users to make well-informed decisions. Since the early 1970s, the concept of DSSs has evolved significantly. Initially, Gorry and Scott-Morton suggested that information systems supporting semi-structured and unstructured decisions should be called DSSs [Gorry and Scott Morton, 1971]. As computational power and data complexity have improved, other definitions have emerged. For example, Daniel Power describes DSSs as "interactive computer-based systems that help people use computer communications, data, documents, knowledge, and models to solve problems and make decisions" [Power, 2002].

Another technology that has gathered significant interest and could enhance DSSs in workforce management is AutoML. This concept is distinguished by its automation of multiple stages within the ML pipeline, streamlining resource-intensive and time-consuming processes associated with generating ML models [Hutter et al., 2019]. Moreover, this tool facilitates accessibility to ML techniques by enabling non-experts to harness these techniques without necessitating advanced domain knowledge.

This chapter focuses on the exploration of AutoML within WFM. Based on AutoML capabilities to forecast future events within certain periods, we present a case study on CRYSTAL (deCision suppoRt sYStem wiTh AutomL), a DSS focused on proposing team-based decisions, including hiring or dismissals.

## 5.1 Decision Support Systems

A DSS is an information system designed to support various business activities [Power, 2008]. These systems assist decision-makers in compiling, combining, and analyzing relevant information from multiple sources, such as documents, business models, and other resources, providing three types of decision-making: structured, semi-structured, and unstructured [Laudon and Laudon, 2013]. Structured decisions are repetitive decisions where decision-makers follow well-defined procedures and rules to handle them. In contrast, unstructured decisions require the decision-maker to rely on personal judgment, experience, and intuition, as the problem is not clearly defined. Semi-structured decisions fall between these two types. DSSs have access to information about the organization, thereby enhancing the generation of more dependable choices. According to the Govindam Business School, a DSS can assume several roles in the business activities, such as [Bhatt and Zaveri, 2002]:

- What-If analysis – involves assessing multiple variables to evaluate how certain features may impact the system and its overall performance. This capability allows the adoption of a proactive stance in decision-making, enabling the creation of competitive advantages and a proactive response against emerging problems, such as cyberattacks [Mora et al., 2002];
- Goal-oriented - studies the input values required to attain specific goals, often entailing numerous test cases and user case studies;
- Risk analysis – assesses and mitigates the risks affecting business plans. Risks are classified as low, medium, or high based on their impact. Intelligent risk analysis capabilities are recommended to predict future risks and proactively address them;
- Model building – aids decision-makers in identifying the most suitable models to address issues related to business activities. This involves considering input variables, the relationship between the variables, and existing constraints;
- Graphical analysis – supports managers in comprehending and visualizing vast amounts of data, allowing data summarization, trend detection, and activity forecasting.

It is essential to distinguish this concept from management information systems [Kroenke and Boyle, 2015]. The latter concept focuses on operational efficiency and day-to-day decision-making, employs structured decisions, and processes internal data. On the other hand, a DSS handles decision-making in more complex and unstructured scenarios, operates with unstructured and semi-structured decisions, and works with internal and environmental data [Asemi et al., 2011].

The integration of these tools offers several advantages to users and companies, enhancing decision-making processes by enabling faster decisions [Juneja, 2015]. Designed to streamline processes and optimize decision quality, DSSs save time that can be redirected toward other tasks. For example, in cybersecurity, it allows enhanced cyberattack diagnosis and prediction [Polatidis et al., 2017, Quiñones-Grueiro et al., 2019], while in healthcare, it assists professionals in gathering second opinions on optimal procedures and automates operations [Sutton et al., 2020]. In business management, they also enhance daily operations, speed decision-making, identify trends, and

allocate resources efficiently [Wang et al., 2008]. Furthermore, DSSs improve decision quality by presenting comprehensive and accurate information from multiple sources, studying distinct scenarios and relevant factors to reduce potential errors, such as enhancing diagnosis accuracy in healthcare [Castaneda et al., 2015]. Additionally, DSSs improve communication and collaboration among decision-makers by providing channels for sharing facts and assumptions in model-driven scenarios and enabling centralized communication in data-driven cases through large databases, promoting fact-based decision-making. Overall, DSSs deliver a wide range of benefits across various fields, including agriculture [Zhai et al., 2020], forest management [ForestDSS, 2013], and the intelligence industry [Li et al., 2021].

Conversely, using DSSs presents challenges that hinder their application in various domains [CFI Team, 2015, Mitchell and Ploem, 2018, Juneja, 2024]. First, their efficiency relies heavily on the quality and quantity of data. The presence of incomplete or outdated data can result in inaccurate decisions. Additionally, the lack of explainability in some DSSs can undermine user trust and decision-making efficacy. In critical areas like cybersecurity, the black-box nature of specific models used to train DSSs may create resistance to their adoption. Furthermore, if not appropriately managed, DSSs can lead to information overload. These systems often aggregate and analyze vast amounts of data, potentially overwhelming users with excessive information, resulting in frustration, anxiety, and reduced decision quality [Kashada et al., 2020]. Finally, deploying DSSs raises legal and security concerns [Power, 2013, Mitchell and Ploem, 2018].

The research community presented solutions for using DSSs in various domains, including workforce management and cybersecurity defense, among others. Workforce scheduling and planning aspects were excluded from the scope of this analysis.

In 2013, Sencer and Ozel proposed a real-time simulation-based DSS for workforce management in call centers [Sencer and Ozel, 2013]. This approach empowers decision-makers to assess the impact of various parameters on call center performance, including call arrival rates, agent allocation plans, and average handle times, through What-If cases. This system comprises a data repository for storing the input and model output, a simulation model that generates performance metrics such as service level, average response time, and other properties based on the agent allocation plans, call arrival rates, and other constraints. The analysis of these metrics also determines whether more analysts should be hired or dismissed. A graphical user interface ensures the platform's usability, and the presentation of the simulated output is also described. The system's performance is evaluated in a Turkish call center to identify an agent allocation plan capable of achieving an 80% service level across all call types (banking operations, fund inquiries, and other services). Upon conducting 25 simulated scenarios, the proposed methodology recommended a 26% increase in operators (from 266 to 355) and an update of the agent allocation plan to better align staffing levels with the dynamic fluctuations in call demand throughout the day.

In 2019, Llort et al. designed a DSS employing a mathematical optimization model geared towards long-term capacity planning within consulting companies [Llort et al., 2019]. This system considers the firm's objectives, turnover dynamics, and learning periods when hiring, dismissing, or promoting consultants. Furthermore, it evaluates the influence of the strategies and policies

on the team composition, performance, and associated costs. The efficiency of this system is inspected through two scenarios: a case study involving an international consultancy company and a more generalized context. In the first scenario, the DSS is assessed across different horizons (3 and 8 years), demand cases (pessimist, optimistic, and very optimistic), and promotion policies (automatic and restricted). Meanwhile, the second experiment explored broader configurations, including diverse staff policies, two turnover rates (reflecting job market dynamics and staff not promoted), clients, and five demand profiles (constant, increasing, decreasing, increasing for the first half of the horizon, and decreasing for the second half of the horizon), resulting in a total of 30 scenarios. The authors conducted a comprehensive analysis, revealing that the DSS consistently recommended hiring staff in response to demand fluctuations. Additionally, they observed that automatic promotion leads to oversized staff and, consequently, lower profits.

One year later, Young and Weckman implemented a Heuristic Evaluation of Artificially Replaced Teammates technique, employing a combination of ML and statistical approaches to assist decision-making managers in the National Football League [Young and Weckman, 2020]. This model incorporates professional attributes, including player age, historical performance data, and other statistics, to ascertain the optimal candidate for enhancing a team's competitive performance. The efficiency of this model is evaluated through a 2007 case study, wherein its utility across various decision categories, such as draft selection trades and financial contracts, among other decisions, is assessed using a one-sample t-test and a one-sample sign test. Despite inherent design and reproducibility limitations, the proposal demonstrated utility across six of the eight decision-making contexts examined.

## 5.2 Cybersecurity

Regarding DSSs dedicated to cybersecurity, a few approaches have been explored to help professionals adopt more efficient solutions.

In 2016, Duand and Yeh implemented task assignment decision support (TADS), a tool for IT managers to optimize task allocation for IT analysts [Duan and Yeh, 2016]. TADS comprises three key components: a decision rule knowledge base, a task-oriented decision criteria weighting model, and a multivariate decision-making model for single tasks, complemented by an optimization model for handling multiple tasks. Using the decision criteria and task assignment method, TADS effectively filters and assesses the most suitable analysts for specific tasks. The authors tested TADS within a large company with the support of each analyst's normalized weighted preference value, demonstrating that the system consistently made optimal decisions.

In 2017, Souissi et al. developed an incident response framework based on a Bayesian classifier to address diverse types of cyberattacks [Souissi et al., 2017]. Their approach employs a supervised learning model and the high level of abstraction of attack vectors to facilitate incident treatment and consolidate potential defense solutions even in novel incident types. The decision-making task encompasses three preprocessing modules (normalization, classification, and prioritization) aimed at augmenting the alert data, serving as input to the ML model, referred to as the defense matching

module. This model correlates the features of the attack vector with available defense procedures to identify the optimal response (similar goal compared to CROSS). The proposed approach was assessed using 180 structured query language (SQL) injection attacks, employing three defense mechanisms (Referencing, Patch, and Quarantine). Performance metrics, including false-positive rate, precision, recall, and others, were used for evaluation. Although the results are preliminary, they demonstrate the system's efficacy in defining appropriate responses to cyber threats.

Two years later, Snyman and Kruger implemented a web-based DSS that analyzes and manages organizational and behavioural aspects in a corporate environment [Snyman and Kruger, 2019]. Using a behavioural threshold model and threshold analysis, this method assesses the patterns of information security behaviour among organizational members. By identifying critical behaviour thresholds, it predicts potential risky behaviours related to information security limitations. This predictive capability allows managers to take proactive stances against potential issues, including training and policy changes. However, it is noteworthy that while this DSS provides helpful suggestions, it does not consider technical aspects of information security, among other drawbacks.

In 2022, Razikin and Soewito devised a decision support model designed to deliver recommendations for security systems aimed at threat mitigation [Razikin and Soewito, 2022]. This model integrates risk analysis with the ISO/IEC 27001 cybersecurity system, assisting policymakers in making informed decisions regarding implementing IT security systems. The evaluation scenario encompassed the occurrence of security attacks in the proposed system. Statistical results revealed an increase in the average assessment value of ISO/IEC 27001 compliance, rising from 36.27 to 82.37. The paired T-test yielded a p-value of 0.002138 ($< 0.05$), indicating a significant correlation between threats to information technology security systems that utilize and do not use the recommendation of the information technology security system to the ISO/IEC 27001 compliance evaluation index value. Husák et al. developed a toolset to enhance cyber situational awareness in large and heterogeneous scenarios [Husák et al., 2022]. The solution facilitates the orchestration of the OODA (Observe, Orient, Decide, and Act) cycle, allowing prompt and adequate incident response. The tool's efficacy was assessed through a user study conducted in the Masaryk University network in collaboration with the university's cybersecurity team. The evaluation included diverse goals like usability and recommendation quality. Despite the general acceptance, data quality was identified as the primary target for improvement. Zicari et al. designed a ticket-deep ensemble classification system augmented with ad-hoc explanation techniques to help operators identify misclassification errors while improving the model [Zicari et al., 2022]. This tool offers two ensemble configurations: stacking-based and mixture-of-experts, incorporating a high-level feature extractor to extract dense ticket representations. Regarding explanation capabilities, the tool provides local interpretable model-agnostic explanations and various word clouds, enabling users to understand the deep ensemble's output for specific tickets. The authors scrutinized the impact of an imbalance-aware loss function on the DL methods explored and compared the performance against baseline deep NN and other classifiers. The results demonstrated high classification accuracy and valuable explanations.

As cyber threats become increasingly sophisticated and data becomes more complex, quickly

and accurately responding is crucial for maintaining the integrity and security of information systems. DSSs excel in analyzing data, predicting potential risks, and recommending optimal paths, enhancing the cybersecurity landscape. Their ability to facilitate fast and informed decisions makes DSSs indispensable for enhancing cybersecurity strategies against evolving threats.

## 5.3 Automated Machine Learning

AutoML has gathered significant attention lately for its potential to accelerate innovation, optimize resource allocation, and derive conclusions from vast volumes of data without requiring high ML expertise [Kanti Karmaker Santu et al., 2021]. Its implementation reduces the time and effort needed to build and deploy ML models and enhances the performance of these models by exploring several algorithms, feature sets, and hyperparameters. Hanussek et al. conducted a benchmarking study comparing AutoML with ML professionals, demonstrating AutoML's superior performance across multiple ML experiments [Hanussek et al., 2021]. This domain automates and optimizes various complex ML tasks, encompassing [He et al., 2021]:

- Data Preprocessing – involves a series of operations to enhance the quality and usability of raw data for analysis. These operations include the collection, cleaning, and augmentation of the data. Typical tasks involve handling missing data and noise, encoding data, and other optimization strategies;
- Feature Engineering – transforms raw data into a format that may optimize modeling performance. This operation involves various processes, including feature construction, selection, and extraction. Feature construction is responsible for creating new features that capture relevant patterns. Feature selection identifies the most impactful features for modeling, while feature extraction extracts informative and non-redundant features from the dataset;
- Model Selection – once the data is prepared and the features are engineered, AutoML explores a diverse array of ML models, evaluating their performance and selecting the one that presents the highest accuracy;
- Hyperparameter Tuning – identifies the optimal set of hyperparameters that deliver the most efficient ML model. These parameters control the behaviour behind the models explored by AutoML and may not be learned from the data. Standard methodologies include grid and random search, Bayesian optimization, gradient-based optimization, and evolutionary approaches, among other methods;
- Model Evaluation – assesses the model's performance using a range of evaluation metrics, including precision, recall, MAE, and other options. The selection of evaluation measures depends on the problem's nature (regression, classification, time series forecasting).

Various surveys and research articles perform AutoML benchmarking to understand their advantages and limitations under different conditions. Publications focused on individual AutoML platforms, such as Auto-sklearn, H2O, and others, are deliberately excluded from this analysis to maintain a broad perspective over the methodologies.

In 2021, He et al. conducted a comprehensive overview of various components within the field of AutoML, delineating their strengths, limitations, and application scenarios [He et al., 2021]. They study each phase of the AutoML pipeline, including data preparation, feature engineering, model generation, and model validation, while reviewing methods introduced in each step. Furthermore, they evaluate multiple neural network architecture search (NAS) architectures using the CIFAR-10 and ImageNet datasets and discuss various NAS research topics, including one/two-stage, one-shot, and joint hyperparameter and model optimization techniques.

Santu et al. introduced a comprehensive seven-tier AutoML framework, structured according to the degree of automation deployed [Kanti Karmaker Santu et al., 2021]. The authors categorize AutoML research based on diverse ML tasks, encompassing data visualization, cleaning and curation, feature engineering, hyperparameter tuning, exploration of alternative models, evaluation and benchmarking, and systems implemented. Each tier of the taxonomy is discussed, with particular emphasis on the last tier, characterized by automatic prediction task formulation and automatic recommendation. Various challenges related to the design and learning goals are identified at this level. Alsharef et al. inspected the time-series forecasting challenge through ML, DL, and AutoML strategies [Alsharef et al., 2022]. The authors briefly describe these three concepts and review selected AutoML phases, such as model selection, hyperparameter optimization, and feature engineering. Furthermore, they analyze various AutoML tools, discerning their respective scopes and methodologies. The tools reviewed include H2O, Auto-sklearn, AutoGluon, TPOT, Auto-Weka, TSPO, AutoKeras, EvalML, and TransmogrifAI.

Two years later, Barbudo et al. performed an exhaustive systematic literature review about AutoML and the contributions made by the research community [Barbudo et al., 2023]. With this analysis, the authors established a taxonomy organizing the related work into three divisions: the automated phases within the knowledge discovery process, tasks for such automation, and methods to address the automation task. The results obtained from this taxonomy reveal a profound lack of research focusing on postprocessing compared to the preprocessing and data mining phases. Additionally, several proposals treat automation as a black-box problem, limiting its interpretability. Other considerations, challenges, and emerging trends are also discussed in this report. Zheng et al. investigated the application of AutoML techniques in deep recommender systems (AutoRecSys) [Zheng et al., 2023]. The authors offer an overview of deep RSs and NAS methodologies and present a taxonomy that categorizes AutoML strategies for RSs into five groups based on search space and search strategy: feature selection search, embedding dimension search, feature interaction search, model architecture search, and other components search. Furthermore, they undertake an empirical evaluation employing the most representative AutoRecSys for click-through prediction and Top-K recommendation tasks, using performance metrics such as recall, NDCGM, and AUC.

In 2020, Drozdal et al. examined the relationship between AutoML models and data scientists, focusing on trust and transparency features [Drozdal et al., 2020]. The researchers offer important considerations in this domain by comprehensively analyzing three studies. Firstly, a qualitative interview is employed to gather information and identify gaps within AutoML tools. Secondly, a controlled experiment investigates the impact of integrating transparency features in AutoML

platforms, aiming to evaluate their influence on trust. Finally, a card-sorting study is conducted to measure the relevance of different types of information related to AutoML.

One year later, Ferreira et al. conducted a benchmarking study of eight open-source AutoML systems across three scenarios: General ML, DL, and XGBoost, to find the most suitable ML technique and optimal hyperparameter configurations [Ferreira et al., 2021]. The tools considered encompass Auto-Keras, Auto-PyTorch, Auto-Sklearn, AutoGluon, H2O AutoML, Rapidminer, TPOT, and TransmogrifAI, which are tested using twelve OpenML datasets covering regression, binary and multi-class classification tasks. The performance of these systems is assessed in terms of prediction accuracy and computational cost. In the General ML case, TransmogrifAI emerged as the best solution for binary classification, while AutoGluon demonstrated superiority in multi-classification, and Rapidminer showcased its efficiency in regression tasks. For the DL scenario, H2O was identified as the optimal choice for binary and regression operations, while AutoGluon exhibited superior performance for regression tasks. In the XGBoost experiment, Rapidminer emerged as the top performer for binary and regression operations, whereas H20 excelled in multi-class classification.

In 2022, Pereira et al. assessed automated time series forecasting tools within the smart cities context, leveraging nine time series datasets of this domain [Pereira et al., 2022]. The authors benchmarked several AutoML platforms, including Pmdarima, Prophet, Ludwig, DeepAR, TFT, FEDOT, AutoTs, and Sktime, with a focus on their predictive accuracy, evaluated through NMAE and computational effort, measured in terms of processing time. To ensure robustness, the evaluation employed a robust rolling window approach. Experimental findings revealed that FEDOT exhibited superior performance, achieving the lowest forecast error (4.58%) while maintaining an acceptable computation overhead compared to the remaining alternatives. Gijsbers et al. introduced an open-source benchmarking framework designed for the systematic and reproducible evaluation of AutoML tools [Gijsbers et al., 2022]. This platform evaluates nine AutoML tools, including Auto-Gluon, Auto-Sklearn, Auto-Sklearn 2, Flaml, Gama, H2O, LightAutoML, MLJAR, NaiveAutoML, and TPOT across 71 classification and 33 regression tasks. The evaluation scenarios comprise model accuracy, inference time efficiency, and failure analysis. Performance metrics, including AUC, log loss, and RMSE, are applied for binary classification, multi-class classification, and regression. While comparative analysis reveals close performance among the assessed tools, using Bradley-Terry trees emphasizes the influence of meta-features such as dataset size, dimensionality, and other characteristics. Models exhibiting superior accuracy, exemplified by AutoGluon, often incur a substantial inference time cost. Furthermore, failure analysis indicates that dataset size emerges as the main cause for model failure within this AutoML benchmark since it raises memory, time, and scalability issues.

## 5.4 CRYSTAL: Decision Support System with AutoML

Over the years, both the research and industrial communities have explored several AutoML tools to streamline ML deployment. Popular AutoML platforms such as Auto-Sklearn [Freiburg, 2022],

H20 [H2O.ai, 2023], FEDOT [NSS, 2023], AutoGluon [Community, 2023], TPOT [Olson and Moore, 2023], Google Automl [Google, 2023], Azure AutoML[Microsoft, 2023] offer diverse features and that support various tasks within the ML pipeline (described in section 5.3). These systems also prioritize usability through user-friendly interfaces for data exploration and model interpretation.

Recent advancements in AutoML have explored the efficiency of various platforms across regression, classification, and time series tasks. Benchmark studies, detailed in section 5.3, showcased the optimal AutoML tools for these operations. Specifically, H20 and AutoGluon emerge as the top contenders for regression tasks, while FEDOT demonstrates value in time series forecasting [Pereira et al., 2022]. Guided by these findings, we decided to incorporate FEDOT into CRYSTAL to forecast the volume of tickets for the following $n$ days.

FEDOT is an open-source AutoML framework rooted in evolutionary methodologies designed to address ML operations, including regression, classification, clustering, and time series prediction problems [Nikitin et al., 2022]. With the support of graph optimization and learning by evolutionary methods (GOLEM[1]), FEDOT delineates various pipeline blocks exploring distinct algorithms to orchestrate the automated model development process. Figure 5.1 illustrates a pipeline created by FEDOT where lagged and ridge regression converge to the regression decision tree.

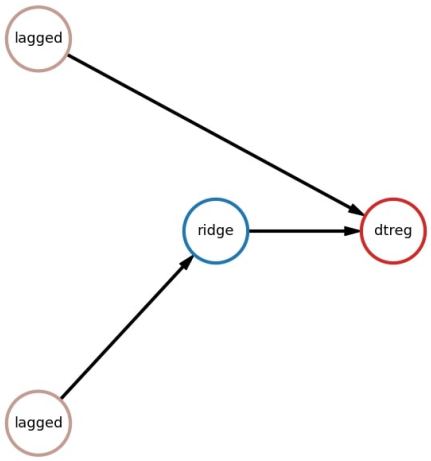



Figure 5.1: FEDOT Pipeline[2]

Distinguished by its flexibility and compatibility, FEDOT supports manual and automatic model-building approaches. In the manual setup, users have greater control, allowing them to select and customize pipeline nodes according to their specifications, including selecting desired models, preprocessing techniques, and fine-tuning the ML pipeline, among other operations. FEDOT explores and evaluates multiple pipeline settings in the automatic configuration, selecting the optimal model based on predefined metrics. In both scenarios, users are prompted to provide information about the hyperparameters, such as problem type (regression, classification, time series forecasting), forecast length (number of future steps to forecast), horizon (similar to forecast length

[1]Documentation available at https://thegolem.readthedocs.io/en/latest/

but with higher flexibility to the starting point), timeout (maximum time for model design), cv_folds (number of folds in cross-validation), n_jobs (number of parallel jobs), seed (reproducibility), and other criteria.

We explore a case study on CRYSTAL, a DSS for team size management. This tool employs AutoML to boost efficiency across various work shifts and teams by predicting future incidents and enabling proactive adjustments to staff size. Figure 5.2 illustrates CRYSTAL's architecture, spanning from the processing of the training data to the delivery of recommendations regarding the composition of SOC teams.

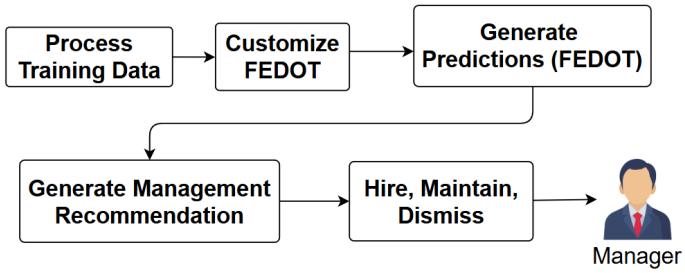



Figure 5.2: CRYSTAL Architecture

This module processes historically treated tickets and builds a dataset comprising the number of tickets for each forecasted day. Since ML requires vast data for accurate predictions, each day is subdivided into 8-hour intervals/shifts, tripling the dataset's instances. While this approach increases the data volume and granularity, it may also introduce problems, such as increased computational performance. With the augmented dataset, FEDOT executes the pipeline outlined in section 5.3 and generates predictions regarding the number of tickets in the next $n$ days.

Once the predictions are generated, CRYSTAL calculates the average time spent per ticket based on the time spent in the prior three months and estimates the time required to process the forecasted number of tickets. Given the time difference between the total shift duration (considering the number of operators) and the computed time to fix the predicted number of tickets, CRYSTAL proposes the following decision types:

- No suggestion – if it falls within the acceptable range;
- Hire $n$ analysts – if the difference is greater than the maximum acceptable range;
- Dismiss $n$ operators – if the difference is less than the minimum acceptable range.

The acceptable range depends on the decision tolerance threshold (granular feature defined by the user), which prevents rigid changes. The number of operators to be hired and dismissed is determined by Eqs. 5.1 an 5.2, respectively.

$$O_{hire} = \left\lceil \frac{(T_{shift} - T_{tickets})}{T_{shift}} \right\rceil \tag{5.1}$$

$$O_{dismiss} = max(1, O_{hire}) \tag{5.2}$$

where $O$ represents the number of operators, $T_{\text{shift}}$ denotes the total time duration of all operators within a shift, and $T_{\text{tickets}}$ indicates the time required to resolve forecasted tickets. This process iterates until the difference between $T_{\text{shift}}$ and $T_{\text{tickets}}$ falls within the acceptable range.

Using a majority voting strategy, CRYSTAL recommends a compiled decision for all shifts in each month of each run and seed. For example, if shift 0 suggests dismissing one operator in May, shift 1 recommends hiring two, and shift 2 suggests maintaining the current staff, the compiled decision would be to hire one person. At the operational level, this would involve transferring one person from shift 0 to shift 2 and hiring one additional person. Afterward, CRYSTAL aggregates the decisions across all runs and then seeds, also using majority voting. In case the decision ends in a tie, CRYSTAL prioritizes actions in the following order: maintain, hire, and dismiss. This arbitrary prioritization can be adjusted based on the business logic implemented. The number of operators is calculated as the average of the suggested values for the corresponding decision type. Different runs for each seed are used due to reproducibility issues in FEDOT, which are currently being addressed by the FEDOT development team.

The user inputs various configurations, including the forecast length, seed, and tolerance threshold. The forecast length ($n$) dictates the number of future timestamps for which predictions are generated, while seed ensures reproducibility across different runs. The threshold tolerance allows CRYSTAL to suggest flexible decisions. For example, with a tolerance level of 4 hours, the proposed decision is accepted if the remaining available time (calculated as total time minus the time required to resolve the tickets) falls within the range of [-4, 4] hours. Otherwise, the decision is re-evaluated with fewer operators hired or dismissed. Should the remaining time be outside the threshold range, the final decision is to maintain the current team composition. This threshold value can be adjusted to reflect CRYSTAL's desired strictness and align with the application scenario's specific constraints.

## 5.5 Results

This section focuses on the results obtained with CRYSTAL, after analyzing the prediction accuracy of FEDOT AutoML across diverse scenarios and its role in team size management. The following experiments employ FEDOT AutoML version 0.7.3[3]. Due to reproducibility issues currently being addressed by its development team, multiple seeds are tested, yielding results from over 10 iterations per seed.

### 5.5.1 Prediction Accuracy

The AutoML prediction accuracy is assessed through three experiments, each characterized by distinct training data distributions. These experiments evaluate the impact of S21sec distribution data and increased ticket tendency over the years on FEDOT performance with different training data sizes.

[3]Documentation available at https://github.com/aimclub/FEDOT

The goal is to evaluate FEDOT's predictive capacity in forecasting $n$ future events over the upcoming year, using training data spanning two, three, and four years (one year resulted in several pipeline issues). These training sets are derived from partitioning a baseline dataset generated by SNOOKER, comprising 150000 tickets over five years. This study analyzes ticket prediction at 8-hour intervals instead of daily, resulting in a forecast length of 1095 intervals instead of 365 (one year). Performance metrics, including MAE, RMSE, MAPE, and computational time (minutes), are employed for this evaluation. Users must also define the problem type involved in FEDOT (forecasting) and the forecast length.

Given the customizable seed parameters in SNOOKER and FEDOT, we experimented with artificial datasets using different configurations and various seeds. For enhanced comprehension and analysis, the following figures and tables showcase only the average values for each seed, omitting individual run results (details on each run per seed are provided in the Appendix A).

Figure 5.3 compares the predicted number of tickets for the last year, based on the training with two, three, and four years of data using real information, and the test dataset when using real data. As verified, the tendency lines generated from training with three and four years align more closely with the observed behaviour than the one achieved with two years of training.

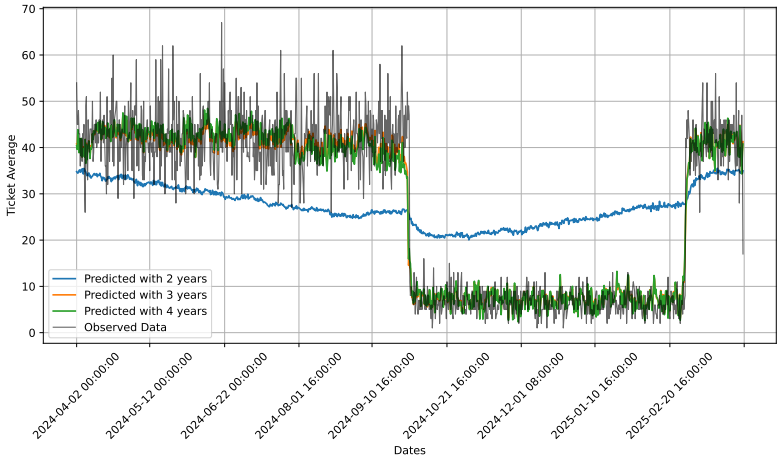



Figure 5.3: Comparison between predicted ticket volume and observed using only real data

Table 5.1 complements this analysis by presenting the performance metrics achieved by FEDOT using the same training segments and S21sec distribution data about tickets and families. Real data was included in FEDOT training to ensure that the models learned from patterns, noise, and variability present in real-life environments.

Although more data often improves prediction accuracy, the three-year dataset yielded more accurate results than both the two-year and four-year datasets. This outcome can be mainly attributed to limitations associated with using AutoML [Azevedo et al., 2024]. The two-year dataset likely underperformed due to insufficient training data, which may have limited the model's capacity to capture patterns and seasonal trends of real-world ticket scenarios. In contrast, the four-year dataset introduced an additional layer of complexity that overwhelmed the AutoML pipeline. Moreover,

Table 5.1: FEDOT with real distribution

| Seed | two years | | | | three years | | | | four years | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAE | RMSE | MAPE | TIME | MAE | RMSE | MAPE | TIME | MAE | RMSE | MAPE | TIME |
| 0 | 15.27 | 17.13 | 1.66 | **11.72** | **3.97** | **5.77** | **0.28** | 243.1 | 5.2 | 7.16 | 0.34 | 96.6 |
| 1 | 12.57 | 14.98 | 1.31 | **5.3** | **3.97** | **5.5** | 2.94 | 73.24 | 4.57 | 6.26 | **0.29** | 57.6 |
| 2 | 17.45 | 19.75 | 1.91 | **5.27** | 5.6 | 7.62 | 0.34 | 28.1 | **4.28** | **6.01** | **0.31** | 111.59 |
| 3 | 16.95 | 19.05 | 1.85 | 7.2 | **4.98** | **6.6** | **0.37** | **5.01** | **4.98** | **6.6** | **0.37** | **5.01** |
| 4 | 16.39 | 18.31 | 1.78 | **12.42** | **4.11** | **5.83** | **0.31** | 379.53 | 5.23 | 7.17 | 0.33 | 160.73 |
| 5 | 8.01 | 10.2 | 0.76 | 21.05 | **4.98** | **6.6** | **0.37** | 5.9 | **4.98** | **6.6** | **0.37** | **5.84** |
| 6 | 16.51 | 18.76 | 1.76 | 5.8 | **4.66** | **6.3** | **0.34** | **5.02** | 4.74 | 6.4 | **0.34** | **5.02** |
| 7 | 17.24 | 18.33 | 1.89 | 6.32 | **4.98** | **6.6** | **0.37** | 9.55 | **4.98** | **6.6** | **0.37** | **9.5** |
| 8 | 17.35 | 19.36 | 1.9 | **5.25** | **4.11** | **5.81** | **0.31** | 81.28 | 5.23 | 7.17 | 0.33 | 36.57 |
| 9 | 14.35 | 16.34 | 1.5 | **34.97** | 5.34 | 7.07 | 0.39 | 150.22 | **4.98** | **6.6** | **0.37** | 172.52 |
| AVG | 15.21 | 17.22 | 1.63 | **11.53** | **4.67** | **6.37** | 0.6 | 98.1 | 4.92 | 6.66 | **0.34** | 66.1 |

Time is in seconds.

memory restrictions could have reduced the performance of various processes within the AutoML pipeline. While AutoML can benefit users with limited expertise, it can encounter problems related to data, infrastructure, ML workflow, model performance, and human factors, leading to potentially inaccurate results [Azevedo et al., 2024].

Figure 5.4 showcases the predicted number of tickets based on different training sizes alongside the observed data, considering only increased ticket tendency over time.

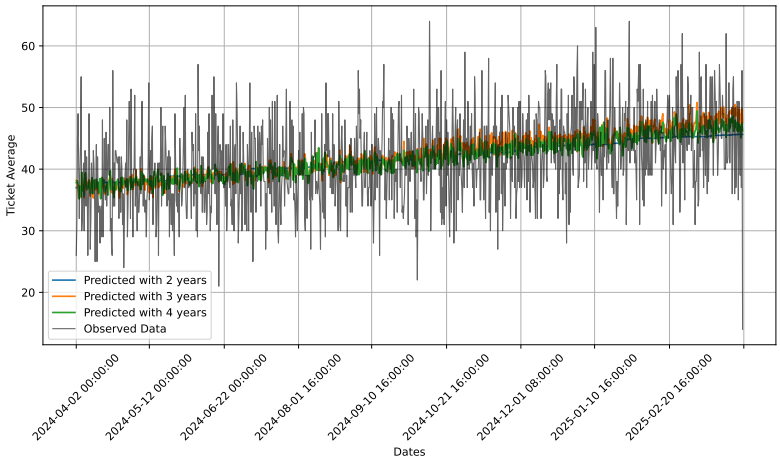



Figure 5.4: Comparison of the predicted ticket volume against the observed using only increased ticket tendency

Although the predictions derived from three and four years of training are more stable than expected, they still show an upward tendency over time. The predictions based on two years of training form a steadily increasing line, noticeable in later dates. Table 5.2 showcases the performance metrics obtained in the scenario shown in Fig. 5.4 where only increased ticket tendency is explored.

While computational time rises with the amount of training data, the error margin remains consistent. This phenomenon can be explained by the low variability in the training data, unlike the

Table 5.2: FEDOT with increased ticket tendency distribution

| Seed | two years | | | | three years | | | | four years | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAE | RMSE | MAPE | TIME | MAE | RMSE | MAPE | TIME | MAE | RMSE | MAPE | TIME |
| 0 | **5.35** | **6.71** | **0.13** | **5.31** | 5.52 | 7 | **0.13** | 7.21 | 5.56 | 7.07 | **0.13** | 25.96 |
| 1 | 5.51 | 6.86 | **0.14** | 5.01 | **5.43** | **6.81** | **0.14** | **4.93** | 5.47 | 6.83 | **0.14** | 7.45 |
| 2 | **5.32** | **6.65** | **0.13** | **5.01** | 5.54 | 6.93 | 0.14 | 8 | 5.39 | 6.72 | 0.14 | 29.19 |
| 3 | 5.33 | 6.65 | **0.13** | **5** | **5.31** | **6.61** | 0.14 | 6.23 | 5.29 | 6.62 | **0.13** | 5.02 |
| 4 | **5.33** | **6.65** | **0.13** | **5.01** | 5.57 | 6.97 | 0.15 | 267.32 | 5.44 | 6.81 | 0.14 | 816.5 |
| 5 | **5.33** | **6.65** | **0.13** | **5.01** | 5.52 | 6.98 | 0.14 | 136.26 | 5.58 | 7.05 | 0.14 | 5.06 |
| 6 | **5.43** | **6.77** | **0.14** | 5.01 | 5.54 | 6.92 | **0.14** | 6.03 | **5.43** | 6.78 | **0.14** | 77.54 |
| 7 | **5.44** | **6.78** | **0.14** | 5.01 | 5.52 | 6.9 | **0.14** | **4.96** | 5.48 | 6.85 | **0.14** | 26.78 |
| 8 | **5.38** | **6.71** | **0.14** | **5** | 5.59 | 6.93 | 0.15 | 5.01 | 5.52 | 6.86 | **0.14** | 14.92 |
| 9 | 5.59 | 7.01 | 0.15 | **5.14** | 5.54 | 6.93 | **0.14** | 168.35 | **5.39** | **6.72** | **0.14** | 387.38 |
| AVG | **5.4** | **6.74** | **0.14** | **5.05** | 5.51 | 6.9 | **0.14** | 61.43 | 5.46 | 6.83 | **0.14** | 139.58 |

Time is in seconds.

first scenario, where the anonymized real dataset exhibits high variability. Figure 5.5 illustrates a similar testing scenario, combining the previous dimensions: real distribution and increased ticket tendency.

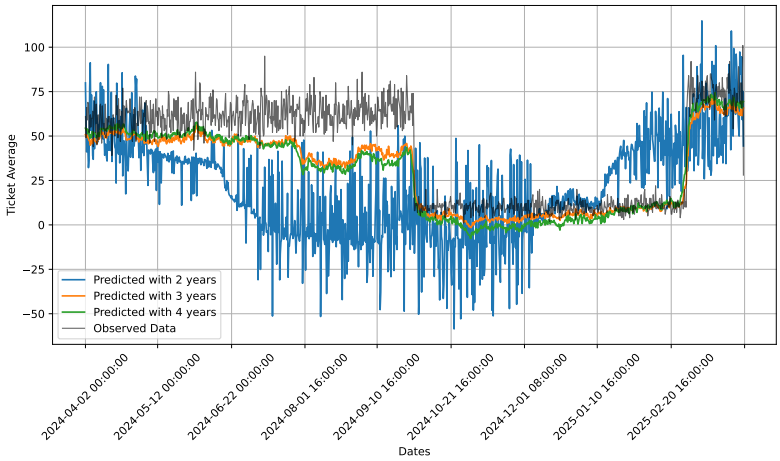



Figure 5.5: Comparison between predicted ticket volume and observed using real data combined with increased ticket tendency

The prediction lines based on three and four years of training closely align with the expected line in terms of seasonality. However, they fail to capture the increasing ticket tendency over time accurately. In contrast, the prediction line derived from two years of training presents an erratic behaviour, failing to emulate the patterns shown in the observed data. Table 5.3 displays the metrics obtained from integrating S21sec distribution data and an increased ticket tendency over the year (final scenario).

The results closely resemble those observed in the initial scenario, where metrics stabilized after three to four years, with better outcomes at three years. This similarity indicates that ticket tendency has a minimal impact on FEDOT performance compared to the actual distribution data.

Based on the experiments conducted, we identify the first scenario, which uses three years of

Table 5.3: FEDOT with increased ticket trend and real distribution

| Seed | two years | | | | three years | | | | four years | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MAE | RMSE | MAPE | TIME | MAE | RMSE | MAPE | TIME | MAE | RMSE | MAPE | TIME |
| 0 | **14.46** | 19.61 | **0.52** | **6.12** | 18.53 | 25.4 | 0.61 | 7.31 | 15.67 | **19.55** | 0.72 | 22.61 |
| 1 | 25.87 | 30.94 | 1.27 | 22.7 | **14.31** | **17.96** | **0.63** | **6.5** | 15.57 | 19.39 | 0.71 | 7.7 |
| 2 | 30.82 | 39.26 | 1.6 | **5.77** | **10.28** | **14.06** | **0.31** | 8.68 | 11.9 | 15.88 | **0.31** | 30.09 |
| 3 | 37.13 | 46 | 1.92 | **14.87** | **10.61** | **14.48** | **0.31** | 143.31 | 10.64 | 14.94 | 0.39 | 20.39 |
| 4 | 72.39 | 68.97 | 3.86 | **6.05** | **9.45** | **12.59** | **0.26** | 767.95 | 9.61 | 12.73 | 0.28 | 1722.5 |
| 5 | 30.16 | 36.11 | 1.75 | 26.53 | **10.2** | **13.54** | **0.14** | 71.98 | 15.61 | 19.5 | 0.71 | **8.58** |
| 6 | 41.12 | 53.02 | 2.51 | 8.13 | **14.32** | **18.01** | **0.63** | **6.28** | 18.13 | 23.22 | 0.86 | 13.41 |
| 7 | 32.19 | 40.47 | 1.66 | **5.73** | **13.91** | **17.42** | **0.65** | 6.28 | 15.18 | 18.82 | 0.74 | 27.91 |
| 8 | 24.07 | 29.61 | 1.12 | **10.78** | **11.25** | **14.92** | **0.31** | 15.98 | 13.21 | 17.71 | 0.33 | 16.83 |
| 9 | 146.49 | 184.88 | 7.48 | **5.68** | 13.42 | 19.02 | 0.35 | 286.47 | **10.74** | **14.05** | **0.3** | 657.03 |
| AVG | 45.47 | 54.89 | 2.37 | **11.24** | **12.63** | **16.74** | **0.42** | 132.07 | 13.63 | 17.58 | 0.54 | 252.71 |

Time is in seconds.

training data, as the most optimal configurational setup for the next test, as it achieves the lowest values across all the metrics referred to. Additionally, the fact that FEDOT could not replicate the increased ticket tendency in the final scenario further supported this decision.

### 5.5.2 Recommendations Efficiency

With the optimal distribution and training size determined, we assess CRYSTAL's performance in recommending changes to the size of SOC teams based on the forecast obtained with synthetic data. This assessment compares the decisions (such as maintaining, hiring, and dismissing staff) proposed by CRYSTAL with those that could have been employed in the testing dataset. It is important to note that this comparison does not involve input from real SOC managers, as none were available. The assessment spans four team configurations (detailed in section 3.3.2), across various seeds, and months. In these scenarios, operator distribution may or may not be balanced across the shifts. The object is to verify whether CRYSTAL's suggestions align with those that could have been proposed.

Instead of using the complete set of seeds from previous experiments (0-9), we focus on those with lower MAE and variability: 0, 1, 3, 4, 5, and 8. These seeds were selected after analyzing the full seed range to focus on more stable and accurate model outputs, essential for the next test. CRYSTAL operates on two levels: operational and compilation. At the operational level, the decisions for each month, run, and seed are obtained after inspecting the shifts detailed earlier (0, 1, and 2). At the compilation step, these decisions are aggregated across all runs and seeds for each month to determine a decision for that month.

The following figures compare the predicted decisions with those that could have been employed across all months and seeds, using a user-defined tolerance threshold of 4 hours (chosen randomly). The number of operators shown on the y-axis represents the number of operators hired (positive values) or dismissed (negative values). It is important to note that decisions are not carried over from one month to another. This choice ensures a more realistic behaviour, as a decision taken in a particular month could influence outcomes in subsequent months. For instance, the decision

to dismiss six operators in a particular month does not imply that seven operators should also be dismissed next month. Each decision is assessed independently for each month. To avoid presenting redundant results across the different team scenarios, only a selected subset of team dispositions is illustrated and analyzed next.

Figure 5.6 compares the predicted decisions with those that could have been adopted with balanced shifts across all seeds.

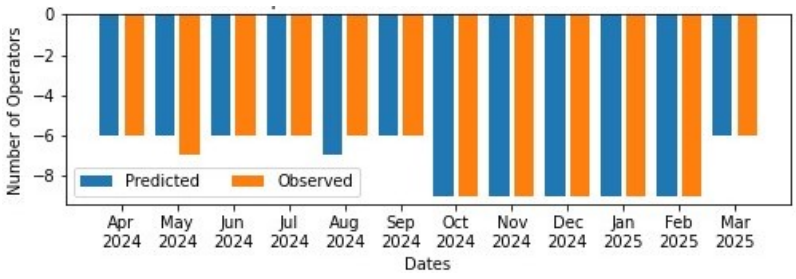



Figure 5.6: Comparison decision between predicted and those that could have been used under balanced shifts

Although there are some exceptions, such as the number of operators to be dismissed in May and August of 2024, the decision type and number of operators predicted closely align with those that could have been used. This indicates that, across all shifts, the number of SOC operators exceeds the demand based on the volume of tickets treated. Therefore, CRYSTAL recommends the dismissal of operators. During months of higher ticket seasonality (March through September), a reduction of six operators is suggested, while in the remaining months, a reduction of nine operators is proposed. The exclusive recommendation for dismissal highlights potential inefficiencies in team management. There is a clear tendency to dismiss more personnel at the beginning and end of the year, which aligns with the seasonal pattern of the tickets, with more of them occurring between March and September compared to the rest of the year.

To evaluate whether CRYSTAL could maintain the same level of performance in hiring and retention decisions, another synthetic dataset was generated with SNOOKER under conditions similar to those previously analyzed. The key difference in the new dataset is the number of operators per shift. To this end, three additional scenarios were explored:

- Scenario 13 – has three operators, one per shift;
- Scenario 14 – has four operators: shift 0 and 1 have one, and shift 1 has two;
- Scenario 15 – has six operators: shift 0 has one, shift 1 has two, and shift 2 has three;

As illustrated in Figs. 5.7, 5.8, and 5.9, CRYSTAL can recommend hiring $n$ operators and maintaining the staff level. In Fig. 5.7, it is recommended to hire operators during months with higher ticket seasonality (from March to September) and maintain the staff during the remaining months. In Fig. 5.8, the results are similar, with the difference being that CRYSTAL recommends dismissing $n$ operators during October 2024 and February 2025 instead of maintaining the staff disposition. In the final scenario, Fig. 5.9, this framework proposes maintaining the personnel during high seasonality months (except May) and dismissing operators during off-season months.

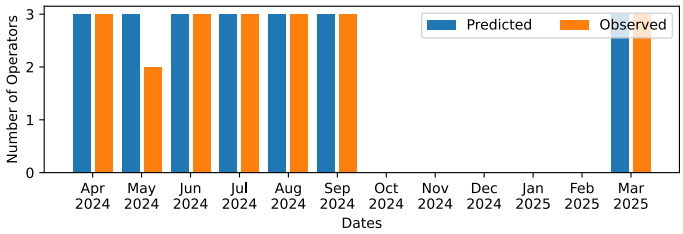



Figure 5.7: Comparison decision between predicted and those that could have been employed under scenario 13

This was the only scenario where the decision proposed by CRYSTAL deviated from the one that could have been adopted in May 2024, suggesting maintaining the number of operators instead of dismissing two.

Although a discrepancy was noted in May of 2024 in these scenarios, CRYSTAL proposed decisions in the remaining months that closely aligned with those that could have been applied.

## 5.6 Discussion

CRYSTAL is a decision support system trained with artificial data generated by SNOOKER and powered by autoML to provide insights into workforce scenarios. For its evaluation, we inspected its prediction accuracy with different distributions and training sizes and its ability in recommending staff sizing needs.

Regarding the prediction accuracy, different outcomes were obtained for the three distributions in the three training sizes. The distribution using only real data showed that the usage of more data (three and four years) improves the error metrics compared to those obtained with two years. Contrary to this behaviour, the distribution employing increased ticket tendency presented slightly better accuracy results over two years compared to the ones retrieved with three and four years. The application of both previous distributions follows the same pattern observed when only real data is used.

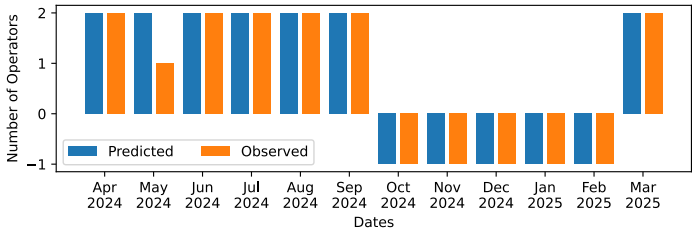



Figure 5.8: Comparison decision between predicted and those that could have been utilized under scenario 14

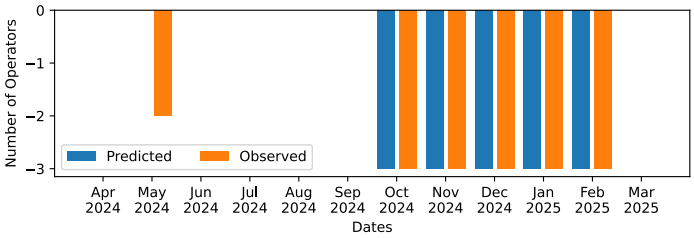



Figure 5.9: Comparison decision between predicted and those that could have been followed under scenario 15

After forecasting the ticket volume for a future timestamp, CRYSTAL potentially suggests team adjustments. In the scenarios explored previously, CRYSTAL recommended decisions closely aligned with expectations, suggesting staff dismissals Fig. 5.6 and hiring and retention Fig. 5.7. The scenarios explored in Figs. 5.8 and 5.9 further validate CRYSTAL's reliability, with consistent recommendations over several months. In some cases, the number of operators to dismiss/hire is slightly different from expectations, and in rare instances, such as in May 2024 in Fig. 5.9, the decision differs.

Although it was not possible to deploy CRYSTAL within S21sec's systems and gather feedback from real-world SOC managers, we can theorize its potential application in practical situations. By aligning with the company's objectives, such as reducing team costs or preventing delays in ticket treatment, CRYSTAL could assist in optimizing the SOC team's shift allocation. In practice, this tool could analyze historical data and forecast incident volumes over a certain period. Based on the forecasts, this system may recommend staff adjustments, potentially leading to more efficient resource management. Additionally, it can recommend different decisions tailored to several situations, which can be simulated using SNOOKER. The latter tool's flexibility enables the generation of diverse datasets that can enrich CRYSTAL's training, allowing for more comprehensive simulations of potential real-world outcomes.

# Chapter 6

# Conclusions

In today's digital landscape, cybersecurity threats' escalating complexity and frequency present crucial challenges for organizations [St. John and Swanston, 2024]. SOC teams try to detect, analyze, and mitigate cybersecurity incidents while maintaining the overall security posture; however, factors such as the overwhelming amount of incidents and the shortage of skilled professionals hinder their performance [McCann, 2023]. As reported in section 1.1, these challenges can lead to inefficient treatment and slower incident responses. One potential solution capable of automating some processes within the incident investigation cycle is the application of RSs. Considering their ability to filter information and analyze patterns in historical data, RSs could suggest the operator responsible for handling an alert, accompanied by the treatment procedure.

For this purpose, three different types of systems were developed to improve the efficiency of ticket treatment and SOC team size management. Firstly, a synthetic data generator (SNOOKER) was built to create tickets emulating real-world incidents without incurring privacy and scarcity concerns. Secondly, a context-aware RS (CROSS) was designed to discover the fastest operator-procedure pair, considering the characteristics of the tickets, improving decision-making and responses' relevance and effectiveness. Lastly, a case study on a DSS supported by FEDOT AutoML (CRYSTAL) was explored to analyze SOC team size using predictive analysis. Together, these components could address critical challenges in SOC operations, contributing to improved security posture and workflow. The SNOOKER tool is publicly available, while the remaining systems, due to contract reasons, are restricted to the public.

Regarding the research questions exposed in section 1.2, the following responses are given:

**RQ1** What approaches can build synthetic data based on real data for ticketing systems? Sections 3.1.1 and 3.1.2 detail the concept behind the available synthetic generation methods and review the research work done across multiple domains. While some researchers employ data distribution techniques due to their simplicity, others prefer more complex processes, such as GANs. In this work, we developed a synthetic generator based on data distribution designed to create tickets with various characteristics. This approach was chosen to achieve greater control over the generation of time-series data while incorporating geographical location as

an additional factor to refine the generation process. Section 3.2.2.2 discusses this decision more in-depth;

**RQ2** Can we generate synthetic data that closely mimics real data regarding distribution and business logic? As discussed in section 3.3.1, synthetic data can replicate specific characteristics or meta-features of the real data, such as the ticket and incident distribution over a year and procedure duration. In this study, SNOOKER generates incidents with distributions similar to those in the S21sec dataset. Regarding the business logic or real-world operations performed within the helpdesk domain, SNOOKER performs ticket assignment, prioritization, scheduling, and escalation (other business tasks are referred to in section 2.4). Section 3.3.2 tests SNOOKER's ticket scheduling under various conditions, with the remaining operations not being tested individually, as they are integrated within the scheduling evaluation;

**RQ3** What is the impact of different team sizes on ticket handling, measured by the percentage of delayed tickets and average wait time? Section 3.3.2 explored three different types of distribution in twelve scenarios, each with a varying number of operators and work shift distributions, to evaluate SNOOKER's ticket treatment. Results show similar behaviours when using normal and uniform distributions. However, following the assumption that SNOOKER can mimic real data patterns, the percentage of delayed tickets and wait time increase substantially due to the seasonality implemented during generation. This behaviour suggests that different configurations in ticket generation, such as seasonal patterns, can impact these metrics. Moreover, fewer operators with unbalanced work shifts aggravated this trend across all the distributions and scenarios studied. These observations show that SNOOKER's ticket handling is highly dependent on two main phases: ticket generation (number of tickets, period selected, and other temporal features) and ticket assignment, contingent on the team's dimensionality and distribution;

**RQ4** What is the influence of historical data and different team sizes on the computational performance of an RS and its scalability? Section 4.4.1 inspects CROSS's temporal efficiency using different TAU values and various team compositions. The results show that using a larger TAU to identify the fastest actions by CROSS increases the average recommendation times while readaptation times remain relatively stable. For example, as shown in Fig. 4.19, the general impression is that average recommendation time exhibits a linear growth among the TAU indicators studied. Nonetheless, as noticed in the experiment, there are cases where an increase in TAU may lead to a decrease in the average recommendation (detailed in section 4.4.1). Regarding scalability, CROSS demonstrates efficiency in terms of average recommendation time and memory usage under different SOC operators and ticket scenarios. Seven new datasets were analyzed and processed, each using a TAU of 180. The results indicate that the number of operators has a slightly higher impact than the number of testing tickets in the metrics analyzed. Both factors present a linear growth pattern in average recommendation time and memory consumption, with the number of operators having a more pronounced trend line;

**RQ5** How does the operator's acceptance level impact the performance of a context-aware RS?

Section 4.4.2 evaluates the evolution of CROSS recommendation in two scenarios, using distinct ROARs (RecOmmendation Acceptance Rates). The first case employs an ROAR of 100% and analyzes the difference between the time required to execute the emulated actions and the time as estimated by CROSS (identical sequence of actions). The other scenario performs a similar analysis with four ROARs (80%, 60%, 40%, 20%) when the same sets of actions (emulated without CROSS and recommended by CROSS) differ. In the former case, the duration difference between the two types of actions gradually decreases, approaching zero. This behaviour shows that CROSS is adaptable and that it can suggest procedures that are more aligned with those emulated. In the latter scenario, there are several fluctuations across the mentioned ROARs (time difference varies between 46% and 63%, with the most rigid ROAR (20%) being the least consistent;

**RQ6** How does the introduction of a context-aware RS impact the wait and response time to tickets? Compared to the ticket treatment without CROSS, the integration of this tool improved the average response time by 37.9% to 45.1%, helping SOC operators focus on other critical and pending tickets. Consequently, the average wait time also decreased between 62.2% and 63.2%. Regarding the impact of ROAR on these metrics, the average wait time remained unchanged, but the response time decreased between 0.4% and 11.7%. Additionally, the number of tickets accepted, either immediately or on subsequent considerations, increased, with the ROAR of 20% leading to ticket acceptance after up to four refusals.

Figure 6.1 provides a more detailed view of the proposal workflow. It illustrates the primary tasks carried out by SNOOKER, analyst emulation, ticket treatment, CROSS, and CRYSTAL.

SNOOKER, coupled with its interface, allows the generation of training and test datasets with diverse features across distinct scenarios. Such information simulates the ticket treatment through various helpdesk operations, with some of them emulating SOC operator behaviour. On the other hand, CROSS also focuses on ticket handling, but with the recommendation of the fastest analyst-procedure combination at its core. Lastly, a case study on CRYSTAL shows its potential to improve SOC team size by providing management decision support without affecting ticket treatment. The suggestions generated could help organizations save resources and prioritize attention on critical incidents, among other goals.

In addition to these tools, we contributed with four articles:

- "SNOOKER: A Dataset Generator for Helpdesk Services" article published at the Journal of Intelligent Information Systems [Ferreira et al., 2024];
- "Recommender Systems in Cybersecurity" survey published at the Journal Knowledge and Information Systems [Ferreira et al., 2023];
- "CROSS: Context-aware Recommender System" article under review;
- "CRYSTAL: Decision Support System with AutoML for Workforce Management" article under review.

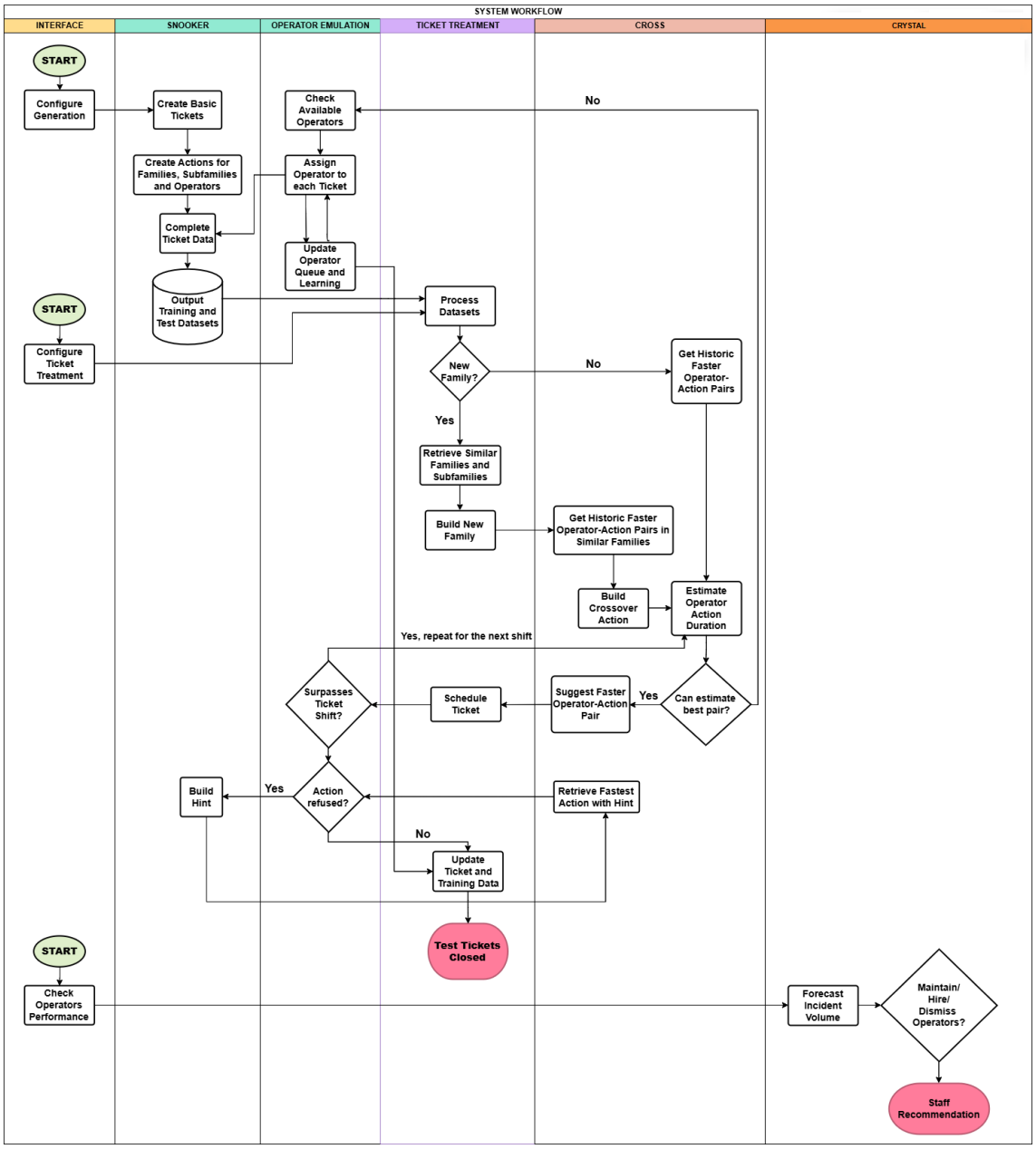



Figure 6.1: Detailed Tool Workflow

## 6.1 Limitations

While training and evaluating systems with artificial data allows for controlled experiments, this approach has limitations. For instance, although SNOOKER generates data based on real-world distribution, some aspects of real-world complexity and unpredictability may not be fully captured. This approach facilitates the testing of multiple scenarios, but transitioning to real-world deployment may benefit from additional tuning with real data to enhance generalization and robustness.

In SNOOKER, one consideration relates to the ticket generation task, which can become more time-intensive depending on user-defined constraints in the interface, including the datetime range,

ticket seasonality throughout the year, and other temporal restrictions. When these constraints are narrowly defined, such as targeting specific months or high-frequency periods during the day, the generation speed may decline. These added factors present an opportunity to explore optimization strategies to improve the speed and efficiency of this operation.

For CROSS, while it effectively prioritizes the fastest operator-procedure pair based on historical data and estimated resolution times, it does not consider other decision-making criteria. Currently, factors such as accuracy, effectiveness, and user satisfaction are not incorporated. Their integration could further improve the relevance, reliability, and effectiveness of the recommendations.

Additionally, CROSS currently uses simulated operator feedback based on ROARs to estimate the likelihood of recommendation acceptance. While this is a helpful approximation, the integration of real-time feedback from real human operators could improve the realism and adaptability of this system and help capture the complexity and variability of real operator behaviour.

CROSS demonstrated efficient performance in generating recommendations, particularly when trained with large datasets. This system relies on a sufficient volume of training data to identify the fastest operator-action combination for each ticket. The volume of training data affects the estimation of actions for various operators, the comparison of their relative speeds, and other operations. In our tests with datasets of 50.000 and 100.000 tickets with seasonality, CROSS successfully provided recommendations for all tickets. However, the availability of training data can impact the accuracy and efficiency of recommendations. As illustrated in Table 4.8, the recommendation time heavily depends on the TAU used: lower TAU values result in quicker recommendation generation, while a larger TAU may increase processing time to reach valid recommendations. Lastly, CROSS may fail to provide suggestions in the following scenarios:

- The team from the open ticket is unknown, and there is no historical data about how users treated similar past incidents (analyzed in the training set);
- The ticket does not include information about client (this case may not happen in S21sec as tickets inherently identify clients);
- A crossover action can not be estimated (applicable in novel incidents), or the fastest operator can not be identified.

While SNOOKER employs multiple imputation to handle missing data in real-world datasets, CROSS was developed using synthetic data that is intentionally complete and free from missing values. This design simplifies experimentation and ensures consistency during testing. However, for real-world deployment, CROSS would require tuning to handle incomplete or partially missing data to improve its resilience and applicability.

FEDOT, the core engine of CRYSTAL, was not developed by us, but our evaluation showed some interesting results about it. As highlighted in Table 5.1, prediction times can vary depending on the seed used, and repeated runs with the same seed do not guarantee the same output. In addition, the pipeline performance is variable when working with small training datasets and longer forecasting lengths.

The design of the proposed systems involved a trade-off between memory usage and execution speed. Given the project's priorities, speed was prioritized over memory efficiency. Dictionary structures were utilized to store our data instead of traditional databases to enhance the speed of reading and writing operations. This approach improves performance by enabling faster data access and manipulation, crucial for real-time operations. However, these advantages come at the cost of increased memory consumption. While dictionaries offer superior speed, they consume more memory than databases optimized for data storage. Section 6.3 addresses a potential improvement that could alleviate this trade-off during the integration on S21sec.

Table 6.1 illustrates other small limitations that may affect the proposed tools.

Table 6.1: Minor limitations

| Component | Limitations |
|---|---|
| SNOOKER | – Can create different operators but not new teams;<br>– The parser can only interpret the features of the real dataset if they align with those recognized by SNOOKER. Moreover, the real dataset must be properly formatted (e.g., some features might contain undefined values or patterns, such as -1);<br>– The family representation is limited to 26 options (A-Z);<br>– Can simulate only 255 different treatment steps;<br>– The probabilities of adding, removing, changing, or doing nothing regarding a simulated action are predefined; |
| CROSS | – Can only read CSV and XLSX datasets;<br>– The processing of training and testing datasets is performed only when certain required features are present; |
| CRYSTAL | – The dataset used for training and validation is only processed when certain features exist with a predefined approach;<br>– The size of the training dataset used in forecasting depends on the characteristics of the dataset employed. In our case, three years was ideal, but the same may not happen with another dataset. |

## 6.2 Integration on S21sec

Integrating these tools into S21sec, where existing frameworks and methodologies vary widely, presents opportunities and challenges.

SNOOKER could complement current testing systems by generating diverse and realistic datasets, enhancing the robustness of implemented security measures. For example, it can create datasets where certain types of incidents are more common in specific periods of the day or week and simulate the presence of tickets indicative of suspicious activities in certain countries, among other benefits. However, such integration requires the development of a parser that bridges S21sec incident data with SNOOKER, ensuring that synthetic datasets accurately emulate the desired behaviours.

Currently, each SOC team within S21sec assigns tickets to operators based solely on the potential severity of the incident. However, this approach does not use specific knowledge of the incident types during the ticket assignment process. The deployment of CROSS together with existing incident management tools could support and facilitate dynamic matching of operators based on their availability, speed, and proficiency with the fastest procedure. Such an assignment methodology could enhance workforce effectiveness and improve incident response times. Nonetheless, an important limitation is that S21sec teams do not register treatment action sequences in a granular manner for future analysis and improvement. Procedures guide the performed actions, but specific details and customizations per client/organization are recorded in a free-text format within each ticket. This approach limits the extraction of tabulated, homogenized, normalized, and comparable discrete data until the development of large language models (LLMs), currently under testing, achieves sufficient maturity. The absence of historical data impedes CROSS from making informed suggestions about handling novel and known incident types. Furthermore, in case real data is used as input to CROSS instead of synthetic data, the data processing pipeline will require additional adjustments. Real data may contain missing values and noise, requiring more extensive processing, cleaning, and normalization. An important aspect to look out for during the integration of CROSS into real-world environments is its support for real-time decision-making. SNOOKER is currently being utilized as a proxy for human operators and the ROAR feature as an emulation acceptance or refusal of recommendations. For a real-world deployment, these components are removed, and the input they provided is replaced by human operators and their actions. This allows the assessment of CROSS's effectiveness in real scenarios.

The SOC department at S21sec achieves high degrees of SLA compliance metrics in an aggregated sense, but there are limitations. On the one hand, there is a lack of curated information for analysis of individual operators' detailed performance across dimensions, such as customer, alert, severity, target, campaign, required skills, etc. On the other hand, there is an absence of tools capable of anticipating fluctuations in ticket volumes. Including CRYSTAL or an adapted version of it could assist team size management by optimizing their performance and resource allocation. However, implementing such a feature would require continuous adaptation and updating to maintain accuracy. For example, generating reliable predictions of future incident volumes would involve a continuous evaluation over different training sizes used in FEDOT to identify the configuration that produces the most accurate forecasts.

## 6.3 Future Work

In the context of future endeavors, our attention extends toward the refinement and expansion of the following aspects:

- Efficient scalability and performance – the current SNOOKER and CROSS version performs well with a moderate volume of tickets (hundreds of thousands) and a limited set of incident types and procedural actions. However, their performance can decrease as the volume of tickets, diversity of incident types, and procedural knowledge increase. Both systems rely

on in-memory data storage, prioritizing execution time over memory efficiency. Alternative storage solutions, such as distributed databases (run in the cloud), cloud-based storage services, or distributed systems, could address this issue. For example, decomposing some of the main tasks of SNOOKER and CROSS into smaller and independent services could improve performance and scalability [Tapia et al., 2020]. It would also be interesting to conduct a more complete analysis of SNOOKER's scalability and the key factors that impact its performance;

- Recommendation quality – currently, CROSS prioritizes speed over quality when looking for the fastest operator-action pair for each open ticket. This aspect could be addressed by incorporating mechanisms capable of assessing recommendations based on qualitative criteria, such as resolution accuracy, user satisfaction, among other metrics. Such a feature would help assess the reliability and relevance of the suggestions. Bobadilla et al. proposed two metrics for assessing the recommendation's quality: reliability quality prediction measure and reliability quality recommendation measure [Bobadilla et al., 2018];
- Step prediction based on analyst similarity – rather than relying solely on relative speeds to estimate a step not used by a SOC operator, this estimation could utilize the identification of similar operators. This approach would eliminate the need to analyze every operator who has used the step studied within similar subfamilies, thereby adding another layer that could reduce computational time and enhance scalability;
- AutoML for SOC evaluation – involves using the FEDOT tool to evaluate the SOC team's performance by predicting the time required to resolve tickets during each shift within a user-defined time frame. Currently, this performance is tested by comparing the average resolution time of the tickets from the last month and extrapolating it for future months. Integrating the duration of ticket resolution into the FEDOT framework could help us gain a deeper understanding of its predictive capabilities. However, such analysis could result in extended prediction time and would require a careful evaluation of the trade-off between the computational costs incurred by this approach and those of the current approach;
- Recommendations with different confidences – adding a confidence level to the suggestions regarding their suitability for addressing a ticket could improve reliability and trust, similarly to the introduction of XAI (explored in section 4.1.4). This addition could lead decision-makers to gauge the reliability and certainty of recommended procedures based on certain features, thereby facilitating more informed decision-making processes in cybersecurity operations. For instance, the confidence score of a recommendation could derived from factors such as the historical frequency of the action being accepted or rejected, whether the estimated duration of the action steps was based on information from the same operator or other, among other strategies;
- Incident identification through feature content analysis – currently, the detection of similar incidents relies on the presence of certain features. Including NLP and other analytical strategies could help analyze these features' semantic and contextual content, potentially enhancing the identification of related incidents. For example, Zhou et al. explored the

incident category and procedure data via the LDA method to improve the analysis of ticket similarity [Zhou et al., 2015]. Integrating LLMs could also enhance semantic and contextual understanding. They facilitate embedding generation for semantic comparisons, zero-shot or few-shot similarity classification, and clustering using embeddings. Xu et al. employed LLMs for zero-shot text identification and label generation in radiology reports to improve semantic analysis and develop similarity metrics for text [Xu et al., 2024]. While LLM models could also enhance the application of recommender systems in cybersecurity, it does not diminish the value of ML models. ML excels with well-structured, labeled datasets, efficiently capturing patterns using methods like regression and decision trees without heavy computational resources. Moreover, ML offers greater explainability and interpretability compared to LLMs. Additional benefits such as resource efficiency, lower data requirements, domain-specific customization, real-time performance, cost-effectiveness, and a proven track record in specialized tasks highlight ML's enduring relevance in addressing practical challenges, even as AI continues to advance;

- Simulation of cybersecurity suspicious behaviour – analyzing scenarios where certain countries exhibit a higher likelihood of delivering/receiving cyberattacks during specific temporal intervals could improve the realism in this area. This functionality could target specific days or timestamps because cyberattacks may occur more frequently during certain periods of the week [Express, 2023]. The identification of these countries could be done through the national cyber power index, which ranks the countries based on their cyber capabilities in the digital area [Voo et al., 2022];
- Expansion of SNOOKER's generation domains – currently, the proposed dataset generator is limited to generating cybersecurity tickets. However, it was designed to include additional areas like healthcare and finance. For example, SNOOKER could create patient management tickets or clinical incident reports in healthcare, while in finance, it could simulate fraud alerts. By integrating features unique to these fields, SNOOKER could emulate a broader range of scenarios, allowing comprehensive testing and analysis;
- Integration of Meta-Learning in SNOOKER – could enhance the synthetic dataset generation by enabling the creation of more adaptable and task-aware datasets through the inclusion of certain meta-features (analyzed in section 3.1.1). A related master's thesis, titled "Meta Feature classification and insertion on a Synthetic dataset generation process" started exploring this topic through a meta-feature extraction model. The study focused on identifying the most relevant meta-features and evaluating the quality of the generated synthetic data [Lemos et al., 2022];
- Granular decisions in CRYSTAL – at the moment, the decision support system, designed to assist with SOC team size management, is limited to three actions: hire, dismissal, and no changes required. While these high-level decisions are common in real-world scenarios, they do not fully capture the range of operational strategies available in SOC environments. The introduction of more granular decisions could improve CRYSTAL's utility. For example, these could be adjusting alert sensitivity, tuning rules and KPIs, cross-training operators, and

investing in temporary contract-based support, among other operations;

- Evaluation of other AutoML tools – explore alternative AutoML systems for time series forecasting. While existing literature positions FEDOT as a prominent tool in this domain, such assessments exhibit limitations, focusing on specific datasets and neglecting the exploration of different seed configurations. We aim to perform a comparative analysis across multiple AutoML platforms to corroborate FEDOT's standing as the most appropriate solution or identify alternative tools that may offer superior performance.

In addition to the future research lines outlined in this thesis, other potential contributions can be explored. For example, an in-depth analysis of how the dataset features impact the variability of FEDOT's performance metrics across diverse seeds could provide relevant results. Our experiments revealed variations in metrics, such as MAE, RMSE, and MAPE, when FEDOT employed differing seeds. A more detailed investigation of the dataset meta-features could help uncover patterns and relationships contributing to the observed variance in FEDOT's outcomes.

# Bibliography

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59. DOI: 10.3233/AIC-1994-7104.

[Abadi et al., 2016] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., and Devin, M. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv e-prints*. DOI: arXiv:1603.04467.

[Abady et al., 2020] Abady, L., Barni, M., Garzelli, A., and Tondi, B. (2020). GAN generation of synthetic multispectral satellite images. In Bruzzone, L., Bovolo, F., and Santi, E., editors, *Image and Signal Processing for Remote Sensing XXVI*, volume 11533, page 115330L. International Society for Optics and Photonics, SPIE. DOI: 10.1117/12.2575765.

[Abbas et al., 2015] Abbas, A., Zhang, L., and Khan, S. U. (2015). A survey on context-aware recommender systems based on computational intelligence techniques. *Computing*, 97(7):667–690. DOI: 10.1007/s00607-015-0448-7.

[Abbasi-Moud et al., 2021] Abbasi-Moud, Z., Vahdat-Nejad, H., and Sadri, J. (2021). Tourism recommendation system based on semantic clustering and sentiment analysis. *Expert Systems with Applications*, 167:114324. DOI: 10.1016/j.eswa.2020.114324.

[Abinaya and Devi, 2021] Abinaya, S. and Devi, M. K. K. (2021). Enhancing Top-N Recommendation Using Stacked Autoencoder in Context-Aware Recommender System. *Neural Processibg Letters*, 53(3):1865–1888. DOI: 10.1007/s11063-021-10475-0.

[Abuhussein et al., 2016] Abuhussein, A., Shiva, S., and Sheldon, F. T. (2016). CSSR: Cloud Services Security Recommender. In *2016 IEEE World Congress on Services (SERVICES)*, pages 48–55, San Francisco, CA, USA. IEEE Computer Society. DOI: 10.1109/SERVICES.2016.13.

[Adadi and Berrada, 2018] Adadi, A. and Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160. DOI: 10.1109/ACCESS.2018.2870052.

[Adomavicius et al., 2022] Adomavicius, G., Bauman, K., Tuzhilin, A., and Unger, M. (2022). Context-Aware Recommender Systems: From Foundations to Recent Developments Context-aware recommender systems. In Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender*

*Systems Handbook*, pages 211–250. Springer US, New York, NY. DOI: 10.1007/978-1-0716-2197-4_6.

[Adomavicius et al., 2011] Adomavicius, G., Mobasher, B., Ricci, F., and Tuzhilin, A. (2011). Context-Aware Recommender Systems. *AI Magazine*, 32(3):67–80. DOI: 10.1609/aimag.v32i3.2364.

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge & Data Engineering*, 17(6):734–749. DOI: 10.1109/TKDE.2005.99.

[Agarwal and Chauhan, 2017] Agarwal, A. and Chauhan, M. (2017). Similarity measures used in recommender systems: a study. *International Journal of Engineering Technology Science and Research IJETSR, ISSN*, 4(6):2394–3386.

[Aggarwal, 2016] Aggarwal, C. C. (2016). *An Introduction to Recommender Systems*, pages 1–28. Springer International Publishing, Cham. DOI: 10.1007/978-3-319-29659-3_1.

[Aghdam, 2019] Aghdam, M. H. (2019). Context-aware recommender systems using hierarchical hidden Markov model. *Physica A: Statistical Mechanics and its Applications*, 518:89–98. DOI: 10.1016/j.physa.2018.11.037.

[Agrawal and Goyal, 2013] Agrawal, S. and Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28 of *ICML'13*, pages 127–135. JMLR. DOI: 10.5555/3042817.3043073.

[Agyepong et al., 2020a] Agyepong, E., Cherdantseva, Y., Reinecke, P., and Burnap, P. (2020a). Challenges and performance metrics for security operations center analysts: a systematic review. *Journal of Cyber Security Technology*, 4(3):125–152. DOI: 10.1080/23742917.2019.1698178.

[Agyepong et al., 2020b] Agyepong, E., Cherdantseva, Y., Reinecke, P., and Burnap, P. (2020b). Towards a Framework for Measuring the Performance of a Security Operations Center Analyst. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–8. DOI: 10.1109/CyberSecurity49315.2020.9138872.

[Agyepong et al., 2023] Agyepong, E., Cherdantseva, Y., Reinecke, P., and Burnap, P. (2023). A systematic method for measuring the performance of a cyber security operations centre analyst. *Computers & Security*, 124:102959. DOI: 10.1016/j.cose.2022.102959.

[Ahmed et al., 2023] Ahmed, S., Alam, M. S., Hassan, M., Rodela, M., Ishtiak, T., Rafa, N., Mofijur, M., Ali, A. B. M. S., and Gandomi, A. (2023). Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artificial Intelligence Review*, 56(11):13521–13617. DOI: 10.1007/s10462-023-10466-8.

[Ahn, 2008] Ahn, H. J. (2008). A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51. DOI: 10.1016/j.ins.2007.07.024.

[Alamuri et al., 2014] Alamuri, M., Surampudi, B. R., and Negi, A. (2014). A survey of distance/similarity measures for categorical data. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 1907–1914. DOI: 10.1109/IJCNN.2014.6889941.

[Albuquerque et al., 2011] Albuquerque, G., Lowe, T., and Magnor, M. (2011). Synthetic Generation of High-Dimensional Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2317–2324. DOI: 10.1109/TVCG.2011.237.

[Alsharef et al., 2022] Alsharef, A., Aggarwal, K., Sonia, Kumar, M., and Mishra, A. (2022). Review of ML and AutoML solutions to forecast time-series data. *Archives of Computational Methods in Engineering*, 29(7):5297–5311. DOI: 10.1007/s11831-022-09765-0.

[Altitude Labs, 2017] Altitude Labs (2017). 5 lessons you can learn from Amazon's recommendation engine. Publisher: Altitude Labs. Available at http://altitudelabs.com/blog/amazon-product-recommendation-engine/ (accessed at June 18th 2019).

[Andrews, 1996] Andrews, G. (1996). What Is Synthetic Data? Defense Advanced Research Projects Agency (DARPA). Available at https://blogs.nvidia.com/blog/2021/06/08/what-is-synthetic-data/ (accessed at January 24th 2022).

[Arekar et al., 2015] Arekar, T., Sonar, M. R., Uke, D. N. J., and Klaus-Robert (2015). A Survey on Recommendation System. *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, 2(1):6–10.

[Ariza-Casabona et al., 2023] Ariza-Casabona, A., Salamó, M., Boratto, L., and Fenu, G. (2023). Towards Self-Explaining Sequence-Aware Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 904–911, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3604915.3608847.

[Arvanitis et al., 2022] Arvanitis, T. N., White, S., Harrison, S., Chaplin, R., and Despotou, G. (2022). A method for machine learning generation of realistic synthetic datasets for validating healthcare applications. *Health Informatics Journal*, 28(2). DOI: 10.1177/14604582221077000.

[Asemi et al., 2011] Asemi, A., Safari, A., and Zavareh, A. A. (2011). The role of management information system (MIS) and Decision support system (DSS) for manager's decision making process. *International Journal of Business and Management*, 6(7):164–173. DOI: 10.5539/ijbm.v6n7p164.

[Ashrapov, 2020] Ashrapov, I. (2020). Tabular GANs for uneven distribution. *arXiv e-prints*. DOI: arXiv.2010.00638.

[Auer et al., 2002a] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256. DOI: 10.1023/A:1013689704352.

[Auer et al., 2002b] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77. DOI: 10.1137/S009753970139837.

[Automation, 2023] Automation, S. (2023). How to evaluate incident response beyond basic security KPIs. Published: Scrut Automation. Available at https://www.scrut.io/post/incident-response (accessed at October 28th 2024).

[Aviñó et al., 2018] Aviñó, L., Ruffini, M., and Gavaldà, R. (2018). Generating Synthetic but Plausible Healthcare Record Datasets. *arXiv e-prints*. DOI: 10.48550/arXiv.1807.01514.

[Ayala et al., 2021] Ayala, C., Jimenez, K., Loza-Aguirre, E., and Andrade, R. O. (2021). A Hybrid Recommender System for Cybersecurity Based on a Rating Approach. In Daimi, K., Arabnia, H. R., Deligiannidis, L., Hwang, M.-S., and Tinetti, F. G., editors, *Advances in Security, Networks, and Internet of Things*, pages 397–409, Cham. Springer International Publishing. DOI: 10.1007/978-3-030-71017-0_28.

[Ayala-Rivera et al., 2013] Ayala-Rivera, V., Mcdonagh, P., Cerqueus, T., and Murphy, L. (2013). Synthetic Data Generation using Benerator Tool. *arXiv preprint*. DOI: arXiv.1311.3312.

[Ayala-Rivera et al., 2016] Ayala-Rivera, V., Portillo-Dominguez, A. O., Murphy, L., and Thorpe, C. (2016). COCOA: A Synthetic Data Generator for Testing Anonymization Techniques. In Domingo-Ferrer, J. and Pejić-Bach, M., editors, *Privacy in Statistical Databases*, pages 163–177, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-45381-1_13.

[Aydin et al., 2014] Aydin, B., Angryk, R. A., and Pillai, K. G. (2014). ERMO-DG: Evolving Region Moving Object Dataset Generator. In Eberle, W. and Boonthum-Denecke, C., editors, *FLAIRS Conference*. http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS14/paper/view/7876.

[Azevedo et al., 2024] Azevedo, K., Quaranta, L., Calefato, F., and Kalinowski, M. (2024). A Multivocal Literature Review on the Benefits and Limitations of Automated Machine Learning Tools. *arXiv e-prints*. DOI: arXiv:2401.11366.

[Baltrunas et al., 2011] Baltrunas, L., Ludwig, B., and Ricci, F. (2011). Matrix factorization techniques for context aware recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, page 301–304, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2043932.2043988.

[Baowaly et al., 2019] Baowaly, M. K., Liu, C.-L., and Chen, K.-T. (2019). Realistic Data Synthesis Using Enhanced Generative Adversarial Networks. In *2019 IEEE Second International*

*Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 289–292. IEEE. DOI: 10.1109/AIKE.2019.00057.

[Barbudo et al., 2023] Barbudo, R., Ventura, S., and Romero, J. R. (2023). Eight years of AutoML: categorisation, review and trends. *Knowledge Information Systems*, 65(12):5097–5149. DOI: 10.1007/s10115-023-01935-1.

[Barkan et al., 2019] Barkan, O., Koenigstein, N., Yogev, E., and Katz, O. (2019). CB2CF: a neural multiview content-to-collaborative filtering model for completely cold item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 228–236, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3298689.3347038.

[Basiri et al., 2010] Basiri, J., Shakery, A., Moshiri, B., and Hayat, M. Z. (2010). Alleviating the cold-start problem of recommender systems using a new hybrid approach. In *2010 5th International Symposium on Telecommunications*, pages 962–967. IEEE. DOI: 10.1109/ISTEL.2010.5734161.

[Beel et al., 2016] Beel, J., Gipp, B., Langer, S., and Breitinger, C. (2016). Paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17:305–338. DOI: 10.1007/s00799-015-0156-0.

[Bell et al., 2007] Bell, R. M., Koren, Y., and Volinsky, C. (2007). The BellKor Solution to the Netflix Grand Prize. Technical report, AT&T Labs.

[Bellovin et al., 2019] Bellovin, S. M., Dutta, P. K., and Reitinger, N. (2019). Privacy and synthetic datasets. *Stanford Technology Law Review, Forthcoming*, 22:1. DOI: 10.2139/ssrn.3255766.

[Ben Sassi et al., 2017] Ben Sassi, I., Mellouli, S., and Ben Yahia, S. (2017). Context-aware recommender systems in mobile environment: On the road of future research. *Information Systems*, 72(C):27–61. DOI: 10.1016/j.is.2017.09.001.

[Bergstra et al., 2011] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of machine learning research*, 13(10):281–305. DOI: 10.5555/2188385.2188395.

[Bernardi et al., 2015] Bernardi, L., Kamps, J., Kiseleva, J., and Müller, M. J. I. (2015). The Continuous Cold Start Problem in e-Commerce Recommender Systems. *arXiv preprint*. DOI: arXiv:1508.01177.

[Bharadhwaj, 2019] Bharadhwaj, H. (2019). Meta-Learning for User Cold-Start Recommendation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. DOI: 10.1109/IJCNN.2019.8852100.

[Bhatt and Zaveri, 2002] Bhatt, G. D. and Zaveri, J. (2002). The enabling role of decision support systems in organizational learning. *Decision Support Systems*, 32(3):297–309. DOI: 10.1016/S0167-9236(01)00120-8.

[Billsus and Pazzani, 2000] Billsus, D. and Pazzani, M. J. (2000). User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction*, 10(2):147–180. DOI: 10.1023/A:1026501525781.

[Blondel et al., 2008] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008. DOI: 10.1088/1742-5468/2008/10/P10008.

[Bobadilla et al., 2018] Bobadilla, J., Gutierrez, A., Ortega, F., and Zhu, B. (2018). Reliability quality measures for recommender systems. *Information Sciences*, 442:145–157.

[Bobadilla et al., 2012a] Bobadilla, J., Ortega, F., and Hernando, A. (2012a). A collaborative filtering similarity measure based on singularities. *Information Processing & Management*, 48(2):204–217. DOI: 10.1016/j.ipm.2011.03.007.

[Bobadilla et al., 2012b] Bobadilla, J., Ortega, F., Hernando, A., and Bernal, J. (2012b). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238. DOI: 10.1016/j.knosys.2011.07.021.

[Bobadilla et al., 2013] Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, 46:109–132. DOI: 10.1016/j.knosys.2013.03.012.

[Boggs et al., 2014] Boggs, N., Zhao, H., Du, S., and Stolfo, S. J. (2014). Synthetic Data Generation and Defense in Depth Measurement of Web Applications. In Stavrou, A., Bos, H., and Portokalidis, G., editors, *Research in Attacks, Intrusions and Defenses*, pages 234–254, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-11379-1_12.

[Bokde et al., 2015] Bokde, D., Girase, S., and Mukhopadhyay, D. (2015). Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey. *Procedia Computer Science. Proceedings of 4th International Conference on Advances in Computing, Communication and Control (ICAC3'15)*, 49:136–146. DOI: 10.1016/j.procs.2015.04.237.

[Bonabeau, 2002] Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287. DOI: 10.1073/pnas.082080899.

[Borah et al., 2020] Borah, P., Bhattacharyya, D., and Kalita, J. (2020). Malware Dataset Generation and Evaluation. In *2020 IEEE 4th Conference on Information Communication Technology (CICT)*, pages 1–6. DOI: 10.1109/CICT51604.2020.9312053.

[Bouraga et al., 2014] Bouraga, S., Jureta, I., Faulkner, S., and Herssens, C. (2014). Knowledge-based recommendation systems: A survey. *International Journal of Intelligent Information Technologies (IJIIT)*, 10(2):1–19. DOI: 10.4018/ijiit.2014040101.

[Braunhofer et al., 2014] Braunhofer, M., Elahi, M., and Ricci, F. (2014). Techniques for cold-starting context-aware mobile recommender systems for tourism. *Intelligenza Artificiale*, 8(2):129–143. DOI: 10.3233/IA-140069.

[Braunhofer et al., 2013] Braunhofer, M., Elahi, M., Ricci, F., and Schievenin, T. (2013). Context-Aware Points of Interest Suggestion with Dynamic Weather Data Management. In Xiang, Z. and Tussyadiah, I., editors, *Information and Communication Technologies in Tourism 2014*, pages 87–100, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-03973-2_7.

[Breese et al., 1998] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Technical report, Microsoft Research, One Microsoft Way, Redmond, WA 98052. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-98-12.pdf.

[Brisse et al., 2021] Brisse, R., Boche, S., Majorczyk, F., and Lalande, J.-F. (2021). KRAKEN: A Knowledge-Based Recommender system for Analysts, to Kick Exploration up a Notch. In *SECITC 2021 - 14th International Conference on Security for Information Technology and Communications*, volume 13195, pages 1–17, Virtual, France. DOI: 10.1007/978-3-031-17510-7_1.

[Buczak et al., 2010] Buczak, A. L., Babin, S., and Moniz, L. (2010). Data-driven approach for creating synthetic electronic medical records. *BMC medical informatics and decision making*, 10:1–28. DOI: 10.1186/1472-6947-10-59.

[Burke et al., 2010] Burke, E., Marecek, J., Parkes, A., and Rudová, H. (2010). A supernodal formulation of vertex colouring with applications in course timetabling. *Annals of Operations Research*, 179:105–130. DOI: 10.1007/s10479-010-0716-z.

[Burke, 2002] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370. DOI: 10.1023/A:1021240730564.

[Callvik and Liu, 2017] Callvik, J. and Liu, A. (2017). *Using Demographic Information to Reduce the New User Problem in Recommender Systems*. PhD thesis, KTH, School of Computer Science and Communication (CSC), SE-100 44 Stockholm Sweden. https://www.diva-portal.org/smash/get/diva2:1111527/FULLTEXT01.pdf.

[Campiolo et al., 2016] Campiolo, R., Kon, F., Batista, D., and Esposte, A. (2016). A Collaboration Model to Recommend Network Security Alerts Based on the Mixed Hybrid Approach. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 34:586–599.

[Candes and Recht, 2012] Candes, E. and Recht, B. (2012). Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119. DOI: 10.1007/s10208-009-9045-5.

[Capuano et al., 2022] Capuano, N., Fenza, G., Loia, V., and Stanzione, C. (2022). Explainable Artificial Intelligence in CyberSecurity: A Survey. *IEEE Access*, 10:93575–93600. DOI: 10.1109/ACCESS.2022.3204171.

[Casillo et al., 2023] Casillo, M., Colace, F., Conte, D., Lombardi, M., Santaniello, D., and Valentino, C. (2023). Context-aware recommender systems and cultural heritage: a survey. *Journal of Ambient Intelligence and Humanized Computing*, 14(4):3109–3127. DOI: 10.1007/s12652-021-03438-9.

[Cassetto, 2024] Cassetto, O. (2024). SOC Metrics: The Key Metrics & KPIs to Measure Your SOC Success. Published: Radiant Security. Available at https://radiantsecurity.ai/learn/soc-metrics-and-kpis/ (accessed at October 28th 2024).

[Castaneda et al., 2015] Castaneda, C., Nalley, K., Mannion, C., Bhattacharyya, P., Blake, P., Pecora, A., Goy, A., and Suh, K. S. (2015). Clinical decision support systems for improving diagnostic accuracy and achieving precision medicine. *Journal of clinical bioinformatics*, 5:1–16. DOI: 10.1186/s13336-015-0019-3.

[CFI Team, 2015] CFI Team (2015). Decision Support System (DSS). Publisher: Corporate Finance Institute (CFI). Available at https://corporatefinanceinstitute.com/resources/management/decision-support-system-dss/ (accessed at August 2nd 2024).

[Chai et al., 2022] Chai, Z., Chen, Z., Li, C., Xiao, R., Li, H., Wu, J., Chen, J., and Tang, H. (2022). User-Aware Multi-Interest Learning for Candidate Matching in Recommenders. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1326–1335, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3477495.3532073.

[Chakrabarti et al., 2004] Chakrabarti, D., Zhan, Y., and Faloutsos, C. (2004). R-MAT: A recursive model for graph mining. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 442–446. Society for Industrial and Applied Mathematics (SIAM). DOI: 10.1137/1.9781611972740.4.

[Champiri et al., 2015] Champiri, Z. D., Shahamiri, S. R., and Salim, S. S. B. (2015). A systematic review of scholar context-aware recommender systems. *Expert Systems with Applications*, 42(3):1743–1758. DOI: 10.1016/j.eswa.2014.09.017.

[Chang et al., 2020] Chang, B., Jang, G., Kim, S., and Kang, J. (2020). Learning Graph-Based Geographical Latent Representation for Point-of-Interest Recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 135–144, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3340531.3411905.

[Chang et al., 2021] Chang, J., Gao, C., Zheng, Y., Hui, Y., Niu, Y., Song, Y., Jin, D., and Li, Y. (2021). Sequential Recommendation with Graph Neural Networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 378–387, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3404835.3462968.

[Chawla et al., 2011] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2011). SMOTE: Synthetic Minority Over-sampling Technique. *arXiv e-prints*. DOI: arXiv:1106.1813.

[CHEESEMAN et al., 1988] CHEESEMAN, P., KELLY, J., SELF, M., STUTZ, J., TAYLOR, W., and FREEMAN, D. (1988). AutoClass: A Bayesian Classification System. In Laird, J., editor, *Machine Learning Proceedings 1988*, pages 54–64. Morgan Kaufmann, San Francisco (CA). DOI: 10.1016/B978-0-934613-64-4.50011-6.

[Chen et al., 2019] Chen, D., Zhang, R., Qi, J., and Yuan, B. (2019). Sequence-Aware Recommendation with Long-Term and Short-Term Attention Memory Networks. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pages 437–442. DOI: 10.1109/MDM.2019.000-6.

[Chen et al., 2018a] Chen, J., Li, K., Rong, H., Bilal, K., Yang, N., and Li, K. (2018a). A disease diagnosis and treatment recommendation system based on big data mining and cloud computing. *Information Sciences*, 435:124–149. DOI: 10.1016/j.ins.2018.01.001.

[Chen et al., 2022] Chen, J., Liu, L., Chen, R., Peng, W., and Huang, X. (2022). SecRec: A Privacy-Preserving Method for the Context-Aware Recommendation System. *IEEE Transactions on Dependable and Secure Computing*, 19(5):3168–3182. DOI: 10.1109/TDSC.2021.3085562.

[Chen and Pu, 2012] Chen, L. and Pu, P. (2012). Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1):125–150. DOI: 10.1007/s11257-011-9108-6.

[Chen et al., 2021] Chen, R. J., Lu, M. Y., Chen, T. Y., Williamson, D. F., and Mahmood, F. (2021). Synthetic data in machine learning for medicine and healthcare. *Nature Biomedical Engineering*, 5(6):493–497. DOI: 10.1038/s41551-021-00751-8.

[Chen et al., 2020] Chen, T., Yin, H., Ye, G., Huang, Z., Wang, Y., and Wang, M. (2020). Try This Instead: Personalized and Interpretable Substitute Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*,

SIGIR '20, page 891–900, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3397271.3401042.

[Chen et al., 2018b] Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. (2018b). PixelSNAIL: An improved autoregressive generative model. In Dy, J. and Krause, A., editors, *International conference on machine learning*, volume 80, pages 864–872, Stockholmsmässan, Stockholm Sweden. PMLR. DOI: arXiv.1712.09763.

[Chen et al., 2018c] Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., and Zha, H. (2018c). Sequential Recommendation with User Memory Networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 108–116, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3159652.3159668.

[Choi et al., 2020] Choi, S.-M., Jang, K., Lee, T.-D., Khreishah, A., and Noh, W. (2020). Alleviating Item-Side Cold-Start Problems in Recommender Systems Using Weak Supervision. *IEEE Access*, 8:167747–167756. DOI: 10.1109/ACCESS.2020.3019464.

[Chowdhury et al., 2009] Chowdhury, M., Thomo, A., and Wadge, W. W. (2009). Trust-Based Infinitesimals for Enhanced Collaborative Filtering. In Chawla, S., Karlapalem, K., and Pudi, V., editors, *Proceedings of the 15th International Conference on Management of Data, December 9-12, 2009, International School of Information Management*, Mysore, India. Computer Society.

[Chu et al., 2019] Chu, Q., Liu, G., Sun, H., and Zhou, C. (2019). Next News Recommendation via Knowledge-Aware Sequential Model. In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings*, volume 11856, page 221–232, Berlin, Heidelberg. Springer-Verlag. DOI: 10.1007/978-3-030-32381-3_18.

[Civin, 2018] Civin, D. (2018). Explainable AI could reduce the impact of biased algorithms. Publisher: VentureBeat (VB). Available at https://venturebeat.com/2018/05/21/explainable-ai-could-reduce-the-impact-of-biased-algorithms/ (accessed at March 24th 2019).

[Community, 2023] Community, A. (2023). *AutoGluon 0.8.0 documentation*. AutoGluon Community, https://auto.gluon.ai/stable/index.html.

[Corporation, 2018] Corporation, T. M. (2018). MITRE ATT&CK (Adversarial Tactics, Techniques and Common Knowledge). Publisher: MITRE ATT&CK. Available at https://attack.mitre.org/ (accessed at November 10th 2022).

[Costa and Dolog, 2018] Costa, F. and Dolog, P. (2018). Hybrid learning model with barzilai-borwein optimization for context-aware recommendations. In *The Thirty-First International Flairs Conference*. AAAI Publications. International Florida Artificial Intelligence Research Society Conference.

[Costa and Dolog, 2019] Costa, F. S. d. and Dolog, P. (2019). Collective embedding for neural context-aware recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 201–209, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3298689.3347028.

[Covington et al., 2016] Covington, P., Adams, J., and Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, Boston, Massachusetts, US. ACM. DOI: 10.1145/2959100.2959190.

[Creswell et al., 2018] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, 35(1):53–65. DOI: 10.1109/MSP.2017.2765202.

[Cybernews, 2023] Cybernews (2023). Ransomware landscape overview 2023. Technical report, Cybernews.

[Dandekar et al., 2018] Dandekar, A., Zen, R. A., and Bressan, S. (2018). A comparative study of synthetic dataset generation techniques. In *Database and Expert Systems Applications: 29th International Conference, DEXA 2018, Regensburg, Germany, September 3–6, 2018, Proceedings, Part II 29*, pages 387–395. Springer. DOI: 10.1007/978-3-319-98812-2_35.

[Dankar and Ibrahim, 2021] Dankar, F. K. and Ibrahim, M. (2021). Fake It Till You Make It: Guidelines for Effective Synthetic Data Generation. *Applied Sciences*, 11(5):2158. DOI: 10.3390/app11052158.

[Das et al., 2007] Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google News Personalization: Scalable Online Collaborative Filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280, Banff, Alberta, Canada. ACM. DOI: 10.1145/1242572.1242610.

[De Rainville et al., 2012] De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: a python framework for evolutionary algorithms. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, page 85–92, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2330784.2330799.

[Dean et al., 2012] Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. (2012). Large Scale Distributed Deep Networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25 of *NIPS'12*, pages 1223–1231, Lake Tahoe, Nevada. Curran Associates Inc. DOI: 2999134.2999271.

[del Carmen Rodríguez-Hernández and Ilarri, 2021] del Carmen Rodríguez-Hernández, M. and Ilarri, S. (2021). AI-based mobile context-aware recommender systems from an information

management perspective: Progress and directions. *Knowledge-Based Systems*, 215:106740. DOI: 10.1016/j.knosys.2021.106740.

[del Carmen Rodríguez-Hernández et al., 2017] del Carmen Rodríguez-Hernández, M., Ilarri, S., Hermoso, R., and Trillo-Lado, R. (2017). DataGenCARS: A generator of synthetic data for the evaluation of context-aware recommendation systems. *Pervasive and Mobile Computing*, 38:516–541. Special Issue IEEE International Conference on Pervasive Computing and Communications (PerCom) 2016, DOI: 10.1016/j.pmcj.2016.09.020.

[Delgado and Ishii, 1999] Delgado, J. and Ishii, N. (1999). Memory-based weighted-majority prediction for recommender systems. page 85.

[Dempsey et al., 2011] Dempsey, K., Johnson, L., Scholl, M., Stine, K., Clay, A., Orebaugh, A., Chawla, N., and Johnston, R. (2011). Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations. Technical report, National Institute of Standards and Technology (NIST), Gaithersburg, MD. DOI: NIST.SP.800-137.

[Deodhar and Ghosh, 2010] Deodhar, M. and Ghosh, J. (2010). SCOAL: A framework for simultaneous co-clustering and learning from complex data. *ACM Transactions Knowledge Discovery Data*, 4(3):1–31. DOI: 10.1145/1839490.1839492.

[Deshpande and Karypis, 2004] Deshpande, M. and Karypis, G. (2004). Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177. DOI: 10.1145/963770.963776.

[Dev and Mohan, 2016] Dev, A. V. and Mohan, A. (2016). Recommendation system for big data applications based on set similarity of user preferences. In *2016 International Conference on Next Generation Intelligent Systems (ICNGIS)*, pages 1–6, Kottayam, India. IEEE. DOI: 10.1109/ICNGIS.2016.7854058.

[Dilmegani, 2020] Dilmegani, C. (2020). The Ultimate Guide to Synthetic Data in 2020. Publisher: AI Multiple. Available at https://research.aimultiple.com/synthetic-data/ (accessed at November 3rd 2020).

[Donahue et al., 2018] Donahue, C., McAuley, J., and Puckette, M. (2018). Adversarial Audio Synthesis. *arXiv e-prints*. DOI: arXiv.1802.04208.

[Dong et al., 2017] Dong, L., Huang, S., Wei, F., Lapata, M., Zhou, M., and Xu, K. (2017). Learning to Generate Product Reviews from Attributes. In Lapata, M., Blunsom, P., and Koller, A., editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632, Valencia, Spain. Association for Computational Linguistics. DOI: 10.18653/v1/E17-1059.

[Dong et al., 2020] Dong, M., Yuan, F., Yao, L., Xu, X., and Zhu, L. (2020). MAMO: Memory-Augmented Meta-Optimization for Cold-start Recommendation. In *Proceedings of the 26th*

*ACM SIGKDD international conference on knowledge discovery & data mining*, page 688–697. DOI: 10.1145/3394486.3403113.

[Drechsler and Reiter, 2008] Drechsler, J. and Reiter, J. P. (2008). Accounting for intruder uncertainty due to sampling when estimating identification disclosure risks in partially synthetic data. In Domingo-Ferrer, J. and Saygın, Y., editors, *International conference on privacy in statistical databases*, volume 5262, pages 227–238. Springer. DOI: 10.1007/978-3-540-87471-3_19.

[Drozdal et al., 2020] Drozdal, J., Weisz, J., Wang, D., Dass, G., Yao, B., Zhao, C., Muller, M., Ju, L., and Su, H. (2020). Trust in AutoML: exploring information needs for establishing trust in automated machine learning systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, page 297–307, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3377325.3377501.

[Du et al., 2019a] Du, F., Plaisant, C., Spring, N., Crowley, K., and Shneiderman, B. (2019a). EventAction: A Visual Analytics Approach to Explainable Recommendation for Event Sequences. *ACM Transactions Interactive Intelligent Systems*, 9(4):1–31. DOI: 10.1145/3301402.

[Du et al., 2020] Du, X., Wang, X., He, X., Li, Z., Tang, J., and Chua, T.-S. (2020). How to Learn Item Representation for Cold-Start Multimedia Recommendation? In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 3469–3477, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3394171.3413628.

[Du et al., 2019b] Du, Z., Wang, X., Yang, H., Zhou, J., and Tang, J. (2019b). Sequential Scenario-Specific Meta Learner for Online Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2895–2904, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3292500.3330726.

[Duan and Yeh, 2016] Duan, Y. and Yeh, C.-H. (2016). TADS: A task assignment decision support system for IT services. In *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, pages 2472–2477. DOI: 10.1109/ICIEA.2016.7604008.

[Dyk and Meng, 2001] Dyk, D. A. v. and Meng, X.-L. (2001). The Art of Data Augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50. DOI: 10.1198/10618600152418584.

[El Yebdri et al., 2021] El Yebdri, Z., Benslimane, S. M., Lahfa, F., Barhamgi, M., and Benslimane, D. (2021). Context-aware recommender system using trust network. *Computing*, 103(9):1919–1937. DOI: 10.1007/s00607-020-00876-9.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211. DOI: 10.1016/0364-0213(90)90002-E.

[Engine, 2023] Engine, M. (2023). Top 7 benefits of help desk ticketing system. Publisher: Manage Engine. Available at https://www.manageengine.com/products/service-desk/help-desk-software/help-desk-benefits.html (accessed at May 13th 2024).

[Eno and Thompson, 2008] Eno, J. and Thompson, C. W. (2008). Generating Synthetic Data to Match Data Mining Patterns. *IEEE Internet Computing*, 12(3):78–82. DOI: 10.1109/MIC.2008.55.

[Esteban et al., 2020] Esteban, A., Zafra, A., and Romero, C. (2020). Helping university students to choose elective courses by using a hybrid multi-criteria recommendation system with genetic optimization. *Knowledge-Based Systems*, 194:105385. DOI: 10.1016/j.knosys.2019.105385.

[Esteban et al., 2017] Esteban, C., Hyland, S. L., and Rätsch, G. (2017). Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv e-prints*. DOI: arXiv.1706.02633.

[Express, 2023] Express, T. C. (2023). Cyber Attack Dwell Time Reduced to 8 from 10 Days and Hackers Attack on Weekends: Repor. Available at https://thecyberexpress.com/cyber-attack-dwell-time-reduced-to-8-from-10-days-and-hackers-attack-on-weekends-report/ (accessed at December 12th 2024).

[Faez et al., 2021] Faez, F., Ommi, Y., Baghshah, M. S., and Rabiee, H. R. (2021). Deep Graph Generators: A Survey. *IEEE Access*, 9:106675–106702. DOI: 10.1109/ACCESS.2021.3098417.

[Fan et al., 2021] Fan, Z., Liu, Z., Zhang, J., Xiong, Y., Zheng, L., and Yu, P. S. (2021). Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 433–442, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3459637.3482242.

[Fang et al., 2020] Fang, H., Zhang, D., Shu, Y., and Guo, G. (2020). Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *ACM Transactions Information Systems*, 39(1):1–42. DOI: 10.1145/3426723.

[Felfernig and Burke, 2008] Felfernig, A. and Burke, R. (2008). Constraint-based Recommender Systems: Technologies and Research Issues. In *Proceedings of the 10th International Conference on Electronic Commerce*, number 3 in ICEC '08, pages 1–3, Innsbruck, Austria. ACM. DOI: 10.1145/1409540.1409544.

[Felfernig et al., 2007] Felfernig, A., Isak, K., Szabo, K., and Zachar, P. (2007). The VITA Financial Services Sales Support Environment. In *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence - Volume 2*, volume 2 of *IAAI'07*, pages 1692–1699, Vancouver, British Columbia, Canada. AAAI Press. DOI: 1620113.1620117.

[Feng et al., 2021] Feng, J., Xia, Z., Feng, X., and Peng, J. (2021). RBPR: A hybrid model for the new user cold start problem in recommender systems. *Knowledge-Based Systems*, 214:106732. DOI: 10.1016/j.knosys.2020.106732.

[Feng et al., 2015] Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y. M., and Yuan, Q. (2015). Personalized ranking metric embedding for next new POI recommendation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2069–2075. AAAI Press. DOI: 10.5555/2832415.2832536.

[Ferreira et al., 2021] Ferreira, L., Pilastri, A., Martins, C. M., Pires, P. M., and Cortez, P. (2021). A comparison of AutoML tools for machine learning, deep learning and XGBoost. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE. DOI: 10.1109/IJCNN52387.2021.9534091.

[Ferreira et al., 2023] Ferreira, L., Silva, D. C., and Itzazelaia, M. U. (2023). Recommender Systems in Cybersecurity. *Knowledge and Information Systems*, 65(12):5523–5559. DOI: 10.1007/s10115-023-01906-6.

[Ferreira et al., 2024] Ferreira, L., Silva, D. C., and Uriarte-Itzazelaia, M. (2024). SNOOKER: a dataset generator for helpdesk services. *Journal of Intelligent Information Systems*, pages 1–23. DOI: 10.1007/s10844-024-00905-5.

[Filchenkov and Pendryak, 2015] Filchenkov, A. and Pendryak, A. (2015). Datasets meta-feature description for recommending feature selection algorithm. In *2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)*, pages 11–18. DOI: 10.1109/AINL-ISMW-FRUCT.2015.7382962.

[ForestDSS, 2013] ForestDSS (2013). ForestDSS Wiki. Publisher: ForestDSS - Community of Practice Forest Management Decision Support Systems. Available at http://www.forestdss.org/CoP/ (accessed at August 2nd 2019).

[Forestier et al., 2017] Forestier, G., Petitjean, F., Dau, H. A., Webb, G. I., and Keogh, E. (2017). Generating Synthetic Time Series to Augment Sparse Datasets. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 865–870. DOI: 10.1109/ICDM.2017.106.

[Forsberg and Frantti, 2023] Forsberg, J. and Frantti, T. (2023). Technical performance metrics of a security operations center. *Computers & Security*, 135(C):103529. DOI: 10.1016/j.cose.2023.103529.

[Franco et al., 2019] Franco, M., Rodrigues, B., and Stiller, B. (2019). MENTOR: The Design and Evaluation of a Protection Services Recommender System. In *2019 15th International Conference on Network and Service Management (CNSM)*, pages 1–7, Halifax, NS, Canada. DOI: 10.23919/CNSM46954.2019.9012686.

[Freiburg, 2022] Freiburg, M. L. P. (2022). *AutoSklearn 0.15.0 documentation*. Machine Learning Professorship Freiburg, https://automl.github.io/auto-sklearn/master/manual.html.

[Gadepally et al., 2016] Gadepally, V. N., Hancock, B. J., Greenfield, K. B., Campbell, J. P., Campbell, W. M., and Reuther, A. I. (2016). Recommender Systems for the Department of Defense and Intelligence Community. *Lincoln Laboratory Journal*, 22(1):74–89. https://www.ll.mit.edu/sites/default/files/publication/doc/recommender-systems-department-defense-intelligence-gadepally-108929.pdf.

[Gao et al., 2023] Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., and Li, Y. (2023). A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions Recommendation Syste,s*, 1(1). DOI: 10.1145/3568022.

[Gartner, 2023] Gartner (2023). Gartner Predicts Nearly Half of Cybersecurity Leaders Will Change Jobs by 2025. Available at https://www.gartner.com/en/newsroom/press-releases/2023-02-22-gartner-predicts-nearly-half-of-cybersecurity-leaders-will-change-jobs-by-2025 (accessed at March 28th 2024).

[Gasmi et al., 2019] Gasmi, H., Laval, J., and Bouras, A. (2019). Cold-start cybersecurity ontology population using information extraction with LSTM. In *2019 International Conference on Cyber Security for Emerging Technologies (CSET)*, pages 1–6. DOI: 10.1109/CSET.2019.8904905.

[Geng et al., 2015] Geng, X., Zhang, H., Bian, J., and Chua, T.-S. (2015). Learning Image and User Features for Recommendation in Social Networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4274–4282, Los Alamitos, CA, USA. IEEE Computer Society. DOI: 10.1109/ICCV.2015.486.

[Ghazanfar and Prugel-Bennett, 2010] Ghazanfar, M. A. and Prugel-Bennett, A. (2010). A Scalable, Accurate Hybrid Recommender System. In *Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining*, WKDD '10, pages 94–98, Phuket, Thailand. IEEE Computer Society. DOI: 10.1109/WKDD.2010.117.

[Ghazanfar et al., 2012] Ghazanfar, M. A., Prügel-Bennett, A., and Szedmak, S. (2012). Kernel-Mapping Recommender system algorithms. *Information Sciences*, 208:81–104. DOI: 10.1016/j.ins.2012.04.012.

[Giannotti et al., 2005] Giannotti, F., Mazzoni, A., Puntoni, S., and Renso, C. (2005). Synthetic Generation of Cellular Network Positioning Data. In *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*, GIS '05, page 12–20, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1097064.1097068.

[Gijsbers et al., 2022] Gijsbers, P., Bueno, M. L. P., Coors, S., LeDell, E., Poirier, S., Thomas, J., Bischl, B., and Vanschoren, J. (2022). AMLB: an AutoML Benchmark. *arXiv e-prints*. DOI: arXiv.2207.12560.

[Golbeck, 2006] Golbeck, J. (2006). Generating Predictive Movie Recommendations from Trust in Social Networks. In *Proceedings of the 4th International Conference on Trust Management*, volume 3986 of *iTrust'06*, pages 93–104, Pisa, Italy. Springer-Verlag. DOI: 10.1007/11755593_8.

[Golbeck et al., 2003] Golbeck, J., Parsia, B., and Hendler, J. (2003). Trust Networks on the Semantic Web. In Klusch, M., Omicini, A., Ossowski, S., and Laamanen, H., editors, *Cooperative Information Agents VII*, volume 2782, pages 238–249, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-45217-1_18.

[Golbeck and Hendler, 2005] Golbeck, J. A. and Hendler, J. (2005). *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland at College Park, USA. AAI3178583.

[Gomaa and Fahmy, 2013] Gomaa, W. and Fahmy, A. (2013). A Survey of Text Similarity Approaches. *international journal of Computer Applications*, 68(13):13–18. DOI: 10.5120/11638-7118.

[Gomez-Uribe and Hunt, 2015] Gomez-Uribe, C. A. and Hunt, N. (2015). The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13:1–13:19. DOI: 10.1145/2843948.

[Goncalves et al., 2020] Goncalves, A., Ray, P., Soper, B., Stevens, J., Coyle, L., and Sales, A. P. (2020). Generation and evaluation of synthetic patient data. *BMC Medical Research Methodology*, 20(1):1–40. DOI: 10.1186/s12874-020-00977-1.

[Gonzales et al., 2023] Gonzales, A., Guruswamy, G., and Smith, S. R. (2023). Synthetic data in health care: A narrative review. *PLOS Digital Health*, 2(1):e0000082. DOI: 10.1371/journal.pdig.0000082.

[Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc. DOI: 10.48550/arXiv.1406.2661.

[Google, 2023] Google (2023). Google Cloud AutoML - Train models without ML expertise. Available at https://cloud.google.com/automl (accessed at July 6th 2023).

[Gope and Jain, 2017] Gope, J. and Jain, S. K. (2017). A survey on solving cold start problem in recommender systems. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 133–138. IEEE. DOI: 10.1109/CCAA.2017.8229786.

[Goralski, 2017] Goralski, W. (2017). Chapter 11 - User Datagram Protocol. In Goralski, W., editor, *The Illustrated Network (Second Edition)*, pages 289–306. Morgan Kaufmann, Boston, second edition edition. DOI: 10.1016/B978-0-12-811027-0.00011-4.

[Gorry and Scott Morton, 1971] Gorry, G. A. and Scott Morton, M. S. (1971). A framework for management information systems. *Sloan Management Review*, 13:55–70.

[Goyal et al., 2016] Goyal, A., Lamb, A., Zhang, Y., Zhang, S., Courville, A., and Bengio, Y. (2016). Professor forcing: a new algorithm for training recurrent networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4608–4616, Red Hook, NY, USA. Curran Associates Inc. DOI: 10.5555/3157382.3157612.

[Grbovic et al., 2015] Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., and Sharp, D. (2015). E-commerce in Your Inbox: Product Recommendations at Scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1809–1818, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2783258.2788627.

[GRIDLEX, 2023] GRIDLEX (2023). Round Robin Ticket Assignment vs. Other Ticket Distribution Methods: Which is Best? Available at https://gridlex.com/a/round-robin-ticket-assignment-vs-other-ticket-distribution-methods-st2582 (accessed at September 13th 2024).

[GRIDLEX, 2024a] GRIDLEX (2024a). Mastering Ticket Prioritization: How to Effectively Manage Your Helpdesk Queue. Available at https://gridlex.com/a/mastering-ticket-prioritization-st339/ (accessed at September 13th 2024).

[GRIDLEX, 2024b] GRIDLEX (2024b). Ticket Escalation Best Practices: When and How to Escalate Helpdesk Issues. Available at https://gridlex.com/a/ticket-escalation-best-practices-st344 (accessed at September 13th 2024).

[Guha et al., 1998] Guha, S., Rastogi, R., and Shim, K. (1998). CURE: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, page 73–84, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/276304.276312.

[Guibas et al., 2017] Guibas, J. T., Virdi, T. S., and Li, P. S. (2017). Synthetic Medical Images from Dual Generative Adversarial Networks. *arXiv preprint*. DOI: arXiv:1709.01872.

[Guo, 2013] Guo, G. (2013). Integrating trust and similarity to ameliorate the data sparsity and cold start for recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 451–454, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2507157.2508071.

[Guo et al., 2015] Guo, G., Zhang, J., and Yorke-Smith, N. (2015). TrustSVD: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1). DOI: 10.1609/aaai.v29i1.9153.

[Guo et al., 2017] Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1725–1731. AAAI Press. DOI: 10.5555/3172077.3172127.

[Guy et al., 2009] Guy, I., Zwerdling, N., Carmel, D., Ronen, I., Uziel, E., Yogev, S., and Ofek-Koifman, S. (2009). Personalized Recommendation of Social Software Items Based on Social Relations. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 53–60, New York, NY, USA. ACM. DOI: 10.1145/1639714.1639725.

[H2O.ai, 2023] H2O.ai (2023). *H2O AutoML: Automatic Machine Learning*. H2O.ai, https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html.

[Han et al., 2018] Han, C., Hayashi, H., Rundo, L., Araki, R., Shimoda, W., Muramatsu, S., Furukawa, Y., Mauri, G., and Nakayama, H. (2018). GAN-based synthetic brain MR image generation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 734–738. DOI: 10.1109/ISBI.2018.8363678.

[Han et al., 2005] Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In Huang, D.-S., Zhang, X.-P., and Huang, G.-B., editors, *Advances in Intelligent Computing*, volume 3644, pages 878–887, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/11538059_91.

[Hanussek et al., 2021] Hanussek, M., Blohm, M., and Kintz, M. (2021). Can AutoML outperform humans? An evaluation on popular OpenML datasets using AutoML Benchmark. In *2020 2nd International Conference on Artificial Intelligence, Robotics and Control*, AIRC'20, page 29–32, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3448326.3448353.

[Hao et al., 2024] Hao, S., Han, W., Jiang, T., Li, Y., Wu, H., Zhong, C., Zhou, Z., and Tang, H. (2024). Synthetic Data in AI: Challenges, Applications, and Ethical Implications. *arXiv preprint arXiv:2401.01629*.

[Hariri et al., 2013] Hariri, N., Mobasher, B., and Burke, R. (2013). Query-driven context aware recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 9–16, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2507157.2507187.

[Hazan et al., 2011] Hazan, E., Koren, T., and Srebro, N. (2011). Beating SGD: learning SVMs in sublinear time. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, volume 24 of *NIPS'11*, page 1233–1241, Red Hook, NY, USA. Curran Associates Inc. DOI: 10.5555/2986459.2986597.

[He et al., 2008] He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on*

*Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. DOI: 10.1109/IJCNN.2008.46339694.

[He et al., 2016] He, J., Li, X., Liao, L., Song, D., and Cheung, W. K. (2016). Inferring a Personalized Next Point-of-Interest Recommendation Model with Latent Behavior Patterns. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 137–143. AAAI Press. DOI: 10.1609/aaai.v30i1.9994.

[He et al., 2017] He, R., Kang, W.-C., and McAuley, J. (2017). Translation-based Recommendation. *arXiv e-prints*. DOI: arXiv.1707.02410.

[He et al., 2020] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. (2020). LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 639–648, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3397271.3401063.

[He et al., 2017] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. DOI: 10.1145/3038912.3052569.

[He et al., 2021] He, X., Zhao, K., and Chu, X. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622. DOI: 10.1016/j.knosys.2020.106622.

[Hennigan and Bottorff, 2024] Hennigan, L. and Bottorff, C. (2024). What Is Workforce Management? Published: Forbes. Available at https://www.darpa.mil/ (accessed at October 28th 2024).

[Herce-Zelaya et al., 2020] Herce-Zelaya, J., Porcel, C., Bernabé-Moreno, J., Tejeda-Lorente, A., and Herrera-Viedma, E. (2020). New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests. *Information Sciences*, 536:156–170. DOI: 10.1016/j.ins.2020.05.071.

[Hidasi and Karatzoglou, 2017] Hidasi, B. and Karatzoglou, A. (2017). Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *arXiv e-prints*. DOI: arXiv.1706.03847.

[Hidasi et al., 2015] Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2015). Session-based Recommendations with Recurrent Neural Networks. *arXiv e-prints*. DOI: arXiv.1511.06939.

[Hidasi et al., 2016] Hidasi, B., Quadrana, M., Karatzoglou, A., and Tikk, D. (2016). Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 241–248, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2959100.2959167.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, 9(8):1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[Hossain and Islam, 2023] Hossain, M. A. and Islam, M. S. (2023). Ensuring network security with a robust intrusion detection system using ensemble-based machine learning. *Array*, 19:100306. DOI: 10.1016/j.array.2023.100306.

[Hota and Ghosh, 2013] Hota, J. and Ghosh, D. (2013). Workforce Analytics Approach: An Emerging Trend of Workforce Management. *AIMS International Journal*, 7(3):167–179. Available at SSRN: https://ssrn.com/abstract=2332713.

[Hsu and Li, 2021] Hsu, C. and Li, C.-T. (2021). RetaGNN: Relational Temporal Attentive Graph Neural Networks for Holistic Sequential Recommendation. In *Proceedings of the Web Conference 2021*, WWW '21, page 2968–2979, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3442381.3449957.

[Huang et al., 2018] Huang, J., Zhao, W. X., Dou, H., Wen, J.-R., and Chang, E. Y. (2018). Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 505–514, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3209978.3210017.

[Huang et al., 2013] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, page 2333–2338, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2505515.25056654.

[Huff et al., 2021] Huff, P., McClanahan, K., Le, T., and Li, Q. (2021). A Recommender System for Tracking Vulnerabilities. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, number 114 in ARES 21, pages 1–7, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3465481.3470039.

[Husák et al., 2022] Husák, M., Sadlek, L., Špaček, S., Laštovička, M., Javorník, M., and Komárková, J. (2022). CRUSOE: A toolset for cyber situational awareness and decision support in incident handling. *Computers & Security*, 115:102609. DOI: 10.1016/j.cose.2022.102609.

[Hutter et al., 2011] Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In Coello, C. A. C., editor, *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, volume 6683, pages 507–523. Springer. DOI: 10.1007/978-3-642-25566-3_40.

[Hutter et al., 2019] Hutter, F., Kotthoff, L., and Vanschoren, J. (2019). *Automated Machine Learning: Methods, Systems, Challenges*. Springer Publishing Company, Incorporated, 1st edition. ISBN: 3030053172.

[Idrissi et al., 2019] Idrissi, N., Zellou, A., Hourrane, O., Bakkoury, Z., and Benlahmar, E. H. (2019). Addressing Cold Start Challenges In Recommender Systems: Towards A New Hybrid Approach. In *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pages 1–6. DOI: 10.1109/SmartNets48225.2019.9069801.

[Inabo, 2023] Inabo, S. (2023). What is a ticketing system? Publisher: Zendesk. Available at https://www.zendesk.com/blog/ticketing-system/ (accessed at March 28th 2024).

[Invensis, 2022] Invensis (2022). What is a Help Desk? Why is it Indispensable for Your Organization? Available online at https://www.invensis.net/blog/what-is-help-desk-and-its-importance (accessed at April 28th 2024).

[Iqbal et al., 2019] Iqbal, M., Ghazanfar, M. A., Sattar, A., Maqsood, M., Khan, S., Mehmood, I., and Baik, S. W. (2019). Kernel Context Recommender System (KCR): A Scalable Context-Aware Recommender System Algorithm. *IEEE Access*, 7:24719–24737. DOI: 10.1109/ACCESS.2019.2897003.

[ISACA, 2022] ISACA (2022). State of Cybersecurity 2022. Technical report, Microsoft. https://www.isaca.org/-/media/files/isacadp/project/isaca/resources/infographics/state-of-cybersecurity-2022-part-1-infographic_0322.pdf?utm_source=twitter.com/thewebvy.

[Isinkaye et al., 2015] Isinkaye, F., Folajimi, Y., and Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273. DOI: 10.1016/j.eij.2015.06.005.

[Islam and Zhang, 2020] Islam, J. and Zhang, Y. (2020). GAN-based synthetic brain PET image generation. *Brain informatics*, 7(1):1–12. DOI: 10.1186/s40708-020-00104-2.

[ITarian, 2023] ITarian (2023). How Organizations Benefit from Embracing a Help Desk Ticketing System. Available at https://www.itarian.com/ticketing-system/benefits-of-ticketing-system.php (accessed at September 18th 2024).

[Jannach and Ludewig, 2017] Jannach, D. and Ludewig, M. (2017). When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 306–310, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3109859.3109872.

[Jomaa et al., 2021] Jomaa, H. S., Schmidt-Thieme, L., and Grabocka, J. (2021). Dataset2vec: Learning dataset meta-features. *Data Mining and Knowledge Discovery*, 35(3):964–985. DOI: 10.1007/s10618-021-00737-9.

[Joshi, 2019] Joshi, N. (2019). Discovering the Benefits of Synthetic Data. Publisher: BBN Times. Available at https://www.bbntimes.com/technology/discovering-the-benefits-of-synthetic-data (accessed at November 3rd 2020).

[Juneja, 2015] Juneja, P. (2015). Gaining Competitive Advantage with Decision Support Systems. Publisher: Management Study Guide (MSG). Available at https://www.managementstudyguide.com/gaining-competitive-advantage-with-decision-support-systems.htm (accessed at July 27th 2019).

[Juneja, 2024] Juneja, P. (2024). Limitations & Disadvantages of Decision Support Systems. Publisher: Management Study Guide (MSG). Available at https://www.managementstudyguide.com/limitations-and-disadvantages-of-decision-support-systems.htm (accessed at July 27th 2019).

[Kang and McAuley, 2018] Kang, W.-C. and McAuley, J. (2018). Self-Attentive Sequential Recommendation. *arXiv e-prints*. DOI: arXiv.1808.09781.

[Kantarci, 2020] Kantarci, A. (2020). Synthetic Data Generation: Techniques, Best Practices & Tools. Publisher: AI Multiple. Available at https://research.aimultiple.com/synthetic-data-generation/ (accessed at November 4th 2020).

[Kanti Karmaker Santu et al., 2021] Kanti Karmaker Santu, S., Mahadi Hassan, M., Smith, M. J., Xu, L., Zhai, C., and Veeramachaneni, K. (2021). AutoML to Date and Beyond: Challenges and Opportunities. *ACM Computing Surveys*, 54(8). DOI: 10.1145/3470918.

[Karatzoglou et al., 2010] Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. (2010). Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 79–86, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1864708.18647274.

[Karimi et al., 2018] Karimi, M., Jannach, D., and Jugovac, M. (2018). News recommender systems – Survey and roads ahead. *Information Processing & Management*, 54(6):1203–1227. DOI: 10.1016/j.ipm.2018.04.008.

[Karlapalepu, 2020] Karlapalepu, H. S. R. (2020). A Taxonomy of Sequential Patterns Based Recommendation Systems. Master's thesis, University of Windsor, Windsor, Ontario, Canada. https://scholar.uwindsor.ca/cgi/viewcontent.cgi?article=9455&context=etd.

[Karypis et al., 1999] Karypis, G., Han, E.-H., and Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *computer*, 32(8):68–75. DOI: 10.1109/2.781637.

[Kashada et al., 2020] Kashada, A., Isnoun, A., and Aldali, N. (2020). Effect of Information Overload on Decision's Quality, Efficiency and Time. *International Journal of Latest Engineering Research and Applications*, 5(1):53–58.

[Kawai et al., 2022] Kawai, M., Sato, H., and Shiohama, T. (2022). Topic model-based recommender systems and their applications to cold-start problems. *Expert Systems with Applications*, 202(C):117129. DOI: 10.1016/j.eswa.2022.117129.

[Kawai et al., 2018] Kawai, M., Shiohama, T., and Sato, H. (2018). Practically Feasible Recommender Systems for Cold Start Problems. In *2018 5th Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*, pages 103–112. DOI: 10.1109/APWConCSE.2018.00025.

[Kerschbaumer, 2018] Kerschbaumer, T. (2018). Convolutional Neural Networks for Sequence-Aware Recommender Systems. Master's thesis, Department of Computer Science and Engineering, Chalmers University of Technology Of Gothenburg, Gothenburg, Sweden.

[Ketler, 1993] Ketler, K. (1993). Case-based reasoning: An introduction. *Expert Systems with Applications. Special Issue: Case-Based Reasoning and its Applications.*, 6(1):3–8. DOI: 10.1016/0957-4174(93)90014-W2.

[Khosravi et al., 2022] Khosravi, H., Shum, S. B., Chen, G., Conati, C., Tsai, Y.-S., Kay, J., Knight, S., Martinez-Maldonado, R., Sadiq, S., and Gašević, D. (2022). Explainable Artificial Intelligence in education. *Computers and Education: Artificial Intelligence*, 3:100074. DOI: 10.1016/j.caeai.2022.100074.

[Kim et al., 2006] Kim, B. M., Li, Q., Park, C. S., Kim, S. G., and Kim, J. Y. (2006). A new approach for combining content-based and collaborative filters. *Journal of Intelligent Information Systems*, 27(1):79–91. DOI: 10.1007/s10844-006-8771-2.

[Kim and Kim, 2019] Kim, D. and Kim, H. K. (2019). Automated dataset generation system for collaborative research of cyber threat analysis. *Security and Communication Networks*, 2019(1):1–10. DOI: 10.1155/2019/6268476.

[Kim et al., 2016] Kim, D., Park, C., Oh, J., Lee, S., and Yu, H. (2016). Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 233–240, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2959100.2959165.

[Kim et al., 2011] Kim, H.-N., El-Saddik, A., and Jo, G.-S. (2011). Collaborative error-reflected models for cold-start recommender systems. *Decision Support Systems*, 51(3):519–531. DOI: 10.1016/j.dss.2011.02.015.

[Klöber-Koch et al., 2017] Klöber-Koch, J., Pielmeier, J., Grimm, S., Brandt, M. M., Schneider, M., and Reinhart, G. (2017). Knowledge-Based Decision Making in a Cyber-Physical Production Scenario. *Procedia Manufacturing*, 9:167–174. 7th Conference on Learning Factories, CLF 2017. DOI: 10.1016/j.promfg.2017.04.014.

[Ko et al., 2022] Ko, H., Lee, S., Park, Y., and Choi, A. (2022). A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. *Electronics*, 11(1). DOI: 10.3390/electronics11010141.

[Koene et al., 2015] Koene, A., Perez, E., Carter, C. J., Statache, R., Adolphs, S., O'Malley, C., Rodden, T., and McAuley, D. (2015). Ethics of Personalized Information Filtering. In Tiropanis, T., Vakali, A., Sartori, L., and Burnap, P., editors, *Internet Science*, volume 9089, pages 123–132, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-18609-2_10.

[Kokulu et al., 2019] Kokulu, F. B., Soneji, A., Bao, T., Shoshitaishvili, Y., Zhao, Z., Doupé, A., and Ahn, G.-J. (2019). Matched and Mismatched SOCs: A Qualitative Study on Security Operations Center Issues. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1955–1970, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3319535.3354239.

[Kong and Wu, 2018] Kong, D. and Wu, F. (2018). HST-LSTM: a hierarchical spatial-temporal long-short term memory network for location prediction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2341–2347. AAAI Press. DOI: 10.24963/ijcai.2018/324.

[Konstan and Riedl, 2012] Konstan, J. A. and Riedl, J. (2012). Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1):101–123. DOI: 10.1007/s11257-011-9112-x.

[Koren, 2008] Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 426–434, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1401890.1401944.

[Koren, 2010] Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1–24. DOI: 10.1145/1644873.1644874.

[Koren et al., 2009] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37. DOI: 10.1109/MC.2009.263.

[Kroenke and Boyle, 2015] Kroenke, D. M. and Boyle, R. J. (2015). *Using MIS*. Prentice Hall Press, Upper Saddle River, NJ, USA, 8th edition. ISBN: 9780133919868.

[Kulkarni and Rodd, 2020] Kulkarni, S. and Rodd, S. F. (2020). Context Aware Recommendation Systems: A review of the state of the art techniques. *Computer Science Review*, 37:100255. DOI: 10.1016/j.cosrev.2020.100255.

[Kumar et al., 2014] Kumar, G., Jerbi, H., Gurrin, C., and O'Mahony, M. P. (2014). Towards Activity Recommendation from Lifelogs. In *Proceedings of the 16th International Conference*

*on Information Integration and Web-Based Applications & Services*, iiWAS '14, page 87–96, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2684200.26842984.

[Kumar et al., 2021] Kumar, G., Jerbi, H., and O'Mahony, M. P. (2021). A sequence-based and context modelling framework for recommendation. *Expert Systems with Applications*, 175(C):114665. DOI: 10.1016/j.eswa.2021.114665.

[Kumar and Reddy, 2014] Kumar, P. N. V. and Reddy, D. V. R. (2014). A Survey on Recommender Systems (RSS) and Its Applications. *International Journal of Innovative Research in Computer and Communication Engineering*, 2:5254–5260.

[Kuppa et al., 2021] Kuppa, A., Aouad, L., and Le-Khac, N.-A. (2021). Towards Improving Privacy of Synthetic DataSet. In *Privacy Technologies and Policy*, pages 106–119. DOI: 10.1007/978-3-030-76663-4_6.

[Kůrková, 1991] Kůrková, V. (1991). Kolmogorov's theorem is relevant. *Neural Computation*, 3(4):617–622. DOI: 10.1162/neco.1991.3.4.617.

[Laudon and Laudon, 2013] Laudon, K. and Laudon, J. (2013). *Management Information Systems: Managing the Digital Firm*. Pearson. Edition: 13th. ISBN: 978-0133050691.

[Le et al., 2021] Le, H. H., Horino, Y., Yamazaki, T., Araki, K., and Yokota, H. (2021). Sequential Pattern Mining of Large Combinable Items with Values for a Set-of-items Recommendation. In *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 56–61, Los Alamitos, CA, USA. IEEE Computer Society. DOI: 10.1109/CBMS52027.2021.00017.

[Le et al., 2020] Le, J., Viswanathan, A., and Zhang, Y. (2020). *Generating High-fidelity Cybersecurity Data With Generative Adversarial Networks*, page 4117. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration. DOI: 10.2514/6.2020-4117.

[Le Guennec et al., 2016] Le Guennec, A., Malinowski, S., and Tavenard, R. (2016). Data augmentation for time series classification using convolutional neural networks. In *ECML/PKDD workshop on advanced analytics and learning on temporal data*, Riva Del Garda, Italy. HAL Id: halshs-01357973.

[Lee et al., 2019] Lee, H., Im, J., Jang, S., Cho, H., and Chung, S. (2019). MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1073–1082, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3292500.3330859.

[Lemos et al., 2022] Lemos, J., Silva, D., and Ferreira, L. (2022). Meta Feature Classification on a Synthetic Dataset Generator. Master's thesis, Faculty of Engineering, University of Porto (FEUP), R. Dr Roberto Frias, 4200-465, Porto, Portugal.

[Li et al., 2019] Li, C., Liu, Z., Wu, M., Xu, Y., Zhao, H., Huang, P., Kang, G., Chen, Q., Li, W., and Lee, D. L. (2019). Multi-interest network with dynamic routing for recommendation at tmall. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2615–2623. DOI: 10.1145/3357384.3357814.

[Li et al., 2017a] Li, G., Zhang, Z., Wang, L., Chen, Q., and Pan, J. (2017a). One-class collaborative filtering based on rating prediction and ranking prediction. *Knowledge-Based Systems*, 124:46–54. DOI: 10.1016/j.knosys.2017.02.034.

[Li et al., 2021] Li, J., Dai, J., Issakhov, A., Almojil, S. F., and Souri, A. (2021). Towards decision support systems for energy management in the smart industry and internet of things. *Computers & Industrial Engineering*, 161:107671. DOI: 10.1016/j.cie.2021.107671.

[Li et al., 2017b] Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., and Ma, J. (2017b). Neural Attentive Session-based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 1419–1428, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3132847.3132926.

[Li et al., 2020] Li, J., Wang, Y., and McAuley, J. (2020). Time Interval Aware Self-Attention for Sequential Recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 322–330, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3336191.3371786.

[Li and Tang, 2016] Li, L. and Tang, X.-j. (2016). A Solution to the Cold-Start Problem in Recommender Systems Based on Social Choice Theory. In Lavangnananda, K., Phon-Amnuaisuk, S., Engchuan, W., and Chan, J. H., editors, *Intelligent and Evolutionary Systems*, volume 5, pages 267–279, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-27000-5_22.

[Li et al., 2021] Li, L., Zhang, Y., and Chen, L. (2021). Personalized Transformer for Explainable Recommendation. *arXiv e-prints*. DOI: arXiv.2105.11601.

[Li et al., 2017c] Li, P., Wang, Z., Ren, Z., Bing, L., and Lam, W. (2017c). Neural Rating Regression with Abstractive Tips Generation for Recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 345–354, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3077136.3080822.

[Li et al., 2015] Li, X., Cong, G., Li, X.-L., Pham, T.-A. N., and Krishnaswamy, S. (2015). Rank-GeoFM: A Ranking based Geographical Factorization Method for Point of Interest Recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 433–442, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2766462.2767722.

[Lian et al., 2014] Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., and Rui, Y. (2014). GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In

*Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 831–840, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2623330.2623638.

[Liao, 2005] Liao, S.-H. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert Systems with Applications*, 28(1):93–103. DOI: 10.1016/j.eswa.2004.08.003.

[Lika et al., 2014] Lika, B., Kolomvatsos, K., and Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4, Part 2):2065–2073. DOI: 10.1016/j.eswa.2013.09.005.

[Lim et al., 2020] Lim, N., Hooi, B., Ng, S.-K., Wang, X., Goh, Y. L., Weng, R., and Varadarajan, J. (2020). STP-UDGAT: Spatial-Temporal-Preference User Dimensional Graph Attention Network for Next POI Recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 845–854, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3340531.3411876.

[Lin et al., 2006] Lin, P., Samadi, B., Cipolone, A., Jeske, D., Cox, S., Rendon, C., Holt, D., and Xiao, R. (2006). Development of a Synthetic Data Set Generator for Building and Testing Information Discovery Systems. In *Third International Conference on Information Technology: New Generations (ITNG'06)*, pages 707–712. DOI: 10.1109/ITNG.2006.51.

[Lin et al., 2020] Lin, Z., Jain, A., Wang, C., Fanti, G., and Sekar, V. (2020). Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 464–483, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3419394.3423643.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80. DOI: 10.1109/MIC.2003.1167344.

[Little et al., 2021] Little, C., Elliot, M., Allmendinger, R., and Samani, S. S. (2021). Generative Adversarial Networks for Synthetic Data Generation: A Comparative Study. *arXiv e-prints*. DOI: arXiv:2112.01925.

[Liu et al., 2022] Liu, F., Cheng, Z., Zhu, L., Liu, C., and Nie, L. (2022). An Attribute-Aware Attentive GCN Model for Attribute Missing in Recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(9):4077–4088. DOI: 10.1109/TKDE.2020.3040772.

[Liu et al., 2018] Liu, Q., Zeng, Y., Mokhosi, R., and Zhang, H. (2018). STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1831–1839, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3219819.3219950.

[Liu and Aberer, 2013] Liu, X. and Aberer, K. (2013). SoCo: a social network aided context-aware recommender system. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 781–802, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2488388.2488457.

[Liu et al., 2014] Liu, Y., Wei, W., Sun, A., and Miao, C. (2014). Exploiting Geographical Neighborhood Characteristics for Location Recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, page 739–748, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2661829.2662002.

[Livne et al., 2022] Livne, A., Tov, E. S., Solomon, A., Elyasaf, A., Shapira, B., and Rokach, L. (2022). Evolving context-aware recommender systems with users in mind. *Expert Systems with Applications*, 189(C):116042. DOI: 10.1016/j.eswa.2021.116042.

[Livne et al., 2019] Livne, A., Unger, M., Shapira, B., and Rokach, L. (2019). Deep Context-Aware Recommender System Utilizing Sequential Latent Context. *arXiv e-prints*. DOI: arXiv.1909.03999.

[Llort et al., 2019] Llort, N., Lusa, A., Martínez-Costa, C., and Mateo, M. (2019). A decision support system and a mathematical model for strategic workforce planning in consultancies. *Flexible Services and Manufacturing Journal*, 31(2):497–523. DOI: 10.1007/s10696-018-9321-2.

[Lonjarret, 2021] Lonjarret, C. (2021). *Sequential recommendation and explanations*. PhD thesis, Universite de Lyon, 92 Rue Pasteur, 69007 Lyon, France. HAL: tel-03117825.

[Lopes and Smith-Miles, 2013] Lopes, L. and Smith-Miles, K. (2013). Generating Applicable Synthetic Instances for Branch Problems. *Operations Research*, 61(3):563–577. DOI: 10.2307/23474003.

[Lorenzi and Ricci, 2005] Lorenzi, F. and Ricci, F. (2005). Case-based Recommender Systems: A Unifying View. In *Proceedings of the 2003 International Conference on Intelligent Techniques for Web Personalization*, volume 3169 of *ITWP'03*, pages 89–113, Acapulco, Mexico. Springer-Verlag. DOI: 10.1007/11577935_5.

[Lozano Murciego et al., 2021] Lozano Murciego, A., Jiménez-Bravo, D. M., Valera Román, A., De Paz Santana, J. F., and Moreno-García, M. N. (2021). Context-Aware Recommender Systems in the Music Domain: A Systematic Literature Review. *Electronics*, 10(13). DOI: 10.3390/electronics10131555.

[Lu et al., 2015] Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems*, 74(C):12–32. DOI: 10.1016/j.dss.2015.03.008.

[Lu et al., 2019] Lu, P.-H., Wang, P.-C., and Yu, C.-M. (2019). Empirical evaluation on synthetic data generation with generative adversarial network. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics*, WIMS2019, pages 1–6, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3326467.3326474.

[Luo et al., 2015] Luo, X., Zhou, M., Li, S., Xia, Y., You, Z., Zhu, Q., and Leung, H. (2015). An Efficient Second-Order Approach to Factorize Sparse Matrices in Recommender Systems. *IEEE Transactions on Industrial Informatics*, 11(4):946–956. DOI: 10.1109/TII.2015.2443723.

[Lyon and Popov, 2021] Lyon, B. and Popov, G. (2021). *"What-if" Analysis Methods*, chapter 7, pages 137–151. John Wiley & Sons, Ltd. DOI: 10.1002/9781119798323.ch7.

[Lyons, 2014] Lyons, K. B. (2014). A Recommender System in the Cyber Defense Domain. Master's thesis, Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way Wright-Patterson Air Force Base, Ohio 45433-7765.

[Lü et al., 2012] Lü, L., Medo, M., Yeung, C. H., Zhang, Y.-C., Zhang, Z.-K., and Zhou, T. (2012). Recommender systems. *Physics Reports*, 519(1):1–49. DOI: 10.1016/j.physrep.2012.02.006.

[Ma et al., 2019] Ma, C., Kang, P., and Liu, X. (2019). Hierarchical Gating Networks for Sequential Recommendation. *arXiv e-prints*. DOI: arXiv.1906.09217.

[Ma et al., 2020] Ma, C., Ma, L., Zhang, Y., Sun, J., Liu, X., and Coates, M. (2020). Memory Augmented Graph Neural Networks for Sequential Recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:5045–5052. DOI: 10.1609/aaai.v34i04.5945.

[Ma et al., 2009] Ma, H., King, I., and Lyu, M. R. (2009). Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210. DOI: 10.1145/1571941.1571978.

[Ma et al., 2008] Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). SoRec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. DOI: 10.1145/1458082.1458205.

[Ma et al., 2011] Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. (2011). Recommender systems with social regularization. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, page 287–296, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1935826.1935877.

[MacKenzie et al., 2013] MacKenzie, I., Meyer, C., and Noble, S. (2013). How retailers can keep up with consumers. Publisher: McKinsey & Company. Available at https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers (accessed at May 16th 2019).

[Mannino and Abouzied, 2020] Mannino, M. and Abouzied, A. (2020). Synner: Generating realistic synthetic data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 2749–2752, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3318464.3384696.

[Mao et al., 2021] Mao, K., Zhu, J., Xiao, X., Lu, B., Wang, Z., and He, X. (2021). UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 1253–1262, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3459637.3482291.

[Martinez-Gonzalez et al., 2020] Martinez-Gonzalez, P., Oprea, S., Garcia-Garcia, A., Jover-Alvarez, A., Orts-Escolano, S., and Garcia-Rodriguez, J. (2020). Unrealrox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *Virtual Reality*, 24(2):271–288. DOI: 10.1007/s10055-019-00399-5.

[Massa and Avesani, 2004] Massa, P. and Avesani, P. (2004). Trust-Aware Collaborative Filtering for Recommender Systems. In Meersman, R. and Tari, Z., editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, volume 3290, pages 492–508, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-30468-5_31.

[Massa and Avesani, 2007] Massa, P. and Avesani, P. (2007). Trust-aware recommender systems. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, page 17–24, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1297231.1297235.

[Massa and Bhattacharjee, 2004] Massa, P. and Bhattacharjee, B. (2004). Using trust in recommender systems: An experimental analysis. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29-April 1, 2004. Proceedings 2*, volume 2995, pages 221–235. Springer. DOI: 10.1007/978-3-540-24747-0_17.

[Masternak, 2023] Masternak, W. (2023). Ticket Management 101: Meet Your Secret Ingredient for Stress-Free Work. Publisher: Help Desk. Available at https://www.helpdesk.com/learn/ticket-management-101/ (accessed at March 28th 2024).

[MathWorks, 2005] MathWorks, I. (2005). *Statistics Toolbox for Use with MATLAB: User's Guide, Version 5*. MathWorks.

[McCann, 2023] McCann, M. (2023). The Quest To Close The Cybersecurity Talent Gap. Publisher: Forbes. Available at https://www.forbes.com/sites/forbeshumanresourcescouncil/2023/10/16/the-quest-to-close-the-cybersecurity-talent-gap/ (accessed at October 29th 2024).

[Mccrae and Costa, 1996] Mccrae, R. and Costa, P. (1996). Toward a new generation of personality theories: theoretical contexts for the Five-Factor Model. *The Five-Factor Model of Personality: Theoretical Perspectives*, 51:51–87.

[McDonnell et al., 2021] McDonnell, S., Nada, O., Abid, M. R., and Amjadian, E. (2021). CyberBERT: A Deep Dynamic-State Session-Based Recommender System for Cyber Threat Recognition. In *2021 IEEE Aerospace Conference (50100)*, pages 1–12. DOI: 10.1109/AERO50100.2021.9438286.

[Mendonça et al., 2020] Mendonça, S. D. P., Brito, Y. P. D. S., Santos, C. G. R. D., Lima, R. D. A. D., Araújo, T. D. O. D., and Meiguins, B. S. (2020). Synthetic Datasets Generator for Testing Information Visualization and Machine Learning Techniques and Tools. *IEEE Access*, 8:82917–82928. DOI: 10.1109/ACCESS.2020.2991949.

[Meng et al., 2014] Meng, S., Dou, W., Zhang, X., and Chen, J. (2014). KASR: A Keyword-Aware Service Recommendation Method on MapReduce for Big Data Applications. *IEEE Transactions on Parallel and Distributed Systems*, 25(12):3221–3231. DOI: 10.1109/TPDS.2013.2297117.

[Meyer-Berg et al., 2020] Meyer-Berg, A., Egert, R., Böck, L., and Mühlhäuser, M. (2020). IoT Dataset Generation Framework for Evaluating Anomaly Detection Mechanisms. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, page 1–6, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3407023.3407036.

[Mican et al., 2012] Mican, D., Mocean, L., and Tomai, N. (2012). Building a Social Recommender System by Harvesting Social Relationships and Trust Scores between Users. In Abramowicz, W., Domingue, J., and Węcel, K., editors, *Business Information Systems Workshops*, volume 127, pages 1–12, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-34228-8_1.

[Microsoft, 2023] Microsoft (2023). Azure Automated Machine Learning - AutoML | Microsoft Azure. Available at https://azure.microsoft.com/en-gb/products/machine-learning/automatedml (accessed at July 6th 2023).

[Microsoft Threat Intelligence, 2023] Microsoft Threat Intelligence (2023). Microsoft Digital Defense Report. Technical report, Microsoft.

[Miloslavskaya, 2016] Miloslavskaya, N. (2016). Security operations centers for information security incident management. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 131–136, Vienna, Austria. IEEE. DOI: 10.1109/FiCloud.2016.26.

[Minh et al., 2022] Minh, D., Wang, H. X., Li, Y. F., and Nguyen, T. N. (2022). Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 55:3503–3568. DOI: 10.1007/s10462-021-10088-y.

[Miok et al., 2019] Miok, K., Nguyen-Doan, D., Zaharie, D., and Robnik-Šikonja, M. (2019). Generating Data using Monte Carlo Dropout. In *2019 IEEE 15th International Conference*

*on Intelligent Computer Communication and Processing (ICCP)*, pages 509–515. IEEE. DOI: 10.1109/ICCP48234.2019.8959787.

[Mitchell and Ploem, 2018] Mitchell, C. and Ploem, C. (2018). Legal challenges for the implementation of advanced clinical digital decision support systems in Europe. *Journal of clinical and translational research*, 3(3):424–430. DOI: 10.18053/jctres.03.2017S3.005.

[Mobasher et al., 2004] Mobasher, B., Jin, X., and Zhou, Y. (2004). Semantically Enhanced Collaborative Filtering on the Web. In Berendt, B., Hotho, A., Mladenič, D., van Someren, M., Spiliopoulou, M., and Stumme, G., editors, *Web Mining: From Web to Semantic Web*, volume 3209, pages 57–76, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-30123-3_4.

[Mogren, 2016] Mogren, O. (2016). C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv e-prints*. DOI: arXiv.1611.09904.

[Mohamed et al., 2019] Mohamed, M. H., Khafagy, M. H., and Ibrahim, M. H. (2019). Recommender Systems Challenges and Solutions Survey. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 149–155, Aswan, Egypt, Egypt. IEEE. DOI: 10.1109/ITCE.2019.8646645.

[Mora et al., 2002] Mora, M., A. Forgionne, G., and Gupta, J. (2002). *Decision-Making Support Systems: Achievements and Challenges for the New Decade*. IGI Global, Hershey, PA, USA. ISBN: 9781591400455.

[Mozo et al., 2021] Mozo, A., González-Prieto, Á., Perales, A. P., Canaval, S. G., and Talavera, E. (2021). Synthetic flow-based cryptomining attack generation through generative adversarial networks. *Scientific Reports*, 12(1):2091. DOI: 0.1038/s41598-022-06057-2.

[Mughal, 2022] Mughal, A. A. (2022). Building and Securing the Modern Security Operations Center (SOC). *International Journal of Business Intelligence and Big Data Analytics*, 5(1):1–15. Retrieved from https://research.tensorgate.org/index.php/IJBIBDA/article/view/21.

[Natarajan et al., 2020] Natarajan, S., Vairavasundaram, S., Natarajan, S., and Gandomi, A. H. (2020). Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data. *Expert Systems with Applications*, 149(3):113248. DOI: 10.1016/j.eswa.2020.113248.

[National Institute of Standards and Technology, 2024] National Institute of Standards and Technology (2024). The NIST Cybersecurity Framework (CSF) 2.0. Technical report, U.S. Department of Commerce, Washington, D.C. DOI: : 10.6028/NIST.CSWP.29.

[Nawara and Kashef, 2021] Nawara, D. and Kashef, R. (2021). Context-Aware Recommendation Systems in the IoT Environment (IoT-CARS)–A Comprehensive Overview. *IEEE Access*, 9:144270–144284. DOI: 10.1109/ACCESS.2021.3122098.

[Nembhard et al., 2019] Nembhard, F. D., Carvalho, M. M., and Eskridge, T. C. (2019). Towards the application of recommender systems to secure coding. *EURASIP Journal on Information Security*, 2019(1):1–24. DOI: 10.1186/s13635-019-0092-4.

[Nettleton, 2016] Nettleton, D. (2016). A Synthetic Data Generator for Online Social Network Graphs. *Social Network Analysis and Mining*, 6(44). DOI: 10.1007/s13278-016-0352-y.

[Nguyen et al., 2018] Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., and Kim, S. (2018). Continuous-Time Dynamic Network Embeddings. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 969–976, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. DOI: 10.1145/3184558.3191526.

[Nguyen et al., 2014] Nguyen, H., Mary, J., and Preux, P. (2014). Cold-start Problems in Recommendation Systems via Contextual-bandit Algorithms. *arXiv e-prints*. DOI: 10.48550/arXiv.1405.7544.

[Nielsen, 1994] Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN: 9780080520292.

[Nikitin et al., 2022] Nikitin, N. O., Vychuzhanin, P., Sarafanov, M., Polonskaia, I. S., Revin, I., Barabanova, I. V., Maximov, G., Kalyuzhnaya, A. V., and Boukhanovsky, A. (2022). Automated evolutionary approach for the design of composite machine learning pipelines. *Future Generation Computer Systems*, 127:109–125. DOI: 10.1016/j.future.2021.08.022.

[Nikolenko, 2019] Nikolenko, S. I. (2019). Synthetic Data for Deep Learning. *arXiv e-prints*. DOI: arXiv.1909.11512.

[Nogueira, 2022] Nogueira, C. A. (2022). Automating an Open-Source Security Operations Center: Deploying, Updating and Scaling. Master's thesis, Universidade do Porto (Portugal), R. Dr Roberto Frias, 4200-465, Porto, Portugal.

[Nowok et al., 2016] Nowok, B., Raab, G. M., and Dibben, C. (2016). synthpop: Bespoke creation of synthetic data in R. *Journal of statistical software*, 74(11):1–26. DOI: 10.18637/jss.v074.i11.

[NSS, 2023] NSS (2023). *FEDOT's documentation*. NSS Lab, https://fedot.readthedocs.io/en/latest/.

[Nunnally et al., 2013] Nunnally, T., Abdullah, K., Uluagac, A. S., Copeland, J. A., and Beyah, R. (2013). NAVSEC: A Recommender System for 3D Network Security Visualizations. In *Proceedings of the Tenth Workshop on Visualization for Cyber Security*, VizSec '13, page 41–48, Atlanta, Georgia, USA. Association for Computing Machinery. DOI: 10.1145/2517957.2517963.

[Olson and Moore, 2023] Olson, R. and Moore, J. (2023). *Tree-Based Pipeline Optimization Tool (TPOT)*. Epistasis Lab at Cedars Sinai, http://epistasislab.github.io/tpot/.

[Omari et al., 2008] Omari, A., Langer, R., and Conrad, S. (2008). TARtool: A Temporal Dataset Generator for Market Basket Analysis. In Tang, C., Ling, C. X., Zhou, X., Cercone, N. J., and Li, X., editors, *Advanced Data Mining and Applications*, volume 5139, pages 400–410, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-88192-6_37.

[O'Shaughnessy and Gray, 2011] O'Shaughnessy, S. and Gray, G. (2011). Development and Evaluation of a Dataset Generator Tool for Generating Synthetic Log Files Containing Computer Attack Signatures. *International Journal of Ambient Computing and Intelligence (IJACI)*, 3(2):64–76. DOI: 10.4018/jaci.2011040105.

[O'Sullivan et al., 2004] O'Sullivan, D., Smyth, B., and C. Wilson, D. (2004). Preserving Recommender Accuracy and Diversity in Sparse Datasets. *International Journal on Artificial Intelligence Tools*, 13(1):219–235. DOI: 10.1142/S0218213004001491.

[OTA, 2019] OTA (2019). Online Trust Alliance (OTA) - Best Practices. Publisher: Internet Society (ISOC). Available at https://www.internetsociety.org/ota/resources-ota-best-practices/ (accessed at July 2nd 2019).

[Owens et al., 2022] Owens, E., Sheehan, B., Mullins, M., Cunneen, M., Ressel, J., and Castignani, G. (2022). Explainable Artificial Intelligence (XAI) in Insurance. *Risks*, 10(12):230. DOI: 10.3390/risks10120230.

[Pan and Chen, 2013a] Pan, W. and Chen, L. (2013a). *CoFiSet: Collaborative Filtering via Learning Pairwise Preferences over Item-sets*, pages 180–188. Society for Industrial and Applied Mathematics (SIAM). DOI: 10.1137/1.9781611972832.20.

[Pan and Chen, 2013b] Pan, W. and Chen, L. (2013b). GBPR: group preference based Bayesian personalized ranking for one-class collaborative filtering. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 2691–2697. AAAI Press. DOI: 10.5555/2540128.2540516.

[Panniello et al., 2014] Panniello, U., Tuzhilin, A., and Gorgoglione, M. (2014). Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*, 24:35–65. DOI: 10.1007/s11257-012-9135-y.

[Park et al., 2012] Park, D. H., Kim, H. K., Choi, I. Y., and Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059–10072. DOI: 10.1016/j.eswa.2012.02.038.

[Park et al., 2017] Park, K., Lee, J., and Choi, J. (2017). Deep Neural Networks for News Recommendations. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 2255–2258, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3132847.3133154.

[Park et al., 2018] Park, N., Mohammadi, M., Gorde, K., Jajodia, S., Park, H., and Kim, Y. (2018). Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083. DOI: 10.14778/3231751.3231757.

[Pastor et al., 2018] Pastor, A., Mozo, A., Lopez, D. R., Folgueira, J., and Kapodistria, A. (2018). The Mouseworld, a security traffic analysis lab based on NFV/SDN. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, number 57 in ARES '18, pages 1–6, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3230833.3233283.

[Pawlicka et al., 2021] Pawlicka, A., Pawlicki, M., Kozik, R., and Choraś, R. S. (2021). A Systematic Review of Recommender Systems and Their Applications in Cybersecurity. *Sensors*, 21(15). DOI: 10.3390/s21155248.

[Paz et al., 2015] Paz, F., Paz, F. A., and Pow-Sang, J. A. (2015). Experimental Case Study of New Usability Heuristics. In Marcus, A., editor, *Design, User Experience, and Usability: Design Discourse*, volume 9186, pages 212–223, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-20886-2_21.

[Pazzani, 1999] Pazzani, M. J. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, 13(5-6):393–408. DOI: 10.1023/A:1006544522159.

[Pei and Zaïane, 2006] Pei, Y. and Zaïane, O. (2006). Synthetic Data Generator for Clustering and Outlier Analysis Technical report. Technical report, University of Alberta, Edmonton, Canada T6G 2E8.

[Pendleton and Xu, 2017] Pendleton, M. and Xu, S. (2017). A dataset generator for next generation system call host intrusion detection systems. In *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, pages 231–236. DOI: 10.1109/MILCOM.2017.8170835.

[Peng et al., 2021] Peng, B., Ren, Z., Parthasarathy, S., and Ning, X. (2021). HAM: Hybrid Associations Models for Sequential Recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(10):4838–4853. DOI: 10.1109/tkde.2021.3049692.

[Pereira and Hruschka, 2015] Pereira, A. L. V. and Hruschka, E. R. (2015). Simultaneous co-clustering and learning to address the cold start problem in recommender systems. *Knowledge-Based Systems*, 82:11–19. DOI: 10.1016/j.knosys.2015.02.016.

[Pereira et al., 2022] Pereira, P. J., Costa, N., Barros, M., Cortez, P., Durães, D., Silva, A., and Machado, J. (2022). A Comparison of Automated Time Series Forecasting Tools for Smart Cities. In *Progress in Artificial Intelligence: 21st EPIA Conference on Artificial Intelligence, EPIA 2022, Lisbon, Portugal, August 31–September 2, 2022, Proceedings*, volume 13566, pages 551–562. DOI: 10.1007/978-3-031-16474-3_45.

[Peska and Vojtas, 2018] Peska, L. and Vojtas, P. (2018). Off-line vs. On-line Evaluation of Recommender Systems in Small E-commerce. *arXiv e-prints*. DOI: arXiv:1809.03186.

[Pillai et al., 2013] Pillai, K. G., Angryk, R. A., and Aydin, B. (2013). A filter-and-refine approach to mine spatiotemporal co-occurrences. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, page 104–113, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2525314.2525367.

[Ping et al., 2017] Ping, H., Stoyanovich, J., and Howe, B. (2017). DataSynthesizer: Privacy-Preserving Synthetic Datasets. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, number 42 in SSDBM '17, pages 1–5, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3085504.3091117.

[Pitkow and Pirolli, 1999] Pitkow, J. and Pirolli, P. (1999). Mining longest repeating subsequences to predict world wide web surfing. In *Proceedings of the 2nd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 2*, USITS'99, page 13, USA. USENIX Association. DOI: 10.5555/1251480.1251493.

[Polatidis et al., 2017] Polatidis, N., Pimenidis, E., Pavlidis, M., and Mouratidis, H. (2017). Recommender Systems Meeting Security: From Product Recommendation to Cyber-Attack Prediction. In Boracchi, G., Iliadis, L., Jayne, C., and Likas, A., editors, *Engineering Applications of Neural Networks*, volume 744, pages 508–519, United Kingdom. Springer International Publishing. DOI: 10.1007/978-3-319-65172-9_43.

[Popić et al., 2019] Popić, S., Pavković, B., Velikić, I., and Teslic, N. (2019). Data generators: a short survey of techniques and use cases with focus on testing. In *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, pages 189–194. DOI: 10.1109/ICCE-Berlin47944.2019.8966202.

[Power, 2002] Power, D. J. (2002). *Decision support systems: concepts and resources for managers*. Westport, Conn. : Quorum Books, Westport, CN. ISBN: 9781567204971.

[Power, 2008] Power, D. J. (2008). Decision Support Systems: A Historical Overview. In *Handbook on Decision Support Systems 1: Basic Themes*, pages 121–140. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI: 10.1007/978-3-540-48713-5_7.

[Power, 2013] Power, D. J. (2013). How can managers improve security for decision support applications. Publisher: DSSResources.com. Available at http://dssresources.com/faq/index.php?action=artikel&cat=&id=264&artlang=en) (accessed at October 30th 2019).

[Pu et al., 2012] Pu, P., Chen, L., and Hu, R. (2012). Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4):317–355. DOI: 10.1007/s11257-011-9115-7.

[Quadrana et al., 2018] Quadrana, M., Cremonesi, P., and Jannach, D. (2018). Sequence-Aware Recommender Systems. *ACM Computing Surveys*, 51(4):1–36. DOI: 10.1145/3190616.

[Quadrana et al., 2017] Quadrana, M., Karatzoglou, A., Hidasi, B., and Cremonesi, P. (2017). Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 130–137, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3109859.3109896.

[Quiñones-Grueiro et al., 2019] Quiñones-Grueiro, M., Prieto-Moreno, A., Verde, C., and Llanes-Santiago, O. (2019). Decision Support System for Cyber Attack Diagnosis in Smart Water Networks. *IFAC-PapersOnLine*, 51(34):329–334. 2nd IFAC Conference on Cyber-Physical and Human Systems CPHS 2018. DOI: 10.1016/j.ifacol.2019.01.024.

[Rachkovskij and Kussul, 1998] Rachkovskij, D. A. and Kussul, E. M. (1998). DataGen: a generator of datasets for evaluation of classification algorithms. *Pattern Recognition Letters*, 19(7):537–544.

[Rao et al., 2015] Rao, N., Yu, H.-F., Ravikumar, P. K., and Dhillon, I. S. (2015). Collaborative Filtering with Graph Information: Consistency and Scalable Methods. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[Rasmussen et al., 2010] Rasmussen, J., Ehrlich, K., Ross, S., Kirk, S., Gruen, D., and Patterson, J. (2010). Nimble Cybersecurity Incident Management through Visualization and Defensible Recommendations. In *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, VizSec '10, page 102–113, Ottawa, Ontario, Canada. Association for Computing Machinery. DOI: 10.1145/1850795.1850807.

[Ray and Mahanti, 2010] Ray, S. and Mahanti, A. (2010). Improving Prediction Accuracy in Trust-Aware Recommender Systems. In *2010 43rd Hawaii International Conference on System Sciences*, pages 1–9. DOI: 10.1109/HICSS.2010.225.

[Raza and Ding, 2019] Raza, S. and Ding, C. (2019). Progress in context-aware recommender systems —An overview. *Computer Science Review*, 31(C):84–97. DOI: 10.1016/j.cosrev.2019.01.001.

[Razghandi et al., 2022] Razghandi, M., Zhou, H., Erol-Kantarci, M., and Turgut, D. (2022). Variational Autoencoder Generative Adversarial Network for Synthetic Data Generation in Smart Home. In *ICC 2022 - IEEE International Conference on Communications*, page 4781–4786. DOI: 10.1109/icc45855.2022.9839249.

[Razikin and Soewito, 2022] Razikin, K. and Soewito, B. (2022). Cybersecurity decision support model to designing information technology security system based on risk analysis and cybersecurity framework. *Egyptian Informatics Journal*, 23(3):383–404. DOI: 10.1016/j.eij.2022.03.001.

[Reif et al., 2012] Reif, M., Shafait, F., and Dengel, A. (2012). Dataset generation for meta-learning. *KI-2012: Poster and Demo Track*, pages 69–73.

[Reif et al., 2014] Reif, M., Shafait, F., Goldstein, M., Breuel, T., and Dengel, A. (2014). Automatic classifier selection for non-experts. *Pattern Analysis and Applications*, 17(1):83–96. DOI: 10.1007/s10044-012-0280-z.

[Rendle et al., 2009] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 452–461, Arlington, Virginia, USA. AUAI Press. DOI: 10.5555/1795114.1795167.

[Rendle et al., 2010] Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010). Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 811–820, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1772690.1772773.

[Rendle et al., 2011] Rendle, S., Gantner, Z., Freudenthaler, C., and Schmidt-Thieme, L. (2011). Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 635–644, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2009916.2010002.

[Ricci et al., 2011] Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to Recommender Systems Handbook. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 1–35. Springer US, Boston, MA. DOI: 10.1007/978-0-387-85820-3_1.

[Rivolli et al., 2022] Rivolli, A., Garcia, L. P., Soares, C., Vanschoren, J., and de Carvalho, A. C. (2022). Meta-features for meta-learning. *Knowledge-Based Systems*, 240:108101. DOI: 10.1016/j.knosys.2021.108101.

[Robin Dickerson, 2004] Robin Dickerson (2004). Incident Management 101 Preparation and Initial Response (aka Identification). Technical report, SANS Institute. https://sansorg.egnyte.com/dl/xA2zHfNRL2.

[Roy and Dutta, 2022] Roy, D. and Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1):59. DOI: 10.1186/s40537-022-00592-5.

[Said et al., 2011] Said, A., De Luca, E. W., and Albayrak, S. (2011). Inferring contextual user profiles-improving recommender performance. In *Proceedings of the 3rd RecSys workshop on context-aware recommender systems*, volume 791.

[Salakhutdinov and Mnih, 2007] Salakhutdinov, R. and Mnih, A. (2007). Probabilistic matrix factorization. In *Proceedings of the 21st International Conference on Neural Information*

*Processing Systems*, NIPS'07, page 1257–1264, Red Hook, NY, USA. Curran Associates Inc. DOI: 10.5555/2981562.2981720.

[Salton and McGill, 1986] Salton, G. and McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, US. ISBN: 0070544840.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, Hong Kong, Hong Kong. ACM. DOI: 10.1145/371920.372071.

[Sarwar et al., 2000] Sarwar, B., Karypis, G., Konstan, J., and T. Riedl, J. (2000). Application of Dimensionality Reduction in Recommender System – A Case Study. Technical report, Department of Computer Science and Engineering University of Minnesota, 4-192 EECS Building 200 Union Street SE Minneapolis, MN 55455-0159 US.

[Scapicchio et al., 2024] Scapicchio, M., Downie, A., and Finio, M. (2024). What is a security operations center (SOC)? Publisher: IBM. Available at https://www.ibm.com/topics/security-operations-center (accessed at March 27th 2024).

[Schafer et al., 2007] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web*, volume 4321, pages 291–324. Springer. DOI: 10.1007/978-3-540-72079-9_9.

[Schinagl et al., 2015] Schinagl, S., Schoon, K., and Paans, R. (2015). A Framework for Designing a Security Operations Centre (SOC). In *2015 48th Hawaii International Conference on System Sciences*, pages 2253–2262, Los Alamitos, CA, USA. IEEE Computer Society. DOI: 10.1109/HICSS.2015.270.

[Schwarz et al., 2017] Schwarz, M., Lobur, M., and Stekh, Y. (2017). Analysis of the effectiveness of similarity measures for recommender systems. In *2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, pages 275–277, Lviv, Ukraine. IEEE. DOI: 10.1109/CADSM.2017.7916133.

[Security, 2024] Security, C. (2024). SOC KPIs: Measuring the Effectiveness of Your Security Operations. Published: Cado Security. Available at https://www.cadosecurity.com/wiki/soc-kpis-measuring-the-effectiveness-of-your-security-operations (accessed at October 29th 2024).

[Sedhain et al., 2015] Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. (2015). AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 111–112, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2740908.2742726.

[Seeger, 2004] Seeger, M. (2004). Gaussian Processes for Machine Learning. *International journal of neural systems*, 14(2):69–106. DOI: 10.1142/S0129065704001899.

[Sencer and Ozel, 2013] Sencer, A. and Ozel, B. B. (2013). A simulation-based decision support system for workforce management in call centers. *SIMULATION*, 89(4):481–497. DOI: 10.1177/0037549712470169.

[Senyucel, 2009] Senyucel, Z. (2009). *Managing the Human Resource in the 21st century*. BookBoon. ISBN: 9788776814687.

[Sethi and Mehrotra, 2021] Sethi, R. and Mehrotra, M. (2021). Cold Start in Recommender Systems—A Survey from Domain Perspective. In Hemanth, J., Bestak, R., and Chen, J. I.-Z., editors, *Intelligent Data Communication Technologies and Internet of Things*, volume 57, pages 223–232, Singapore. Springer Singapore. DOI: 10.1007/978-981-15-9509-7_19.

[Shani and Gunawardana, 2011] Shani, G. and Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, Boston, MA. DOI: 10.1007/978-0-387-85820-3_8.

[Shardanand and Maes, 1995] Shardanand, U. and Maes, P. (1995). Social information filtering: algorithms for automating “word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 210–217, USA. ACM Press/Addison-Wesley Publishing Co. DOI: 10.1145/223904.223931.

[Sharma and Gera, 2013] Sharma, L. and Gera, A. (2013). A survey of recommendation system: Research challenges. *International Journal of Engineering Trends and Technology (IJETT)*, 4(5):1989–1992.

[Sharma et al., 2017] Sharma, R., Gopalani, D., and Meena, Y. (2017). Collaborative filtering-based recommender system: Approaches and research challenges. In *2017 3rd International Conference on Computational Intelligence Communication Technology (CICT)*, pages 1–6. IEEE. DOI: 10.1109/CIACT.2017.7977363.

[Sharma and Singh, 2016] Sharma, R. and Singh, R. (2016). Evolution of Recommender Systems from Ancient Times to Modern Era: A Survey. *Indian Journal of Science and Technology*, 9(20). DOI: 10.17485/ijst/2016/v9i20/88005.

[Shaw et al., 2010] Shaw, G., Xu, Y., and Geva, S. (2010). Using Association Rules to Solve the Cold-Start Problem in Recommender Systems. In Zaki, M. J., Yu, J. X., Ravindran, B., and Pudi, V., editors, *Advances in Knowledge Discovery and Data Mining*, volume 6118, pages 340–347, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-13657-3_37.

[Sheikholeslami et al., 1998] Sheikholeslami, G., Chatterjee, S., and Zhang, A. (1998). Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings*

*of the 24rd International Conference on Very Large Data Bases*, volume 98, pages 428–439. Morgan Kaufmann Publishers Inc.

[Shen et al., 2021] Shen, Y., Wu, Y., Zhang, Y., Shan, C., Zhang, J., Letaief, B. K., and Li, D. (2021). How Powerful is Graph Convolution for Recommendation? In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 1619–1629, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3459637.3482264.

[Shi et al., 2018] Shi, S., Zhang, M., Liu, Y., and Ma, S. (2018). Attention-based Adaptive Model to Unify Warm and Cold Starts Recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 127–136, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3269206.3271710.

[Shi et al., 2012] Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., and Hanjalic, A. (2012). CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, page 139–146, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2365952.2365981.

[Shiligin, 2019] Shiligin, O. (2019). Youtube Recommender System. Publisher: RPubs. Available at http://rpubs.com/OlgaShiligin/503578 (accessed at June 21th 2019).

[Shiravi et al., 2012] Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374. DOI: 10.1016/j.cose.2011.12.012.

[Shirkhorshidi et al., 2015] Shirkhorshidi, A. S., Aghabozorgi, S., and Wah, T. Y. (2015). A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. *PLOS ONE*, 10(12):1–20. DOI: 10.1371/journal.pone.0144059.

[Shokeen and Rana, 2020] Shokeen, J. and Rana, C. (2020). A study on features of social recommender systems. *Artificial Intelligence Review*, 53(2):965–988. DOI: 10.1007/s10462-019-09684-w.

[Silva et al., 2021] Silva, H., Silva, R., and Porto, F. (2021). SAGAD: Synthetic Data Generator for Tabular Datasets. In *Anais do XXXVI Simpósio Brasileiro de Bancos de Dados*, SBBD 2021, pages 1–12, Porto Alegre, RS, Brasil. Sociedade Brasileira de Computação - SBC. DOI: 10.5753/sbbd.2021.17861.

[Silveira et al., 2019] Silveira, T., Zhang, M., Lin, X., Liu, Y., and Ma, S. (2019). How good your recommender system is? A survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics*, 10(5):813–831. DOI: 10.1007/s13042-017-0762-9.

[Sinha and Swearingen, 2001] Sinha, R. R. and Swearingen, K. (2001). Comparing Recommendations Made by Online Systems and Friends. In Smeaton, A. F. and Callan, J., editors, *DELOS*, volume 01/W03 of *ERCIM Workshop Proceedings*. ERCIM.

[Skopik et al., 2014] Skopik, F., Settanni, G., Fiedler, R., and Friedberg, I. (2014). Semi-synthetic data set generation for security software evaluation. In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pages 156–163. DOI: 10.1109/PST.2014.6890935.

[Smirnova, 2018] Smirnova, E. (2018). Action-conditional Sequence Modeling for Recommendation. *arXiv e-prints*.

[Smith and Linden, 2017] Smith, B. and Linden, G. (2017). Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing*, 21(3):12–18. DOI: 10.1109/MIC.2017.72.

[Smith and Smith, 2020] Smith, K. E. and Smith, A. O. (2020). Conditional GAN for timeseries generation. *arXiv preprint*. DOI: arXiv:2006.16477.

[Smyth and Cotter, 2000] Smyth, B. and Cotter, P. (2000). A personalised TV listings service for the digital TV age. *Knowledge-Based Systems*, 13(2):53–59. DOI: 10.1016/S0950-7051(00)00046-0.

[Snyman and Kruger, 2019] Snyman, D. and Kruger, H. (2019). A management decision support system for evaluating information security behaviour. In *International Information Security Conference*, volume 1166, pages 15–27. Springer. DOI: 10.1007/978-3-030-43276-8_2.

[Sohafi-Bonab et al., 2023] Sohafi-Bonab, J., Hosseinzadeh Aghdam, M., and Majidzadeh, K. (2023). DCARS: Deep context-aware recommendation system based on session latent context. *Applied Soft Computing*, 143(6):110416. DOI: 10.1016/j.asoc.2023.110416.

[Sohrabi and Roshani, 2017] Sohrabi, M. K. and Roshani, R. (2017). Frequent itemset mining using cellular learning automata. *Computers in Human Behavior*, 68:244–253. DOI: 0.1016/j.chb.2016.11.036.

[Soldo et al., 2010] Soldo, F., Le, A., and Markopoulou, A. (2010). Predictive Blacklisting as an Implicit Recommendation System. In *Proceedings of the 29th Conference on Information Communications*, INFOCOM'10, page 1640–1648. IEEE Press. DOI: 10.5555/1833515.1833744.

[Son, 2016] Son, L. H. (2016). Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems*, 58:87–104. DOI: 10.1016/j.is.2014.10.001.

[Sondur et al., 2016] Sondur, M. S. D., Chigadani, M. A. P., and Nayak, S. (2016). Similarity measures for recommender systems: a comparative study. *Journal 4 Research - J4R Journal*, 2(3):76–80.

[Song et al., 2021] Song, K., Zeng, X., Zhang, Y., De Jonckheere, J., Yuan, X., and Koehl, L. (2021). An interpretable knowledge-based decision support system and its

applications in pregnancy diagnosis. *Knowledge-Based Systems*, 221(3):106835. DOI: 10.1016/j.knosys.2021.106835.

[Souissi et al., 2017] Souissi, S., Serhrouchni, A., Sliman, L., and Charroux, B. (2017). Security Incident Response: Towards a Novel Decision-Making System. In Madureira, A. M., Abraham, A., Gamboa, D., and Novais, P., editors, *Intelligent Systems Design and Applications*, volume 557, pages 667–676, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-53480-0_66.

[Spasić et al., 2016] Spasić, M., Jovanovik, M., and Prat-Pérez, A. (2016). An RDF Dataset Generator for the Social Network Benchmark with Real-World Coherence. In *Proceedings of the Workshop on Benchmarking Linked Data (BLINK 2016)*, pages 18–25. DOI: 10.13140/RG.2.2.30490.64965/1.

[Springenberg et al., 2016] Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. (2016). Bayesian optimization with robust Bayesian neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, volume 29 of *NIPS'16*, page 4141–4149, Red Hook, NY, USA. Curran Associates Inc.

[Sridharan, 2014] Sridharan, S. (2014). Introducing Serendipity in Recommender Systems Through Collaborative Methods. Master's thesis, University of Rhode Island, 45 Upper College Rd, Kingston, RI 02881, EUA.

[Srivastava et al., 2016] Srivastava, R., Hingmire, S., Palshikar, G. K., Chaurasia, S., and Dixit, A. (2016). CSRS: A Context and Sequence Aware Recommendation System. In *Proceedings of the 8th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE '16, page 8–15, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3015157.3015159.

[St. John and Swanston, 2024] St. John, M. and Swanston, B. (2024). Cybersecurity Stats: Facts And Figures You Should Know. Publisher: Forbes. Available at https://www.forbes.com/advisor/education/it-and-tech/cybersecurity-statistics/ (accessed at March 27th 2024).

[Steiman, 2021] Steiman, H. (2021). Considering customer service outsourcing? Ask these 4 questions first. Publisher: Zendesk. Available at https://www.zendesk.com/blog/considering-customer-service-outsourcing-ask-4-questions-first/ (accessed at March 28th 2024).

[Su et al., 2021] Su, C., Chen, M., and Xie, X. (2021). Graph Convolutional Matrix Completion via Relation Reconstruction. In *Proceedings of the 2021 10th International Conference on Software and Computer Applications*, ICSCA '21, page 51–56, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3457784.3457792.

[Suhaim and Berri, 2021] Suhaim, A. B. and Berri, J. (2021). Context-Aware Recommender Systems for Social Networks: Review, Challenges and Opportunities. *IEEE Access*, 9:57440–57463. DOI: 10.1109/ACCESS.2021.3072165.

[Sula, 2019] Sula, E. (2019). ProtecDDoS: A Recommender System for Distributed Denial-of-Service Protection Services. Master's thesis, University of Zurich, Zürich, Switzerland.

[Sun et al., 2019] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. (2019). BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv e-prints*. DOI: arXiv.1904.06690.

[Sun et al., 2020] Sun, K., Qian, T., Chen, T., Liang, Y., Nguyen, Q. V. H., and Yin, H. (2020). Where to Go Next: Modeling Long- and Short-Term User Preferences for Point-of-Interest Recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):214–221. DOI: 10.1609/aaai.v34i01.5353.

[Surendra and Mohan, 2017] Surendra, H. and Mohan, H. S. (2017). A Review Of Synthetic Data Generation Methods For Privacy Preserving Data Publishing. *International Journal of Scientific & Technology Research*, 6(3):95–101.

[Sutskever et al., 2011] Sutskever, I., Martens, J., and Hinton, G. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 1017–1024, Madison, WI, USA. Omnipress. DOI: 10.5555/3104482.3104610.

[Sutton et al., 2020] Sutton, R. T., Pincock, D., Baumgart, D. C., Sadowski, D. C., Fedorak, R. N., and Kroeker, K. I. (2020). An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ digital medicine*, 3(1):17. DOI: 10.1038/s41746-020-0221-y.

[Tang and Wang, 2018] Tang, J. and Wang, K. (2018). Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. *arXiv e-prints*, page arXiv:1809.07426.

[Tapia et al., 2020] Tapia, F., Mora, M. A., Fuertes, W., Aules, H., Flores, E., and Toulkeridis, T. (2020). From Monolithic Systems to Microservices: A Comparative Study of Performance. *Applied Sciences*, 10(17). DOI: 10.3390/app10175797.

[Tarus et al., 2018] Tarus, J. K., Niu, Z., and Mustafa, G. (2018). Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial intelligence review*, 50:21–48. DOI: 10.1007/s10462-017-9539-5.

[Torres, 2018] Torres, D. G. (2018). *Generation of Synthetic Data with Generative Adversarial Networks*. PhD thesis, KTH Royal Institute of Technology, School of Eletrical Engineering and Computer Science, Stockholm, Sweden.

[Tucker et al., 2020] Tucker, A., Wang, Z., Rotalinti, Y., and Myles, P. (2020). Generating high-fidelity synthetic patient data for assessing machine learning healthcare software. *Nature Publishing Group digital medicine*, 3(1):1–13. DOI: 10.1038/s41746-020-00353-9.

[Ungar et al., 1998] Ungar, L., Foster, D., Andre, E., Wars, S., Wars, F. S., Wars, D. S., and Whispers, J. H. (1998). Clustering Methods for Collaborative Filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence*, volume 1, pages 114–129. AAAI Press.

[Unger et al., 2016] Unger, M., Bar, A., Shapira, B., and Rokach, L. (2016). Towards latent context-aware recommendation systems. *Knowledge-Based Systems*, 104:165–178. DOI: 10.1016/j.knosys.2016.04.020.

[Unger et al., 2020] Unger, M., Tuzhilin, A., and Livne, A. (2020). Context-Aware Recommendations Based on Deep Learning Frameworks. *ACM Trans. Manage. Inf. Syst.*, 11(2):1–15. DOI: 10.1145/3386243.

[van den Oord et al., 2016] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. *arXiv e-prints*. DOI: arXiv.1609.03499.

[van Os, 2007] van Os, R. (2007). SOC-CMM Metrics 101 - A best practice for performance measurement in the SOC. Technical report, SOC-CMM, Vijfhuizenbaan 8, 5133NH, Riel, Netherlands.

[Vanschoren, 2018] Vanschoren, J. (2018). Meta-Learning: A Survey. *arXiv e-prints*. DOI: arXiv.1810.03548.

[Vanschoren et al., 2014] Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2014). Openml: networked science in machine learning. *SIGKDD Explorations Newsletter*, 15(2):49–60. DOI: 10.1145/2641190.2641198.

[Vaswani et al., 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention Is All You Need. *arXiv e-prints*.

[Vega-Márquez et al., 2019] Vega-Márquez, B., Rubio-Escudero, C., Riquelme, J. C., and Nepomuceno-Chamorro, I. (2019). Creation of synthetic data with conditional generative adversarial networks. In *International Workshop on Soft Computing Models in Industrial and Environmental Applications*, pages 231–240. Springer. DOI: 10.1007/978-3-030-20055-8_22.

[Verbert et al., 2012] Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I., and Duval, E. (2012). Context-Aware Recommender Systems for Learning: A Survey and Future Challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335. DOI: 10.1109/TLT.2012.11.

[Vielberth et al., 2020] Vielberth, M., Böhm, F., Fichtinger, I., and Pernul, G. (2020). Security Operations Center: A Systematic Study and Open Challenges. *IEEE Access*, 8:227756–227779. DOI: 10.1109/ACCESS.2020.3045514.

[Villegas et al., 2018] Villegas, N. M., Sánchez, C., Díaz-Cely, J., and Tamura, G. (2018). Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, 140(1):173–200. DOI: 10.1016/j.knosys.2017.11.003.

[Volkovs et al., 2017a] Volkovs, M., Yu, G., and Poutanen, T. (2017a). Dropoutnet: addressing cold start in recommender systems. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 30 of *NIPS'17*, page 4964–4973, Red Hook, NY, USA. Curran Associates Inc. DOI: 10.5555/3295222.3295249.

[Volkovs et al., 2017b] Volkovs, M., Yu, G. W., and Poutanen, T. (2017b). Content-based Neighbor Models for Cold Start in Recommender Systems. In *Proceedings of the Recommender Systems Challenge 2017*, number 7 in RecSys Challenge '17, pages 1–6, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3124791.3124792.

[Voo et al., 2022] Voo, J., Hemani, I., and Cassidy, D. (2022). National Cyber Power Index 2022. Technical report, Belfer Center for Science and International Affairs, Harvard Kennedy School. Report: https://www.belfercenter.org/publication/national-cyber-power-index-2022.

[Wan et al., 2017] Wan, Z., Zhang, Y., and He, H. (2017). Variational autoencoder based synthetic data generation for imbalanced learning. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. DOI: 10.1109/SSCI.2017.8285168.

[Wang and Cai, 2020] Wang, B. and Cai, W. (2020). Knowledge-Enhanced Graph Neural Networks for Sequential Recommendation. *Information*, 11(8):388. DOI: 10.3390/info11080388.

[Wang et al., 2018a] Wang, D., Liang, Y., Xu, D., Feng, X., and Guan, R. (2018a). A content-based recommender system for computer science publications. *Knowledge-Based Systems*, 157:1–9. DOI: 10.1016/j.knosys.2018.05.001.

[Wang et al., 2018b] Wang, H., Shen, H., Ouyang, W., and Cheng, X. (2018b). Exploiting POI-specific geographical influence for point-of-interest recommendation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 3877–3883. AAAI Press. DOI: 10.24963/ijcai.2018/539.

[Wang et al., 2019a] Wang, K., Xu, L., Huang, L., Wang, C.-D., and Lai, J.-H. (2019a). SDDRS: Stacked Discriminative Denoising Auto-Encoder based Recommender System. *Cognitive Systems Research*, 55:164–174. DOI: 10.1016/j.cogsys.2019.01.011.

[Wang et al., 2015a] Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., and Cheng, X. (2015a). Learning Hierarchical Representation Model for NextBasket Recommendation. In *Proceedings of the 38th*

*International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 403–412, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2766462.2767694.

[Wang et al., 2015b] Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., and Cheng, X. (2015b). Learning Hierarchical Representation Model for NextBasket Recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 403–412, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2766462.2767694.

[Wang et al., 2010] Wang, Q., Yuan, X., and Sun, M. (2010). Collaborative filtering recommendation algorithm based on hybrid user model. In *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, volume 4, pages 1985–1990, Yantai, China. IEEE. DOI: 10.1109/FSKD.2010.5569479.

[Wang et al., 2017] Wang, R., Fu, B., Fu, G., and Wang, M. (2017). Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17*, number 12 in ADKDD'17, pages 1–7, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3124749.3124754.

[Wang et al., 2021a] Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., and Chi, E. (2021a). DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021*, WWW '21, page 1785–1797, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3442381.3450078.

[Wang et al., 2021b] Wang, S., Cao, L., Wang, Y., Sheng, Q. Z., Orgun, M. A., and Lian, D. (2021b). A Survey on Session-based Recommender Systems. *ACM Computing Surveys*, 54(7):1–38. DOI: 10.1145/3465401.

[Wang et al., 2019b] Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q. Z., and Orgun, M. (2019b). Sequential recommender systems: challenges, progress and prospects. *arXiv preprint*. DOI: arXiv:2001.04830.

[Wang et al., 2016] Wang, W., Yin, H., Sadiq, S., Chen, L., Xie, M., and Zhou, X. (2016). SPORE: A sequential personalized spatial item recommender system. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 954–965, Los Alamitos, CA, USA. IEEE Computer Society. DOI: 10.1109/ICDE.2016.7498304.

[Wang et al., 2008] Wang, W.-K., Huang, H.-C., and Lai, M.-C. (2008). Design of a knowledge-based performance evaluation system: A case of high-tech state-owned enterprises in an emerging economy. *Expert Systems with Applications*, 34(3):1795–1803. DOI: 10.1016/j.eswa.2007.01.032.

[Wang et al., 2019c] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., and Yu, P. S. (2019c). Heterogeneous Graph Attention Network. In *The World Wide Web Conference*, WWW

'19, page 2022–2032, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3308558.3313562.

[Wang et al., 2020] Wang, X., Jin, H., Zhang, A., He, X., Xu, T., and Chua, T.-S. (2020). Disentangled Graph Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1001–1010, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3397271.3401137.

[Wehmeyer, 2021] Wehmeyer, C. (2021). How do you generate synthetic data? Publisher: Statice. Available at https://www.statice.ai/post/how-generate-synthetic-data (accessed at February 21st 2022).

[Wei and Chow, 2023] Wei, T. and Chow, T. W. (2023). FGCR: Fused graph context-aware recommender system. *Knowledge-Based Systems*, 277(6):110806. DOI: 10.1016/j.knosys.2023.110806.

[Wei et al., 2021] Wei, Y., Wang, X., Li, Q., Nie, L., Li, Y., Li, X., and Chua, T.-S. (2021). Contrastive Learning for Cold-Start Recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 5382–5390, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3474085.3475665.

[Wickramasinghe, 2023] Wickramasinghe, S. (2023). SOC Metrics: Security Metrics & KPIs for Measuring SOC Success. Published: Splunk. Available at https://www.splunk.com/en_us/blog/learn/security-operations-metrics.html (accessed at October 28th 2024).

[Wilburn and Wilburn, 2018] Wilburn, K. M. and Wilburn, H. R. (2018). The impact of technology on business and society. *Global Journal of Business Research*, 12(1):23–39. Available at SSRN: https://ssrn.com/abstract=3210856.

[Wu et al., 2022] Wu, J., He, X., Wang, X., Wang, Q., Chen, W., Lian, J., and Xie, X. (2022). Graph convolution machine for context-aware recommender system. *Frontiers of Computer Science*, 16(6):166614. DOI: 10.1007/s11704-021-0261-8.

[Wu et al., 2020] Wu, L., Li, S., Hsieh, C.-J., and Sharpnack, J. (2020). SSE-PT: Sequential Recommendation Via Personalized Transformer. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 328–337, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3383313.3412258.

[Wu et al., 2020] Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B. (2020). Graph Neural Networks in Recommender Systems: A Survey. *arXiv e-prints*. DOI: arXiv.2011.02260.

[Wu et al., 2018] Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., and Tan, T. (2018). Session-based Recommendation with Graph Neural Networks. *arXiv e-prints*. DOI: arXiv.1811.00855.

[Wu et al., 2017] Wu, W., Zhao, J., Zhang, C., Meng, F., Zhang, Z., Zhang, Y., and Sun, Q. (2017). Improving performance of tensor-based context-aware recommenders using Bias Tensor Factorization with context feature auto-encoding. *Knowledge-Based Systems*, 128:71–77. DOI: 10.1016/j.knosys.2017.04.011.

[Wu et al., 2016] Wu, Y., DuBois, C., Zheng, A. X., and Ester, M. (2016). Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, page 153–162, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2835776.2835837.

[Xian et al., 2021] Xian, Y., Zhao, T., Li, J., Chan, J., Kan, A., Ma, J., Dong, X. L., Faloutsos, C., Karypis, G., Muthukrishnan, S., and Zhang, Y. (2021). EX3: Explainable Attribute-aware Item-set Recommendations. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 484–494, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3460231.3474240.

[Xie et al., 2018] Xie, L., Lin, K., Wang, S., Wang, F., and Zhou, J. (2018). Differentially Private Generative Adversarial Network. *arXiv e-prints*. DOI: arXiv.1802.06739.

[Xie et al., 2016] Xie, M., Yin, H., Xu, F., Wang, H., and Zhou, X. (2016). Graph-Based Metric Embedding for Next POI Recommendation. In *Proceedings of the 17th International Conference on Web Information Systems Engineering - Volume 10042*, volume 10042 of *WISE 2016*, page 207–222, Berlin, Heidelberg. Springer-Verlag. DOI: 10.1007/978-3-319-48743-4_17.

[Xu et al., 2019a] Xu, C., Zhao, P., Liu, Y., Sheng, V. S., Xu, J., Zhuang, F., Fang, J., and Zhou, X. (2019a). Graph contextualized self-attention network for session-based recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 3940–3946. AAAI Press. DOI: 10.24963/ijcai.2019/547.

[Xu et al., 2016] Xu, J., Xing, T., and van der Schaar, M. (2016). Personalized Course Sequence Recommendations. *Transactions on Signal Processing*, 64(20):5340–5352. DOI: 10.1109/TSP.2016.2595495.

[Xu et al., 2019b] Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. (2019b). *Modeling tabular data using conditional GAN*, pages 7335–7345. Curran Associates Inc., Red Hook, NY, USA. DOI: 10.5555/3454287.3454946.

[Xu and Veeramachaneni, 2018] Xu, L. and Veeramachaneni, K. (2018). Synthesizing tabular data using generative adversarial networks. *arXiv e-prints*. DOI: arXiv:1811.11264.

[Xu et al., 2024] Xu, S., Wu, Z., Zhao, H., Shu, P., Liu, Z., Liao, W., Li, S., Sikora, A., Liu, T., and Li, X. (2024). Reasoning before comparison: LLM-enhanced semantic similarity metrics for domain specialized text analysis. *arXiv preprint*. DOI: arXiv:2402.11398.

[Yan et al., 2019] Yan, A., Cheng, S., Kang, W.-C., Wan, M., and McAuley, J. (2019). CosRec: 2D Convolutional Neural Networks for Sequential Recommendation. *arXiv e-prints*. DOI: arXiv.1908.09972.

[Yang et al., 2017] Yang, B., Lei, Y., Liu, J., and Li, W. (2017). Social Collaborative Filtering by Trust. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1633–1647. DOI: 10.1109/TPAMI.2016.2605085.

[Yang et al., 2013] Yang, X., Guo, Y., and Liu, Y. (2013). Bayesian-Inference-Based Recommendation in Online Social Networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(4):642–651. DOI: 10.1109/TPDS.2012.192.

[Ying et al., 2018] Ying, H., Zhuang, F., Zhang, F., Liu, Y., Xu, G., Xie, X., Xiong, H., and Wu, J. (2018). Sequential recommender system based on hierarchical attention network. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 3926–3932. AAAI Press. DOI: 10.24963/ijcai.2018/546.

[Yoo, 2022] Yoo, G. (2022). The Importance Of Time And Speed In Cybersecurity. Publisher: Forbes. Available at https://www.forbes.com/councils/forbestechcouncil/2021/01/22/the-importance-of-time-and-speed-in-cybersecurity/ (accessed at January 3rd 2025).

[Yoon et al., 2019a] Yoon, J., Jarrett, D., and van der Schaar, M. (2019a). Time-series Generative Adversarial Networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[Yoon et al., 2019b] Yoon, J., Jordon, J., and Schaar, M. v. d. (2019b). PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*.

[Young and Weckman, 2020] Young, W. and Weckman, G. (2020). A Team-Compatibility Decision Support System for the National Football League. *International Journal of Computer Science in Sport*, 19(1):60–101. DOI: 10.2478/ijcss-2020-0005.

[Yu et al., 2016] Yu, F., Liu, Q., Wu, S., Wang, L., and Tan, T. (2016). A Dynamic Recurrent Model for Next Basket Recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 729–732, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2911451.2914683.

[Yu et al., 2019] Yu, Z., Lian, J., Mahmoody, A., Liu, G., and Xie, X. (2019). Adaptive user modeling with long and short-term preferences for personalized recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 4213–4219. AAAI Press. DOI: 10.24963/ijcai.2019/585.

[Zaheer et al., 2017] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. (2017). Deep Sets. *arXiv e-prints*. DOI: 10.48550/arXiv.1703.06114.

[Zanker and Jessenitschnig, 2009] Zanker, M. and Jessenitschnig, M. (2009). Collaborative Feature-Combination Recommender Exploiting Explicit and Implicit User Feedback. In *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing*, CEC '09, pages 49–56, Vienna, Austria. IEEE Computer Society. DOI: 10.1109/CEC.2009.84.

[Zendesk, 2024] Zendesk (2024). What is a help desk? Definition, benefits, and functions. Available online at https://www.zendesk.com/blog/help-desk/ (accessed at April 26th 2024).

[Zhai et al., 2020] Zhai, Z., Martínez, J. F., Beltran, V., and Martínez, N. L. (2020). Decision support systems for agriculture 4.0: Survey and challenges. *Computers and Electronics in Agriculture*, 170:105256. DOI: 10.1016/j.compag.2020.105256.

[Zhang and Chen, 2019] Zhang, M. and Chen, Y. (2019). Inductive Matrix Completion Based on Graph Neural Networks. *arXiv e-prints*. DOI: arXiv.1904.12058.

[Zhang et al., 2014] Zhang, M., Tang, J., Zhang, X., and Xue, X. (2014). Addressing Cold Start in Recommender Systems: A Semi-Supervised Co-Training Algorithm. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, page 73–82, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/2600428.2609599.

[Zhang et al., 2018] Zhang, S., Tay, Y., Yao, L., and Sun, A. (2018). Next Item Recommendation with Self-Attention. *arXiv e-prints*. DOI: 10.arXiv.1808.06414.

[Zhang et al., 2023] Zhang, X., Kuang, Z., Zhang, Z., Huang, F., and Tan, X. (2023). Cold & Warm Net: Addressing Cold-Start Users in Recommender Systems. In *International Conference on Database Systems for Advanced Applications*. Springer. DOI: 10.1007/978-3-031-30678-5_40.

[Zhang and Chen, 2018] Zhang, Y. and Chen, X. (2018). Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192.

[Zhang et al., 2010] Zhang, Z.-K., Liu, C., Zhang, Y.-C., and Zhou, T. (2010). Solving the cold-start problem in recommender systems with social tags. *EPL (Europhysics Letters)*, 92(2):28002. DOI: 10.1209/0295-5075/92/28002.

[Zhao et al., 2020] Zhao, J., Wang, W., Zhang, Z., Sun, Q., Huo, H., Qu, L., and Zheng, S. (2020). TrustTF: A tensor factorization model using user trust and implicit feedback for context-aware recommender systems. *Knowledge-Based Systems*, 209(2):106434. DOI: 10.1016/j.knosys.2020.106434.

[Zhao et al., 2022] Zhao, P., Luo, A., Liu, Y., Xu, J., Li, Z., Zhuang, F., Sheng, V. S., and Zhou, X. (2022). Where to Go Next: A Spatio-Temporal Gated Network for Next POI Recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(5):2512–2524. DOI: 10.1109/TKDE.2020.3007194.

[Zhao et al., 2017] Zhao, S., Zhao, T., King, I., and Lyu, M. R. (2017). Geo-Teaser: Geo-Temporal Sequential Embedding Rank for Point-of-interest Recommendation. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 153–162, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. DOI: 10.1145/3041021.3054138.

[Zheng et al., 2023] Zheng, R., Qu, L., Cui, B., Shi, Y., and Yin, H. (2023). AutoML for Deep Recommender Systems: A Survey. *ACM Transactions Information Systems*, 41(4). DOI: 10.1145/3579355.

[Zheng et al., 2012] Zheng, Y., Burke, R., and Mobasher, B. (2012). Optimal feature selection for context-aware recommendation using differential relaxation. *ACM Recsys*, 12. DOI: 10.13140/2.1.3708.7525.

[Zheng et al., 2013] Zheng, Y., Burke, R., and Mobasher, B. (2013). Recommendation with Differential Context Weighting. In Carberry, S., Weibelzahl, S., Micarelli, A., and Semeraro, G., editors, *User Modeling, Adaptation, and Personalization*, volume 7899, pages 152–164, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-38844-6_13.

[Zhong et al., 2012] Zhong, E., Fan, W., and Yang, Q. (2012). Contextual collaborative filtering via hierarchical matrix factorization. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 744–755. SIAM. DOI: 10.1137/1.9781611972825.64.

[Zhou et al., 2019a] Zhou, F., Yin, R., Zhang, K., Trajcevski, G., Zhong, T., and Wu, J. (2019a). Adversarial Point-of-Interest Recommendation. In *The World Wide Web Conference*, WWW '19, page 3462–34618, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3308558.3313609.

[Zhou et al., 2019b] Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., and Gai, K. (2019b). Deep interest evolution network for click-through rate prediction. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, volume 33 of *AAAI'19/IAAI'19/EAAI'19*, pages 5941–5948. AAAI Press. DOI: 10.1609/aaai.v33i01.33015941.

[Zhou et al., 2018a] Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., Yan, Y., Jin, J., Li, H., and Gai, K. (2018a). Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*,

KDD '18, page 1059–1068, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3219819.3219823.

[Zhou et al., 2015] Zhou, W., Tang, L., Li, T., Shwartz, L., and Grabarnik, G. Y. (2015). Resolution recommendation for event tickets in service management. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 287–295. DOI: 10.1109/INM.2015.7140303.

[Zhou et al., 2018b] Zhou, W., Wen, J., Qu, Q., Zeng, J., and Cheng, T. (2018b). Shilling attack detection for recommender systems based on credibility of group users and rating time series. *PLOS ONE*, 13(5):1–17. DOI: 10.1371/journal.pone.0196533.

[Zhu et al., 2020] Zhu, Z., Sefati, S., Saadatpanah, P., and Caverlee, J. (2020). Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1121–1130, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3397271.3401178.

[Zicari et al., 2022] Zicari, P., Folino, G., Guarascio, M., and Pontieri, L. (2022). Combining deep ensemble learning and explanation for intelligent ticket management. *Expert Systems with Applications*, 206:117815. DOI: 10.1016/j.eswa.2022.117815.

[Ziegler et al., 2004] Ziegler, C.-N., Lausen, G., and Schmidt-Thieme, L. (2004). Taxonomy-Driven Computation of Product Recommendations. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, page 406–415, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/1031171.1031252.

[Zou et al., 2015] Zou, H., Gong, Z., Zhang, N., Zhao, W., and Guo, J. (2015). TrustRank: a Cold-Start tolerant recommender system. *Enterprise Information Systems*, 9(2):117–138. DOI: 10.1080/17517575.2013.804587.

# Appendix A

# Appendix

## A.1 Influence of Team Dimensionality

Table A.1: Ticket Scheduling with S21sec Distribution - 20.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 5 | 0.39 | 2.07 | 6 | 0.41 | 2.09 | 4 | 0.26 | 1.65 | 6 | 0.41 | 2.13 | 5 | 0.3 | 1.78 | 4 | 0.28 | 1.76 | 3 | 0.2 | 1.38 | 5 | 0.32 | 1.88 | 5 | 0.31 | 1.78 | 4 | 0.28 | 1.73 |
| Scenario 2 | 13 | 0.91 | 3.24 | 13 | 1.05 | 3.65 | 10 | 0.62 | 2.51 | 14 | 1.05 | 3.59 | 11 | 0.75 | 2.88 | 10 | 0.68 | 2.76 | 7 | 0.41 | 2 | 11 | 0.82 | 3.19 | 11 | 0.77 | 2.92 | 10 | 0.7 | 2.81 |
| Scenario 3 | 4 | 0.27 | 1.73 | 4 | 0.28 | 1.76 | 3 | 0.18 | 1.32 | 4 | 0.28 | 1.81 | 3 | 0.23 | 1.59 | 3 | 0.19 | 1.43 | 2 | 0.15 | 1.19 | 3 | 0.21 | 1.48 | 3 | 0.23 | 1.55 | 3 | 0.21 | 1.5 |
| Scenario 4 | 7 | 0.49 | 2.32 | 9 | 0.67 | 3.01 | 5 | 0.33 | 1.8 | 8 | 0.55 | 2.46 | 6 | 0.41 | 2.03 | 5 | 0.34 | 1.89 | 4 | 0.25 | 1.62 | 6 | 0.46 | 2.36 | 7 | 0.47 | 2.3 | 6 | 0.44 | 2.34 |
| Scenario 5 | 17 | 1.63 | 5.36 | 20 | 2.7 | 9.31 | 14 | 1.08 | 3.88 | 20 | 2.08 | 6.22 | 15 | 1.43 | 5.05 | 14 | 1.14 | 3.92 | 10 | 0.74 | 3.46 | 15 | 1.55 | 5.59 | 17 | 1.88 | 6.61 | 15 | 1.49 | 5.17 |
| Scenario 6 | 4 | 0.28 | 1.76 | 5 | 0.34 | 1.96 | 3 | 0.2 | 1.38 | 4 | 0.31 | 1.84 | 3 | 0.25 | 1.62 | 3 | 0.2 | 1.49 | 3 | 0.16 | 1.24 | 4 | 0.26 | 1.73 | 4 | 0.26 | 1.65 | 3 | 0.24 | 1.6 |
| Scenario 7 | 8 | 0.58 | 2.49 | 11 | 0.92 | 3.49 | 7 | 0.43 | 2.04 | 10 | 0.75 | 2.93 | 7 | 0.51 | 2.37 | 8 | 0.55 | 2.5 | 5 | 0.3 | 1.74 | 8 | 0.55 | 2.56 | 8 | 0.61 | 2.67 | 7 | 0.48 | 2.32 |
| Scenario 8 | 23 | 2.49 | 7.03 | 29 | 4.38 | 12.72 | 20 | 1.91 | 5.76 | 27 | 3.54 | 9.28 | 20 | 1.93 | 5.64 | 22 | 2.46 | 7.26 | 14 | 1.08 | 4.02 | 21 | 2.07 | 6.54 | 23 | 2.59 | 7.69 | 20 | 1.99 | 6.15 |
| Scenario 9 | 6 | 0.43 | 2.14 | 8 | 0.67 | 2.91 | 5 | 0.3 | 1.71 | 7 | 0.52 | 2.38 | 6 | 0.38 | 1.96 | 5 | 0.34 | 1.89 | 4 | 0.24 | 1.65 | 6 | 0.42 | 2.28 | 7 | 0.47 | 2.28 | 6 | 0.41 | 2.22 |
| Scenario 10 | 30 | 14.03 | 40.17 | 33 | 23.74 | 57.48 | 27 | 8.05 | 24.39 | 32 | 18.53 | 48.93 | 29 | 12.52 | 36.45 | 27 | 8.3 | 24.95 | 23 | 5.74 | 20.51 | 28 | 11.84 | 34.58 | 31 | 19.91 | 51.34 | 30 | 14.7 | 39.75 |
| Scenario 11 | 39 | 15.16 | 40.4 | 44 | 26.55 | 61.22 | 36 | 9.23 | 25.91 | 44 | 21.49 | 53.08 | 37 | 13.96 | 38.42 | 38 | 9.68 | 25.36 | 28 | 5.99 | 20.63 | 37 | 13.07 | 36.03 | 39 | 21.04 | 51.28 | 37 | 16.11 | 41.73 |
| Scenario 12 | 13 | 1.31 | 4.79 | 17 | 2.56 | 9.51 | 11 | 0.91 | 3.75 | 15 | 1.72 | 5.96 | 13 | 1.23 | 4.59 | 11 | 0.88 | 3.48 | 8 | 0.67 | 3.31 | 13 | 1.5 | 6.66 | 15 | 1.75 | 6.61 | 13 | 1.4 | 5.58 |

Scenarios: 1 - Balanced Shifts with Fewer Operators; 2 - Balanced Shifts; 3 - Balanced Shifts with More Operators; 4 - Shifts with Low Imbalance and Fewer Operators; 5 - Shifts with Low Imbalance; 6 - Shifts with Low Imbalance and More Operators; 7 - Shifts with Medium Imbalance with Fewer Operators; 8 - Shifts with Medium Imbalance; 9 - Shifts with Medium Imbalance and More Operators; 10 - Shifts with High Imbalance with Fewer Operators; 11 - Shifts with High Imbalance; 12 - Shifts with High Imbalance and More Operators. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.

Table A.2: Ticket Scheduling with S21sec Distribution - 25.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 6 | 0.37 | 1.9 | 6 | 0.37 | 1.9 | 6 | 0.39 | 1.98 | 5 | 0.27 | 1.61 | 5 | 0.31 | 1.65 | 7 | 0.43 | 2.11 | 9 | 0.69 | 2.91 | 8 | 0.5 | 2.29 | 5 | 0.28 | 1.64 | 6 | 0.36 | 1.88 |
| Scenario 2 | 15 | 1.1 | 3.78 | 15 | 1.05 | 3.48 | 16 | 1.15 | 3.61 | 12 | 0.75 | 2.92 | 13 | 0.84 | 2.86 | 18 | 1.33 | 4.12 | 23 | 2.12 | 5.75 | 19 | 1.55 | 4.92 | 13 | 0.78 | 2.86 | 16 | 1.06 | 3.5 |
| Scenario 3 | 4 | 0.24 | 1.58 | 3 | 0.23 | 1.58 | 4 | 0.23 | 1.54 | 3 | 0.18 | 1.33 | 3 | 0.19 | 1.28 | 4 | 0.27 | 1.68 | 5 | 0.35 | 2.01 | 4 | 0.28 | 1.69 | 3 | 0.18 | 1.28 | 3 | 0.21 | 1.4 |
| Scenario 4 | 8 | 0.52 | 2.35 | 9 | 0.61 | 2.68 | 8 | 0.48 | 2.14 | 7 | 0.4 | 2.12 | 8 | 0.46 | 2.13 | 9 | 0.62 | 2.64 | 11 | 0.84 | 3.16 | 10 | 0.71 | 2.95 | 6 | 0.33 | 1.71 | 9 | 0.58 | 2.48 |
| Scenario 5 | 2 | 1.96 | 6.15 | 22 | 2.68 | 8.81 | 20 | 1.79 | 5.35 | 17 | 1.56 | 5.77 | 19 | 1.76 | 5.49 | 23 | 2.23 | 6.55 | 28 | 3.51 | 9.94 | 25 | 2.78 | 8.55 | 16 | 1.17 | 3.92 | 22 | 2.47 | 8.38 |
| Scenario 6 | 4 | 0.27 | 1.67 | 4 | 0.26 | 1.58 | 4 | 0.24 | 1.55 | 3 | 0.21 | 1.46 | 4 | 0.22 | 1.41 | 4 | 0.29 | 1.71 | 5 | 0.38 | 2.03 | 5 | 0.32 | 1.81 | 3 | 0.18 | 1.29 | 4 | 0.25 | 1.53 |
| Scenario 7 | 10 | 0.68 | 2.75 | 11 | 0.79 | 3 | 11 | 0.74 | 2.8 | 8 | 0.46 | 2.23 | 10 | 0.61 | 2.44 | 13 | 0.94 | 3.37 | 16 | 1.44 | 4.63 | 14 | 1.05 | 4.03 | 8 | 0.49 | 2.23 | 12 | 0.78 | 2.95 |
| Scenario 8 | 27 | 3.21 | 9.02 | 30 | 4.23 | 11.94 | 29 | 3.63 | 9.37 | 22 | 1.96 | 6.18 | 27 | 2.82 | 7.77 | 33 | 4.44 | 11.1 | 40 | 8.62 | 21.43 | 35 | 5.72 | 16.31 | 23 | 2.3 | 6.84 | 30 | 3.96 | 10.44 |
| Scenario 9 | 8 | 0.5 | 2.34 | 9 | 0.59 | 2.68 | 7 | 0.44 | 2.02 | 6 | 0.36 | 2 | 7 | 0.43 | 2.03 | 8 | 0.54 | 2.38 | 11 | 0.79 | 3.04 | 9 | 0.66 | 2.89 | 6 | 0.31 | 1.69 | 8 | 0.53 | 2.36 |
| Scenario 10 | 33 | 18.12 | 47.1 | 34 | 26.32 | 60.94 | 32 | 13.56 | 38.66 | 30 | 13.19 | 37.17 | 32 | 16.13 | 43.3 | 34 | 18.28 | 47.25 | 36 | 27.27 | 63.37 | 34 | 21.7 | 54.11 | 28 | 7.97 | 25.34 | 34 | 23.17 | 56.35 |
| Scenario 11 | 44 | 19.84 | 48.61 | 45 | 29.71 | 64.4 | 45 | 16.52 | 41.62 | 38 | 13.77 | 36.77 | 43 | 18.38 | 45.19 | 49 | 22.5 | 50.86 | 55 | 36.51 | 73.53 | 51 | 28.53 | 62.86 | 39 | 9.73 | 27.19 | 47 | 25.87 | 59.49 |
| Scenario 12 | 16 | 1.63 | 5.95 | 18 | 2.51 | 9.25 | 15 | 1.44 | 5.13 | 14 | 1.45 | 5.91 | 16 | 1.61 | 5.83 | 17 | 1.76 | 6.03 | 21 | 2.62 | 8.38 | 19 | 2.32 | 8.3 | 12 | 0.94 | 3.69 | 17 | 1.97 | 7.66 |

Same scenarios and metrics as in Table A.1

Table A.3: Ticket Scheduling with S21sec Distribution - 30.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 13 | 0.98 | 3.48 | 8 | 0.53 | 2.42 | 7 | 0.37 | 1.71 | 10 | 0.65 | 2.58 | 7 | 0.37 | 1.8 | 9 | 0.56 | 2.37 | 16 | 1.34 | 4.46 | 18 | 1.59 | 5.58 | 13 | 0.87 | 3.2 | 9 | 0.64 | 2.88 |
| Scenario 2 | 31 | 3.34 | 7.95 | 21 | 1.64 | 5.14 | 18 | 1.18 | 3.52 | 26 | 2.16 | 5.56 | 18 | 1.22 | 3.93 | 23 | 1.87 | 5.06 | 38 | 4.96 | 11.25 | 39 | 5.86 | 15.35 | 31 | 3.35 | 9.04 | 23 | 2.1 | 6.77 |
| Scenario 3 | 7 | 0.48 | 2.36 | 4 | 0.29 | 1.81 | 4 | 0.2 | 1.27 | 5 | 0.32 | 1.79 | 4 | 0.2 | 1.33 | 5 | 0.29 | 1.69 | 8 | 0.58 | 2.73 | 9 | 0.7 | 3.13 | 6 | 0.43 | 2.23 | 5 | 0.31 | 1.9 |
| Scenario 4 | 15 | 1.19 | 4 | 10 | 0.75 | 3.2 | 9 | 0.54 | 2.2 | 14 | 0.99 | 3.5 | 9 | 0.49 | 2.11 | 11 | 0.79 | 3.25 | 19 | 1.79 | 5.57 | 22 | 3.32 | 12.72 | 18 | 1.92 | 7.06 | 10 | 0.68 | 2.85 |
| Scenario 5 | 36 | 4.8 | 11.6 | 26 | 3.39 | 10.89 | 25 | 2.33 | 6.53 | 32 | 4.62 | 12.99 | 23 | 2.06 | 6.3 | 29 | 3.29 | 9.21 | 44 | 9.66 | 24.78 | 46 | 18.53 | 46.56 | 38 | 12.38 | 34.98 | 26 | 2.88 | 8.99 |
| Scenario 6 | 7 | 0.51 | 2.4 | 5 | 0.33 | 1.92 | 4 | 0.24 | 1.42 | 6 | 0.38 | 1.98 | 4 | 0.22 | 1.39 | 5 | 0.33 | 1.88 | 9 | 0.69 | 2.97 | 11 | 0.99 | 4.33 | 8 | 0.62 | 2.89 | 5 | 0.33 | 1.91 |
| Scenario 7 | 22 | 2.3 | 6.66 | 13 | 0.87 | 3.29 | 13 | 0.85 | 2.99 | 18 | 1.56 | 4.74 | 12 | 0.67 | 2.51 | 16 | 1.25 | 4.15 | 25 | 2.83 | 7.62 | 28 | 4.38 | 14.69 | 22 | 2.51 | 8.22 | 13 | 0.89 | 3.24 |
| Scenario 8 | 50 | 17.67 | 40.81 | 33 | 4.32 | 11.77 | 34 | 4.52 | 10.88 | 43 | 10.01 | 24.94 | 31 | 3.23 | 8.23 | 40 | 7.51 | 18.91 | 56 | 24.81 | 55.24 | 56 | 31.78 | 69.18 | 47 | 18.53 | 46.2 | 33 | 3.95 | 9.84 |
| Scenario 9 | 14 | 1.09 | 3.79 | 10 | 0.74 | 3.32 | 9 | 0.52 | 2.19 | 12 | 0.9 | 3.49 | 8 | 0.44 | 1.99 | 11 | 0.72 | 3.05 | 18 | 1.66 | 5.24 | 20 | 2.96 | 12.37 | 17 | 1.96 | 7.37 | 9 | 0.59 | 2.62 |
| Scenario 10 | 40 | 29.86 | 67.43 | 34 | 21.62 | 55.1 | 35 | 23.95 | 57.43 | 38 | 33.25 | 72.13 | 33 | 17.16 | 45.96 | 36 | 23.7 | 58.03 | 42 | 41.72 | 82.33 | 43 | 53.32 | 97.64 | 40 | 48.82 | 92.37 | 34 | 21.04 | 51.69 |
| Scenario 11 | 62 | 54.51 | 96.64 | 48 | 24.23 | 56.48 | 51 | 28.96 | 63.27 | 57 | 45.33 | 85.67 | 46 | 20.27 | 48.68 | 55 | 33.07 | 69.33 | 64 | 66.76 | 109.6 | 64 | 77.79 | 123.53 | 57 | 60.17 | 103.24 | 48 | 22.69 | 52.13 |
| Scenario 12 | 25 | 3.4 | 10.51 | 19 | 2.87 | 10.26 | 19 | 1.99 | 6.56 | 25 | 4.15 | 13.62 | 17 | 1.77 | 6.39 | 21 | 2.38 | 8.04 | 30 | 6.79 | 20.99 | 32 | 13.94 | 39.61 | 29 | 11.55 | 4.29 | 18 | 2.16 | 7.7 |

Scenarios: 1 - Balanced Shifts with Fewer Operators; 2 - Balanced Shifts; 3 - Balanced Shifts with More Operators; 4 - Shifts with Low Imbalance and Fewer Operators; 5 - Shifts with Low Imbalance; 6 - Shifts with Low Imbalance and More Operators; 7 - Shifts with Medium Imbalance with Fewer Operators; 8 - Shifts with Medium Imbalance; 9 - Shifts with Medium Imbalance and More Operators; 10 - Shifts with High Imbalance with Fewer Operators; 11 - Shifts with High Imbalance; 12 - Shifts with High Imbalance and More Operators. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.

Table A.4: Ticket Scheduling with S21sec Distribution - 35.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 14 | 0.84 | 2.99 | 22 | 1.84 | 5.32 | 11 | 0.61 | 2.41 | 16 | 1.22 | 4.41 | 23 | 2.01 | 5.52 | 15 | 1.1 | 3.77 | 18 | 1.35 | 4.1 | 17 | 1.35 | 4.7 | 15 | 1.03 | 3.53 | 17 | 1.28 | 4.16 |
| Scenario 2 | 32 | 3.08 | 7.41 | 48 | 8.43 | 21.22 | 28 | 2.16 | 5.85 | 38 | 4.82 | 12.36 | 51 | 8.46 | 17.94 | 35 | 4.08 | 9.97 | 42 | 5.2 | 11.24 | 38 | 5.38 | 15.95 | 36 | 4.04 | 9.8 | 40 | 4.97 | 11.63 |
| Scenario 3 | 6 | 0.37 | 1.82 | 10 | 0.74 | 3.04 | 5 | 0.27 | 1.58 | 7 | 0.48 | 2.4 | 11 | 0.78 | 2.98 | 7 | 0.49 | 2.37 | 8 | 0.57 | 2.54 | 8 | 0.53 | 2.56 | 7 | 0.42 | 2.06 | 8 | 0.53 | 2.48 |
| Scenario 4 | 16 | 1.06 | 3.55 | 28 | 4.31 | 14.57 | 15 | 0.96 | 3.41 | 18 | 1.29 | 4.27 | 28 | 3.67 | 11.12 | 20 | 2.04 | 7.15 | 22 | 2.19 | 6.93 | 24 | 3.31 | 11.41 | 18 | 1.54 | 5.26 | 22 | 2.26 | 7.65 |
| Scenario 5 | 37 | 5.18 | 14.86 | 55 | 28.06 | 60.95 | 35 | 5.16 | 15.24 | 42 | 6.42 | 16.48 | 57 | 23.85 | 52.93 | 44 | 13.25 | 34.44 | 48 | 13.19 | 36.57 | 47 | 20.98 | 51.78 | 41 | 9.09 | 27.32 | 46 | 13.89 | 36.08 |
| Scenario 6 | 7 | 0.41 | 1.95 | 14 | 1.2 | 4.45 | 6 | 0.34 | 1.78 | 8 | 0.5 | 2.39 | 14 | 1.1 | 3.94 | 9 | 0.65 | 2.83 | 10 | 0.75 | 3.07 | 11 | 0.9 | 4 | 8 | 0.55 | 2.54 | 10 | 0.75 | 3.33 |
| Scenario 7 | 21 | 1.7 | 5.02 | 37 | 6.79 | 20.02 | 21 | 1.66 | 5.34 | 24 | 2.41 | 7.3 | 36 | 5.74 | 15.41 | 26 | 3.09 | 9.18 | 29 | 3.5 | 9.29 | 30 | 5.03 | 17 | 24 | 2.42 | 7.11 | 28 | 3.4 | 9.84 |
| Scenario 8 | 49 | 12.47 | 30.39 | 65 | 58.27 | 96.22 | 48 | 13.31 | 33.58 | 54 | 18.9 | 41.18 | 66 | 50.57 | 88.72 | 55 | 27.44 | 57.94 | 59 | 32.5 | 66.47 | 58 | 39.57 | 78.47 | 53 | 21.04 | 48.9 | 57 | 30.08 | 63.63 |
| Scenario 9 | 14 | 0.97 | 3.43 | 26 | 4.13 | 13.93 | 14 | 0.85 | 3.06 | 15 | 1.12 | 4.04 | 26 | 3.5 | 11.08 | 20 | 2.08 | 7.48 | 20 | 2.08 | 6.83 | 23 | 3.21 | 11.47 | 16 | 1.47 | 5.39 | 20 | 2.08 | 7.14 |
| Scenario 10 | 39 | 33.74 | 71.47 | 47 | 60.05 | 104.96 | 39 | 34.06 | 71.91 | 41 | 32.49 | 69.87 | 47 | 54.9 | 98.14 | 42 | 49.58 | 91.89 | 44 | 45.46 | 87.82 | 44 | 57.44 | 101.57 | 41 | 40.07 | 81.26 | 43 | 47.63 | 89.95 |
| Scenario 11 | 60 | 48.28 | 86.13 | 72 | 105.38 | 143.63 | 59 | 50.19 | 89.25 | 64 | 57.59 | 94.98 | 72 | 98.9 | 137.85 | 64 | 70.92 | 110.77 | 67 | 76.89 | 117.84 | 66 | 88.18 | 130.99 | 63 | 63.39 | 105.49 | 65 | 74.77 | 117.32 |
| Scenario 12 | 26 | 3.92 | 12.85 | 38 | 20.27 | 49.74 | 26 | 4.25 | 14.76 | 27 | 4.15 | 13.33 | 37 | 16.16 | 42.15 | 32 | 10.56 | 31.45 | 33 | 9.55 | 30.07 | 35 | 16.03 | 42.52 | 29 | 7.48 | 25.48 | 33 | 10.7 | 31.38 |

Same scenarios and metrics as in Table A.3

Table A.5: Ticket Scheduling with S21sec Distribution - 40.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 24 | 1.92 | 5.28 | 27 | 2.41 | 6.61 | 16 | 1.06 | 3.55 | 24 | 2.06 | 5.74 | 18 | 1.2 | 3.76 | 20 | 1.55 | 5.01 | 24 | 2.02 | 5.79 | 18 | 1.09 | 3.39 | 19 | 1.27 | 3.91 | 16 | 0.95 | 3.14 |
| Scenario 2 | 53 | 9.48 | 21 | 57 | 13.41 | 31.36 | 38 | 4.82 | 11.62 | 53 | 11.08 | 23.38 | 43 | 5.14 | 11.87 | 44 | 6.68 | 17.1 | 52 | 8.9 | 20.49 | 40 | 4.19 | 9.52 | 42 | 4.68 | 10.43 | 37 | 3.65 | 9.07 |
| Scenario 3 | 11 | 0.73 | 2.85 | 13 | 0.83 | 3.14 | 7 | 0.42 | 2.08 | 11 | 0.73 | 2.89 | 8 | 0.45 | 2.08 | 9 | 0.58 | 2.57 | 11 | 0.74 | 2.89 | 8 | 0.42 | 1.95 | 8 | 0.47 | 2.13 | 7 | 0.4 | 1.92 |
| Scenario 4 | 28 | 3.03 | 8.99 | 32 | 4.03 | 11.85 | 18 | 1.27 | 4.12 | 27 | 2.58 | 7.8 | 23 | 2.18 | 6.67 | 26 | 3.81 | 14.36 | 31 | 4.35 | 13.87 | 21 | 1.68 | 5.49 | 23 | 2.05 | 6.63 | 21 | 1.71 | 6.04 |
| Scenario 5 | 58 | 19.48 | 44.96 | 62 | 33.11 | 66.55 | 43 | 6.72 | 17.19 | 57 | 16.18 | 37.32 | 50 | 14.04 | 35.08 | 51 | 24.86 | 57.61 | 59 | 27.53 | 61.13 | 47 | 10.28 | 27.85 | 49 | 13.11 | 35.97 | 46 | 10.56 | 29.61 |
| Scenario 6 | 13 | 0.92 | 3.45 | 15 | 1.11 | 3.74 | 8 | 0.44 | 2.05 | 12 | 0.81 | 3.31 | 11 | 0.65 | 2.67 | 12 | 1 | 4.33 | 15 | 1.21 | 4.31 | 10 | 0.55 | 2.35 | 10 | 0.6 | 2.52 | 9 | 0.54 | 2.33 |
| Scenario 7 | 36 | 5.25 | 13.35 | 43 | 11.29 | 31.05 | 26 | 3.2 | 9.7 | 34 | 4.39 | 11.24 | 30 | 3.38 | 9.13 | 32 | 4.86 | 15.8 | 39 | 6.46 | 17.27 | 29 | 3.06 | 8.41 | 30 | 3.27 | 8.96 | 28 | 2.8 | 8.03 |
| Scenario 8 | 68 | 60.52 | 94.54 | 71 | 88.52 | 117.36 | 55 | 26.94 | 55.57 | 67 | 49.84 | 82 | 61 | 35.16 | 67.04 | 61 | 38.41 | 75.08 | 69 | 62.19 | 97.56 | 59 | 31.8 | 62.77 | 61 | 36.29 | 69.04 | 58 | 29.52 | 60.84 |
| Scenario 9 | 25 | 2.64 | 8.06 | 30 | 3.86 | 12.02 | 17 | 1.15 | 3.88 | 23 | 2.24 | 7.15 | 22 | 2.07 | 6.62 | 24 | 3.72 | 13.57 | 28 | 4.03 | 13.01 | 20 | 1.6 | 5.49 | 20 | 1.78 | 5.89 | 19 | 1.63 | 5.79 |
| Scenario 10 | 48 | 51.07 | 92.91 | 51 | 57.94 | 100.16 | 42 | 24.56 | 57.81 | 47 | 47.63 | 89.05 | 45 | 47.73 | 89.08 | 45 | 58.96 | 102.6 | 49 | 57.36 | 100.03 | 43 | 43.5 | 84.85 | 44 | 47.22 | 89.09 | 42 | 43.08 | 84.35 |
| Scenario 11 | 73 | 105.84 | 138.07 | 76 | 138.82 | 160.91 | 65 | 62.45 | 101.51 | 73 | 98.24 | 131.45 | 68 | 81.32 | 119.32 | 68 | 82.65 | 123.06 | 74 | 107.84 | 140.75 | 67 | 78.29 | 118.36 | 68 | 83.09 | 121.04 | 66 | 74.7 | 114.24 |
| Scenario 12 | 37 | 12.65 | 35.46 | 42 | 17.91 | 44.68 | 28 | 3.51 | 11.04 | 36 | 10.46 | 30.3 | 34 | 9.93 | 29.03 | 35 | 18.98 | 48.79 | 39 | 18.05 | 47.63 | 32 | 7.89 | 25.41 | 34 | 10.17 | 31.27 | 33 | 8.28 | 26.59 |

Scenarios: 1 - Balanced Shifts with Fewer Operators; 2 - Balanced Shifts; 3 - Balanced Shifts with More Operators; 4 - Shifts with Low Imbalance and Fewer Operators; 5 - Shifts with Low Imbalance; 6 - Shifts with Low Imbalance and More Operators; 7 - Shifts with Medium Imbalance with Fewer Operators; 8 - Shifts with Medium Imbalance; 9 - Shifts with Medium Imbalance and More Operators; 10 - Shifts with High Imbalance with Fewer Operators; 11 - Shifts with High Imbalance; 12 - Shifts with High Imbalance and More Operators. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.



Table A.6: Ticket Scheduling without S21sec Distribution - 20.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 4 | 0.28 | 1.75 | 3 | 0.26 | 1.66 | 3 | 0.23 | 1.57 | 4 | 0.31 | 1.86 | 3 | 0.24 | 1.66 | 3 | 0.23 | 1.61 | 3 | 0.19 | 1.49 | 3 | 0.24 | 1.6 | 3 | 0.23 | 1.55 | 3 | 0.23 | 1.58 |
| Scenario 2 | 7 | 0.5 | 2.44 | 7 | 0.5 | 2.35 | 5 | 0.36 | 1.96 | 8 | 0.55 | 2.46 | 6 | 0.46 | 2.35 | 6 | 0.39 | 2.09 | 5 | 0.32 | 1.9 | 6 | 0.41 | 2.06 | 6 | 0.39 | 2.05 | 6 | 0.43 | 2.21 |
| Scenario 3 | 3 | 0.24 | 1.61 | 3 | 0.24 | 1.66 | 2 | 0.18 | 1.36 | 3 | 0.26 | 1.73 | 3 | 0.2 | 1.5 | 3 | 0.2 | 1.49 | 2 | 0.16 | 1.28 | 3 | 0.21 | 1.49 | 3 | 0.2 | 1.45 | 3 | 0.2 | 1.48 |
| Scenario 4 | 4 | 0.29 | 1.78 | 5 | 0.36 | 2.03 | 4 | 0.27 | 1.72 | 5 | 0.37 | 2.06 | 4 | 0.31 | 1.92 | 4 | 0.28 | 1.77 | 3 | 0.22 | 1.62 | 4 | 0.27 | 1.78 | 4 | 0.26 | 1.62 | 4 | 0.29 | 1.77 |
| Scenario 5 | 9 | 0.72 | 3.01 | 11 | 1.03 | 3.96 | 9 | 0.71 | 3.03 | 11 | 0.97 | 3.75 | 10 | 0.84 | 3.68 | 9 | 0.67 | 2.92 | 7 | 0.57 | 2.81 | 9 | 0.68 | 3.12 | 9 | 0.68 | 2.88 | 10 | 0.78 | 3.21 |
| Scenario 6 | 3 | 0.24 | 1.64 | 3 | 0.24 | 1.6 | 3 | 0.2 | 1.47 | 3 | 0.27 | 1.76 | 3 | 0.22 | 1.63 | 3 | 0.21 | 1.52 | 2 | 0.17 | 1.37 | 3 | 0.2 | 1.42 | 3 | 0.18 | 1.33 | 3 | 0.22 | 1.53 |
| Scenario 7 | 5 | 0.35 | 1.94 | 6 | 0.43 | 2.19 | 4 | 0.29 | 1.73 | 6 | 0.45 | 2.3 | 5 | 0.34 | 1.96 | 5 | 0.34 | 1.92 | 4 | 0.25 | 1.68 | 4 | 0.31 | 1.79 | 5 | 0.33 | 1.83 | 5 | 0.33 | 1.89 |
| Scenario 8 | 13 | 1.04 | 3.7 | 16 | 1.6 | 5.02 | 12 | 0.96 | 3.5 | 15 | 1.43 | 4.68 | 13 | 1.08 | 4.04 | 14 | 1.18 | 4.08 | 10 | 0.76 | 3.22 | 13 | 1.05 | 3.81 | 13 | 1.15 | 4.17 | 13 | 1.03 | 3.62 |
| Scenario 9 | 4 | 0.29 | 1.79 | 5 | 0.35 | 2.01 | 4 | 0.25 | 1.63 | 5 | 0.37 | 2.07 | 4 | 0.28 | 1.77 | 4 | 0.25 | 1.67 | 3 | 0.2 | 1.5 | 4 | 0.26 | 1.63 | 4 | 0.25 | 1.57 | 4 | 0.3 | 1.8 |
| Scenario 10 | 23 | 4.76 | 14.96 | 27 | 8.37 | 25.3 | 23 | 5.24 | 16.26 | 26 | 7.91 | 24.77 | 24 | 6.35 | 20.2 | 22 | 4.29 | 13.57 | 20 | 3.99 | 14.45 | 22 | 4.72 | 14.85 | 23 | 5.07 | 16.15 | 25 | 7.36 | 22.79 |
| Scenario 11 | 28 | 5.39 | 15.91 | 33 | 8.55 | 23.85 | 28 | 5.87 | 17.81 | 32 | 9.31 | 27.73 | 29 | 7.03 | 21.06 | 28 | 5.13 | 14.59 | 23 | 4.36 | 15.13 | 28 | 5.43 | 16.68 | 29 | 5.81 | 16.54 | 30 | 7.45 | 22.41 |
| Scenario 12 | 8 | 0.58 | 2.7 | 10 | 0.94 | 3.87 | 8 | 0.64 | 2.91 | 10 | 0.94 | 3.85 | 9 | 0.76 | 3.46 | 7 | 0.53 | 2.58 | 7 | 0.51 | 2.69 | 8 | 0.61 | 2.97 | 8 | 0.56 | 2.56 | 9 | 0.76 | 3.35 |

Same scenarios and metrics as in Table A.5

Table A.7: Ticket Scheduling without S21sec Distribution - 25.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 5 | 0.35 | 2.01 | 4 | 0.35 | 2.04 | 3 | 0.22 | 1.48 | 4 | 0.25 | 1.58 | 4 | 0.22 | 1.44 | 4 | 0.24 | 1.55 | 5 | 0.38 | 2.16 | 4 | 0.27 | 1.67 | 4 | 0.25 | 1.59 | 3 | 0.18 | 1.25 |
| Scenario 2 | 10 | 0.75 | 2.98 | 9 | 0.7 | 2.97 | 8 | 0.46 | 2.13 | 8 | 0.5 | 2.26 | 7 | 0.45 | 2.09 | 8 | 0.49 | 2.24 | 11 | 0.88 | 3.42 | 9 | 0.61 | 2.64 | 8 | 0.52 | 2.32 | 7. | 0.38 | 1.86 |
| Scenario 3 | 3 | 0.28 | 1.86 | 3 | 0.25 | 1.7 | 3 | 0.18 | 1.34 | 3 | 0.2 | 1.44 | 3 | 0.19 | 1.36 | 3 | 0.19 | 1.37 | 4 | 0.28 | 1.81 | 3 | 0.22 | 1.53 | 3 | 0.2 | 1.44 | 3 | 0.16 | 1.2 |
| Scenario 4 | 6 | 0.49 | 2.4 | 6 | 0.43 | 2.3 | 4 | 0.26 | 1.57 | 5 | 0.34 | 1.97 | 5 | 0.31 | 1.74 | 5 | 0.3 | 1.73 | 6 | 0.49 | 2.4 | 5 | 0.37 | 1.98 | 5 | 0.3 | 1.74 | 4 | 0.23 | 1.39 |
| Scenario 5 | 15 | 1.5 | 5.02 | 14 | 1.3 | 4.43 | 11 | 0.78 | 3.05 | 12 | 1.05 | 4.12 | 12 | 1.03 | 3.92 | 12 | 0.96 | 3.51 | 16 | 1.51 | 5.24 | 13 | 1.13 | 4 | 12 | 0.9 | 3.47 | 10 | 0.71 | 2.89 |
| Scenario 6 | 4 | 0.31 | 1.93 | 4 | 0.27 | 1.76 | 3 | 0.19 | 1.39 | 3 | 0.21 | 1.49 | 3 | 0.2 | 1.4 | 3 | 0.2 | 1.39 | 4 | 0.29 | 1.78 | 3 | 0.23 | 1.53 | 3 | 0.21 | 1.47 | 3 | 0.16 | 1.19 |
| Scenario 7 | 8 | 0.61 | 2.73 | 8 | 0.58 | 2.67 | 6 | 0.35 | 1.92 | 6 | 0.39 | 2.11 | 6 | 0.38 | 2.02 | 6 | 0.4 | 1.99 | 9 | 0.69 | 2.94 | 7 | 0.47 | 2.25 | 6 | 0.38 | 1.94 | 5 | 0.29 | 1.61 |
| Scenario 8 | 22 | 2.34 | 6.67 | 21 | 2.35 | 6.91 | 16 | 1.3 | 4.14 | 16 | 1.37 | 4.7 | 18 | 1.51 | 4.75 | 18 | 1.56 | 4.67 | 22 | 2.55 | 7.25 | 19 | 1.81 | 5.46 | 17 | 1.48 | 4.6 | 15 | 1.17 | 3.91 |
| Scenario 9 | 6 | 0.48 | 2.4 | 5 | 0.4 | 2.15 | 4 | 0.24 | 1.47 | 5 | 0.32 | 1.88 | 5 | 0.3 | 1.73 | 4 | 0.29 | 1.67 | 6 | 0.46 | 2.32 | 5 | 0.35 | 1.91 | 4 | 0.29 | 1.68 | 4 | 0.22 | 1.39 |
| Scenario 10 | 31 | 12.57 | 35.4 | 30 | 11.47 | 32.99 | 26 | 6.31 | 19.49 | 27 | 8.74 | 26.81 | 28 | 9.19 | 27.58 | 28 | 7.91 | 24.28 | 30 | 11.82 | 34.09 | 28 | 8.67 | 26.34 | 27 | 7.26 | 22.29 | 25 | 5.95 | 20.32 |
| Scenario 11 | 39 | 14.28 | 37.15 | 39 | 12.96 | 34.65 | 33 | 6.73 | 19.41 | 32 | 8.62 | 25.26 | 35 | 9.93 | 28.04 | 35 | 8.5 | 23.4 | 40 | 13.42 | 34.95 | 36 | 9.67 | 21.21 | 34 | 7.83 | 21.49 | 31 | 6.18 | 18.65 |
| Scenario 12 | 13 | 1.36 | 4.87 | 12 | 1.08 | 4.09 | 9 | 0.67 | 2.85 | 10 | 0.93 | 3.98 | 11 | 0.98 | 4.01 | 10 | 0.77 | 3.13 | 12 | 1.25 | 4.91 | 11 | 0.96 | 3.7 | 10 | 0.81 | 3.35 | 8 | 0.6 | 2.79 |

Scenarios: 1 - Balanced Shifts with Fewer Operators; 2 - Balanced Shifts; 3 - Balanced Shifts with More Operators; 4 - Shifts with Low Imbalance and Fewer Operators; 5 - Shifts with Low Imbalance; 6 - Shifts with Low Imbalance and More Operators; 7 - Shifts with Medium Imbalance with Fewer Operators; 8 - Shifts with Medium Imbalance; 9 - Shifts with Medium Imbalance and More Operators; 10 - Shifts with High Imbalance with Fewer Operators; 11 - Shifts with High Imbalance; 12 - Shifts with High Imbalance and More Operators. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.

Table A.8: Ticket Scheduling without S21sec Distribution - 30.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 6 | 0.46 | 2.28 | 5 | 0.38 | 2.11 | 4 | 0.26 | 1.55 | 6 | 0.4 | 2.1 | 4 | 0.27 | 1.62 | 4 | 0.26 | 1.62 | 8 | 0.59 | 2.69 | 7 | 0.56 | 2.76 | 7 | 0.53 | 2.54 | 5 | 0.35 | 1.95 |
| Scenario 2 | 14 | 1.14 | 3.79 | 13 | 0.93 | 3.45 | 10 | 0.6 | 2.44 | 13 | 0.99 | 3.51 | 10 | 0.63 | 2.51 | 10 | 0.61 | 2.48 | 19 | 1.61 | 4.81 | 17 | 1.51 | 4.96 | 16 | 1.28 | 4.19 | 12 | 0.88 | 3.24 |
| Scenario 3 | 4 | 0.3 | 1.8 | 3 | 0.26 | 1.75 | 10 | 0.6 | 2.44 | 4 | 0.27 | 1.74 | 3 | 0.2 | 1.39 | 3 | 0.18 | 1.31 | 5 | 0.37 | 2.2 | 4 | 0.37 | 2.22 | 4 | 0.35 | 2.09 | 3 | 0.24 | 1.6 |
| Scenario 4 | 8 | 0.56 | 2.52 | 7 | 0.55 | 2.67 | 6 | 0.38 | 1.92 | 7 | 0.55 | 2.59 | 6 | 0.38 | 1.97 | 5 | 0.34 | 1.89 | 10 | 0.8 | 3.25 | 9 | 0.75 | 3.36 | 9 | 0.69 | 3 | 7 | 0.47 | 2.29 |
| Scenario 5 | 19 | 1.88 | 5.64 | 19 | 1.96 | 6.48 | 15 | 1.32 | 4.78 | 19 | 1.92 | 6.01 | 15 | 1.28 | 4.78 | 14 | 1.13 | 4 | 25 | 3.04 | 8.31 | 22 | 2.78 | 9.19 | 21 | 2.46 | 7.38 | 17 | 1.56 | 5.34 |
| Scenario 6 | 4 | 0.32 | 1.88 | 4 | 0.29 | 1.85 | 3 | 0.21 | 1.39 | 4 | 0.29 | 1.79 | 3 | 0.21 | 1.42 | 3 | 0.2 | 1.4 | 5 | 0.42 | 2.25 | 5 | 0.39 | 2.26 | 5 | 0.38 | 2.16 | 4 | 0.27 | 1.74 |
| Scenario 7 | 10 | 0.79 | 3.09 | 9 | 0.7 | 3 | 8 | 0.48 | 2.18 | 10 | 0.74 | 3.07 | 7 | 0.46 | 2.17 | 7 | 0.46 | 2.15 | 14 | 1.14 | 3.97 | 11 | 0.95 | 3.72 | 11 | 0.94 | 3.61 | 8 | 0.57 | 2.53 |
| Scenario 8 | 28 | 3.37 | 8.88 | 26 | 2.87 | 7.84 | 22 | 2.06 | 5.94 | 27 | 3.36 | 9.02 | 22 | 1.97 | 5.84 | 21 | 1.84 | 5.3 | 34 | 5.22 | 12.56 | 29 | 4.11 | 10.97 | 30 | 4.09 | 10.58 | 23 | 2.33 | 6.71 |
| Scenario 9 | 7 | 0.53 | 2.45 | 7 | 0.51 | 2.5 | 6 | 0.36 | 1.88 | 7 | 0.51 | 2.45 | 6 | 0.36 | 1.95 | 5 | 0.31 | 1.74 | 10 | 0.76 | 3.1 | 8 | 0.7 | 3.2 | 8 | 0.64 | 2.92 | 6 | 0.42 | 2.1 |
| Scenario 10 | 33 | 15.02 | 40.42 | 33 | 16.32 | 43.21 | 31 | 12.16 | 35.26 | 34 | 17.87 | 46.91 | 31 | 11.03 | 32.35 | 29 | 9.76 | 29.33 | 36 | 23.56 | 58.12 | 34 | 22.19 | 54.9 | 34 | 21.15 | 53.8 | 31 | 12.76 | 36.25 |
| Scenario 11 | 47 | 17.67 | 42.46 | 43 | 17.23 | 42.49 | 41 | 13.76 | 36.73 | 45 | 20.46 | 49.28 | 40 | 11.92 | 32.35 | 39 | 11.07 | 30.47 | 51 | 29.63 | 63.92 | 47 | 26.11 | 58.94 | 47 | 24.45 | 55.57 | 41 | 13.68 | 35.93 |
| Scenario 12 | 15 | 1.48 | 5.16 | 15 | 1.71 | 6.2 | 13 | 1.08 | 4.29 | 15 | 1.56 | 5.54 | 12 | 1.08 | 4.34 | 12 | 0.95 | 3.76 | 19 | 2.47 | 8.02 | 17 | 2.41 | 9.01 | 17 | 2.09 | 7.11 | 14 | 1.31 | 5.07 |

Same scenarios and metrics as in Table A.7

Table A.9: Ticket Scheduling without S21sec Distribution - 35.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 6 | 0.35 | 1.88 | 8 | 0.62 | 2.76 | 5 | 0.32 | 1.8 | 8 | 0.55 | 2.53 | 9 | 0.62 | 2.65 | 7 | 0.51 | 2.46 | 7 | 0.53 | 2.57 | 7 | 0.47 | 2.29 | 7 | 0.51 | 2.47 | 7 | 0.48 | 2.33 |
| Scenario 2 | 14 | 0.93 | 3.36 | 20 | 1.7 | 5.24 | 13 | 0.88 | 3.32 | 19 | 1.57 | 4.73 | 21 | 1.73 | 4.86 | 19 | 1.52 | 4.78 | 9 | 1.55 | 4.76 | 17 | 1.33 | 4.25 | 18 | 1.42 | 4.65 | 18 | 1.34 | 4.25 |
| Scenario 3 | 3 | 0.21 | 1.44 | 5 | 0.37 | 2.15 | 3 | 0.21 | 1.45 | 4 | 0.33 | 2.01 | 5 | 0.37 | 2.07 | 4 | 0.31 | 1.93 | 4 | 0.32 | 2.03 | 4 | 0.28 | 1.79 | 4 | 0.31 | 1.94 | 4 | 0.28 | 1.81 |
| Scenario 4 | 7 | 0.42 | 2.06 | 11 | 0.92 | 3.64 | 7 | 0.44 | 2.15 | 10 | 0.73 | 2.99 | 11 | 0.89 | 3.46 | 10 | 0.74 | 3.14 | 10 | 0.84 | 3.5 | 10 | 0.76 | 3.12 | 9 | 0.68 | 2.91 | 10 | 0.72 | 3.03 |
| Scenario 5 | 17 | 1.4 | 4.62 | 27 | 3.85 | 11.89 | 19 | 1.58 | 5.05 | 25 | 2.71 | 7.81 | 28 | 3.68 | 10.87 | 25 | 3.02 | 9.25 | 26 | 3.51 | 10.42 | 25 | 3.42 | 11.05 | 24 | 2.67 | 8.2 | 24 | 2.88 | 8.87 |
| Scenario 6 | 4 | 0.23 | 1.48 | 6 | 0.44 | 2.35 | 3 | 0.23 | 1.5 | 5 | 0.35 | 2.06 | 6 | 0.43 | 2.23 | 5 | 0.36 | 2.08 | 5 | 0.38 | 2.17 | 5 | 0.35 | 1.99 | 5 | 0.35 | 2.03 | 5 | 0.33 | 1.92 |
| Scenario 7 | 9 | 0.57 | 2.5 | 15 | 1.31 | 4.72 | 10 | 0.72 | 3.08 | 14 | 1.1 | 3.96 | 15 | 1.21 | 4.14 | 14 | 1.15 | 4.21 | 13 | 1.09 | 4 | 13 | 1.1 | 4.03 | 14 | 1.13 | 4.18 | 13 | 0.96 | 3.55 |
| Scenario 8 | 26 | 2.78 | 8.09 | 37 | 6.72 | 16.96 | 28 | 3.55 | 10 | 36 | 5.58 | 13.77 | 38 | 6.08 | 14.54 | 35 | 5.84 | 15.2 | 35 | 5.45 | 13.47 | 35 | 6.01 | 16.4 | 35 | 6.19 | 16.64 | 33 | 4.46 | 11.16 |
| Scenario 9 | 6 | 0.39 | 1.97 | 10 | 0.84 | 3.49 | 7 | 0.4 | 2.02 | 9 | 0.64 | 2.77 | 10 | 0.84 | 3.4 | 10 | 0.74 | 3.21 | 10 | 0.78 | 3.28 | 10 | 0.75 | 3.2 | 9 | 0.64 | 2.81 | 9 | 0.68 | 2.96 |
| Scenario 10 | 31 | 12.48 | 35.15 | 38 | 30.71 | 68.4 | 33 | 14.57 | 39.51 | 36 | 21.47 | 52.84 | 38 | 29.63 | 66.84 | 36 | 25.78 | 60.57 | 37 | 29.08 | 66.71 | 37 | 28.24 | 64.47 | 35 | 23.23 | 56.9 | 35 | 21.97 | 54.01 |
| Scenario 11 | 44 | 14.48 | 36.23 | 54 | 37.58 | 74.1 | 48 | 18.52 | 42.98 | 53 | 27.33 | 57.78 | 54 | 35.13 | 70.72 | 52 | 31.38 | 63.91 | 51 | 33.85 | 70.51 | 51 | 34.29 | 70.79 | 53 | 30.59 | 63.34 | 49 | 25.86 | 57.04 |
| Scenario 12 | 13 | 1.13 | 4.27 | 21 | 3.27 | 11.53 | 14 | 1.15 | 4.13 | 19 | 2.07 | 6.89 | 21 | 3.05 | 10.34 | 19 | 2.46 | 8.85 | 21 | 3.14 | 10.58 | 20 | 2.93 | 10.54 | 18 | 2.02 | 7.08 | 19 | 2.52 | 8.82 |

Scenarios: 1 - Balanced Shifts with Fewer Operators; 2 - Balanced Shifts; 3 - Balanced Shifts with More Operators; 4 - Shifts with Low Imbalance and Fewer Operators; 5 - Shifts with Low Imbalance; 6 - Shifts with Low Imbalance and More Operators; 7 - Shifts with Medium Imbalance with Fewer Operators; 8 - Shifts with Medium Imbalance; 9 - Shifts with Medium Imbalance and More Operators; 10 - Shifts with High Imbalance with Fewer Operators; 11 - Shifts with High Imbalance; 12 - Shifts with High Imbalance and More Operators. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.



Table A.10: Ticket Scheduling without S21sec Distribution - 40.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 8 | 0.52 | 2.35 | 10 | 0.67 | 2.84 | 6 | 0.38 | 1.94 | 9 | 0.58 | 2.55 | 6 | 0.35 | 1.8 | 7 | 0.46 | 2.25 | 11 | 0.73 | 2.95 | 6 | 0.34 | 1.77 | 7 | 0.4 | 1.92 | 9 | 0.62 | 2.65 |
| Scenario 2 | 21 | 1.59 | 4.64 | 25 | 2.21 | 6.17 | 17 | 1.11 | 3.48 | 22 | 1.87 | 5.51 | 17 | 1.05 | 3.33 | 19 | 1.37 | 4.28 | 26 | 2.37 | 6.14 | 16 | 1.03 | 3.37 | 18 | 1.19 | 3.74 | 23 | 1.91 | 5.41 |
| Scenario 3 | 4 | 0.29 | 1.74 | 5 | 0.36 | 2.07 | 3 | 0.22 | 1.51 | 4 | 0.3 | 1.79 | 3 | 0.2 | 1.39 | 4 | 0.25 | 1.66 | 5 | 0.38 | 2.09 | 3 | 0.21 | 1.41 | 4 | 0.23 | 1.47 | 5 | 0.33 | 1.87 |
| Scenario 4 | 10 | 0.65 | 2.72 | 13 | 0.99 | 3.57 | 9 | 0.56 | 2.4 | 10 | 0.7 | 2.89 | 9 | 0.57 | 2.48 | 10 | 0.74 | 3.22 | 15 | 1.29 | 4.35 | 8 | 0.48 | 2.18 | 9 | 0.55 | 2.37 | 13 | 1.02 | 3.67 |
| Scenario 5 | 25 | 2.52 | 7.37 | 32 | 4.61 | 12.43 | 23 | 2.18 | 6.29 | 27 | 2.98 | 8.57 | 25 | 2.68 | 8.25 | 25 | 3.36 | 11.28 | 36 | 6.38 | 17.71 | 22 | 1.93 | 6.44 | 24 | 2.37 | 7.89 | 31 | 4.59 | 12.9 |
| Scenario 6 | 5 | 0.31 | 1.74 | 6 | 0.44 | 2.3 | 4 | 0.25 | 1.59 | 5 | 0.32 | 1.85 | 4 | 0.25 | 1.56 | 5 | 0.32 | 1.9 | 7 | 0.5 | 2.39 | 4 | 0.24 | 1.49 | 4 | 0.25 | 1.53 | 6 | 0.42 | 2.14 |
| Scenario 7 | 14 | 1.02 | 3.61 | 19 | 1.67 | 5.11 | 12 | 0.84 | 3.17 | 14 | 1.02 | 3.61 | 12 | 0.79 | 2.92 | 13 | 0.92 | 3.49 | 20 | 1.9 | 5.67 | 11 | 0.68 | 2.69 | 12 | 0.81 | 2.98 | 18 | 1.5 | 4.78 |
| Scenario 8 | 36 | 5.56 | 14.32 | 45 | 10.86 | 25.26 | 34 | 4.29 | 10.4 | 37 | 5.29 | 12.79 | 34 | 4.05 | 9.9 | 34 | 4.98 | 13.56 | 47 | 11.62 | 26.3 | 30 | 3.37 | 8.8 | 33 | 4.33 | 11.55 | 43 | 8.9 | 21.01 |
| Scenario 9 | 9 | 0.61 | 2.6 | 13 | 0.96 | 3.5 | 8 | 0.51 | 2.29 | 10 | 0.66 | 2.83 | 9 | 0.55 | 2.44 | 9 | 0.7 | 3.16 | 15 | 1.24 | 4.32 | 8 | 0.46 | 2.14 | 8 | 0.51 | 2.26 | 13 | 0.98 | 3.65 |
| Scenario 10 | 36 | 22.33 | 53.34 | 40 | 33.39 | 70.29 | 36 | 18.92 | 47.42 | 37 | 24.96 | 58.37 | 36 | 24.97 | 58.27 | 37 | 30.63 | 66.89 | 41 | 37.99 | 76.99 | 35 | 18.39 | 46.63 | 35 | 22.89 | 55.2 | 39 | 33.06 | 70.4 |
| Scenario 11 | 54 | 28.17 | 57.57 | 60 | 46.58 | 81.03 | 52 | 23.95 | 52.45 | 53 | 28.63 | 59.93 | 50 | 28.41 | 60.68 | 50 | 34.63 | 69.91 | 60 | 49.7 | 85.51 | 49 | 22.11 | 49.83 | 51 | 27.29 | 58.58 | 58 | 42.49 | 77.79 |
| Scenario 12 | 18 | 1.97 | 6.9 | 24 | 3.75 | 11.96 | 18 | 1.71 | 5.51 | 20 | 2.32 | 8.17 | 19 | 2.25 | 7.65 | 20 | 2.94 | 11.22 | 27 | 5.22 | 16.65 | 17 | 1.59 | 5.97 | 18 | 1.92 | 7.29 | 25 | 3.89 | 12.4 |

Same scenarios and metrics as in Table A.9

Table A.11: Ticket Scheduling with Uniform Distribution - 20.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 4 | 0.28 | 1.79 | 4 | 0.27 | 1.73 | 3 | 0.24 | 1.58 | 4 | 0.31 | 1.88 | 3 | 0.26 | 1.73 | 3 | 0.22 | 1.59 | 3 | 0.21 | 1.54 | 3 | 0.24 | 1.62 | 3 | 0.23 | 1.53 | 3 | 0.25 | 1.71 |
| Scenario 2 | 7 | 0.5 | 2.41 | 7 | 0.49 | 2.37 | 6 | 0.39 | 2.97 | 7 | 0.55 | 2.53 | 6 | 0.45 | 2.28 | 6 | 0.39 | 2.11 | 5 | 0.32 | 1.85 | 6 | 0.39 | 2.06 | 6 | 0.39 | 2.03 | 6 | 0.42 | 2.16 |
| Scenario 3 | 3 | 0.24 | 1.62 | 3 | 0.22 | 1.58 | 3 | 0.19 | 1.39 | 3 | 0.26 | 1.72 | 3 | 0.22 | 1.58 | 3 | 0.19 | 1.47 | 2 | 0.16 | 1.3 | 3 | 0.21 | 1.5 | 3 | 0.2 | 1.44 | 3 | 0.21 | 1.5 |
| Scenario 4 | 4 | 0.33 | 1.93 | 5 | 0.34 | 1.95 | 4 | 0.27 | 1.67 | 5 | 0.41 | 2.23 | 4 | 0.32 | 1.93 | 4 | 0.28 | 1.79 | 3 | 0.24 | 1.62 | 4 | 0.27 | 1.72 | 4 | 0.27 | 1.66 | 4 | 0.29 | 1.8 |
| Scenario 5 | 10 | 0.86 | 3.53 | 10 | 0.92 | 3.75 | 9 | 0.69 | 2.94 | 11 | 1.06 | 4.01 | 10 | 0.85 | 3.46 | 9 | 0.69 | 2.95 | 8 | 0.58 | 2.8 | 9 | 0.7 | 3.05 | 9 | 0.68 | 2.92 | 9 | 0.76 | 3.24 |
| Scenario 6 | 3 | 0.25 | 1.68 | 3 | 0.23 | 1.58 | 3 | 0.21 | 1.51 | 4 | 0.28 | 1.79 | 3 | 0.23 | 1.61 | 3 | 0.21 | 1.54 | 3 | 0.18 | 1.38 | 3 | 0.21 | 1.45 | 3 | 0.2 | 1.42 | 3 | 0.22 | 1.54 |
| Scenario 7 | 5 | 0.4 | 2.13 | 5 | 0.39 | 2.06 | 5 | 0.32 | 1.88 | 6 | 0.46 | 2.31 | 5 | 0.37 | 2.07 | 5 | 0.31 | 1.8 | 4 | 0.29 | 1.79 | 5 | 0.34 | 1.94 | 5 | 0.3 | 1.68 | 5 | 0.35 | 1.97 |
| Scenario 8 | 15 | 1.35 | 4.51 | 14 | 1.26 | 4.37 | 13 | 1.07 | 3.79 | 16 | 1.53 | 4.93 | 14 | 1.27 | 4.36 | 13 | 1.13 | 3.97 | 11 | 0.88 | 3.47 | 13 | 1.14 | 4.02 | 13 | 1.1 | 3.95 | 14 | 1.16 | 4.02 |
| Scenario 9 | 4 | 0.33 | 1.94 | 4 | 0.33 | 1.93 | 4 | 0.25 | 1.61 | 5 | 0.38 | 2.1 | 4 | 0.3 | 1.81 | 4 | 0.25 | 1.67 | 3 | 0.22 | 1.53 | 4 | 0.27 | 1.69 | 4 | 0.25 | 1.58 | 4 | 0.28 | 1.78 |
| Scenario 10 | 25 | 6.79 | 21.46 | 25 | 6.44 | 20.34 | 24 | 5.59 | 17.77 | 26 | 8.18 | 25.29 | 25 | 6.33 | 20.5 | 23 | 5.42 | 17.74 | 21 | 4.49 | 16.17 | 23 | 5.17 | 17.02 | 24 | 5.42 | 17.41 | 25 | 6.46 | 20.51 |
| Scenario 11 | 3 | 7.45 | 21.95 | 30 | 7.13 | 20.95 | 28 | 5.69 | 16.74 | 32 | 9.15 | 26.39 | 30 | 6.98 | 21.46 | 28 | 5.81 | 17.12 | 26 | 4.96 | 16.49 | 28 | 5.97 | 18.04 | 29 | 5.82 | 17.51 | 30 | 6.64 | 19.6 |
| Scenario 12 | 9 | 0.75 | 3.24 | 9 | 0.85 | 3.8 | 8 | 0.64 | 2.86 | 10 | 0.95 | 3.83 | 9 | 0.71 | 3.12 | 8 | 0.62 | 2.85 | 7 | 0.52 | 2.61 | 8 | 0.62 | 2.9 | 8 | 0.59 | 2.75 | 8 | 0.69 | 3.06 |

Scenarios: 1 - Balanced Shifts with Fewer Operators; 2 - Balanced Shifts; 3 - Balanced Shifts with More Operators; 4 - Shifts with Low Imbalance and Fewer Operators; 5 - Shifts with Low Imbalance; 6 - Shifts with Low Imbalance and More Operators; 7 - Shifts with Medium Imbalance with Fewer Operators; 8 - Shifts with Medium Imbalance; 9 - Shifts with Medium Imbalance and More Operators; 10 - Shifts with High Imbalance with Fewer Operators; 11 - Shifts with High Imbalance; 12 - Shifts with High Imbalance and More Operators. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.

Table A.12: Ticket Scheduling with Uniform Distribution - 25.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 5 | 0.37 | 2.12 | 4 | 0.34 | 2 | 4 | 0.23 | 1.53 | 4 | 0.28 | 1.74 | 3 | 0.23 | 1.53 | 4 | 0.24 | 1.56 | 5 | 0.38 | 2.1 | 4 | 0.3 | 1.81 | 4 | 0.26 | 1.64 | 3 | 0.19 | 1.32 |
| Scenario 2 | 10 | 0.77 | 3.09 | 9 | 0.72 | 2.99 | 7 | 0.46 | 2.15 | 8 | 0.54 | 2.43 | 7 | 0.45 | 2.14 | 8 | 0.49 | 2.28 | 11 | 0.83 | 3.21 | 9 | 0.66 | 2.76 | 8 | 0.52 | 2.31 | 6 | 0.37 | 1.82 |
| Scenario 3 | 3 | 0.29 | 1.87 | 3 | 0.26 | 1.77 | 3 | 0.18 | 1.34 | 3 | 0.21 | 1.49 | 3 | 0.19 | 1.37 | 3 | 0.2 | 1.43 | 3 | 0.27 | 1.76 | 3 | 0.22 | 1.54 | 3 | 0.21 | 1.47 | 2 | 0.15 | 1.13 |
| Scenario 4 | 6 | 0.46 | 2.36 | 6 | 0.42 | 2.24 | 4 | 0.28 | 1.64 | 5 | 0.32 | 1.85 | 5 | 0.31 | 1.77 | 5 | 0.31 | 1.77 | 7 | 0.54 | 2.67 | 5 | 0.37 | 2.02 | 5 | 0.33 | 1.85 | 4 | 0.23 | 1.43 |
| Scenario 5 | 15 | 1.38 | 4.82 | 14 | 1.26 | 4.48 | 11 | 0.83 | 3.18 | 12 | 1.02 | 3.83 | 12 | 0.9 | 3.54 | 12 | 0.94 | 3.5 | 17 | 1.72 | 5.83 | 13 | 1.17 | 4.3 | 12 | 0.96 | 3.61 | 10 | 0.69 | 2.81 |
| Scenario 6 | 4 | 0.3 | 1.91 | 3 | 0.27 | 1.78 | 3 | 0.19 | 1.36 | 3 | 0.22 | 1.53 | 3 | 0.19 | 1.35 | 3 | 0.21 | 1.45 | 4 | 0.31 | 1.9 | 3 | 0.24 | 1.6 | 3 | 0.23 | 1.59 | 3 | 0.16 | 1.16 |
| Scenario 7 | 7 | 0.58 | 2.65 | 7 | 0.53 | 2.57 | 6 | 0.36 | 1.85 | 6 | 0.39 | 2.01 | 6 | 0.36 | 1.89 | 6 | 0.4 | 2.07 | 9 | 0.67 | 2.91 | 7 | 0.49 | 2.35 | 6 | 0.4 | 2.05 | 5 | 0.28 | 1.54 |
| Scenario 8 | 21 | 2.23 | 6.4 | 20 | 2.01 | 6.1 | 16 | 1.34 | 4.33 | 17 | 1.54 | 4.91 | 17 | 1.42 | 4.53 | 17 | 1.42 | 4.29 | 22 | 2.56 | 7.26 | 19 | 1.78 | 5.31 | 17 | 1.52 | 4.67 | 15 | 1.05 | 3.49 |
| Scenario 9 | 6 | 0.45 | 2.36 | 5 | 0.39 | 2.15 | 4 | 0.27 | 1.61 | 5 | 0.32 | 1.85 | 4 | 0.28 | 1.69 | 5 | 0.31 | 1.78 | 7 | 0.51 | 2.6 | 5 | 0.37 | 2.05 | 5 | 0.31 | 1.83 | 4 | 0.23 | 1.42 |
| Scenario 10 | 30 | 11.85 | 34.38 | 29 | 10.43 | 30.07 | 27 | 7.18 | 21.78 | 28 | 9.19 | 28.09 | 27 | 7.54 | 23.44 | 27 | 7.66 | 23.87 | 31 | 13.21 | 37.09 | 28 | 9.58 | 28.42 | 28 | 7.77 | 23.82 | 25 | 5.76 | 18.69 |
| Scenario 11 | 39 | 13.99 | 36.83 | 38 | 12.22 | 32.24 | 33 | 7.78 | 21.99 | 34 | 9.92 | 28.39 | 34 | 8.05 | 23.49 | 35 | 9.34 | 27.32 | 41 | 14.96 | 37.89 | 37 | 11.37 | 20.88 | 35 | 8.8 | 24.75 | 31 | 6.03 | 18.34 |
| Scenario 12 | 12 | 1.17 | 4.4 | 11 | 1.04 | 4.01 | 10 | 0.75 | 3.09 | 10 | 0.89 | 3.73 | 10 | 0.82 | 3.46 | 10 | 0.82 | 3.37 | 13 | 1.42 | 5.39 | 11 | 1 | 3.97 | 10 | 0.83 | 3.41 | 8 | 0.59 | 2.6 |

Same scenarios and metrics as in Table A.11

Table A.13: Ticket Scheduling with Uniform Distribution - 30.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 6 | 0.47 | 2.34 | 5 | 0.38 | 2.12 | 4 | 0.25 | 1.52 | 5 | 0.37 | 2.01 | 5 | 0.29 | 1.67 | 4 | 0.25 | 1.6 | 8 | 0.6 | 2.71 | 7 | 0.57 | 2.76 | 7 | 0.51 | 2.49 | 5 | 0.37 | 2.04 |
| Scenario 2 | 14 | 1.12 | 3.84 | 13 | 0.96 | 3.49 | 9 | 0.56 | 2.32 | 13 | 0.95 | 3.4 | 10 | 0.64 | 2.57 | 10 | 0.61 | 2.55 | 19 | 1.61 | 4.86 | 16 | 1.42 | 4.58 | 16 | 1.29 | 4.19 | 13 | 0.9 | 3.26 |
| Scenario 3 | 4 | 0.33 | 1.96 | 3 | 0.28 | 1.85 | 3 | 0.19 | 1.33 | 4 | 0.27 | 1.74 | 3 | 0.21 | 1.45 | 3 | 0.18 | 1.3 | 5 | 0.38 | 2.21 | 4 | 0.35 | 2.16 | 4 | 0.35 | 2.07 | 3 | 0.26 | 1.75 |
| Scenario 4 | 8 | 0.65 | 2.92 | 7 | 0.53 | 2.51 | 5 | 0.34 | 1.81 | 7 | 0.52 | 2.44 | 6 | 0.4 | 2.11 | 6 | 0.34 | 1.86 | 10 | 0.86 | 3.46 | 9 | 0.78 | 3.32 | 9 | 0.7 | 3.1 | 7 | 0.51 | 2.44 |
| Scenario 5 | 20 | 2.22 | 6.86 | 18 | 1.84 | 5.96 | 14 | 1.15 | 4.14 | 19 | 1.83 | 5.86 | 16 | 1.41 | 5.28 | 15 | 1.21 | 4.38 | 25 | 3.17 | 8.82 | 23 | 2.84 | 8.95 | 22 | 2.51 | 7.91 | 18 | 1.78 | 5.98 |
| Scenario 6 | 4 | 0.35 | 2.03 | 4 | 0.29 | 1.89 | 3 | 0.2 | 1.34 | 4 | 0.29 | 1.76 | 3 | 0.22 | 1.45 | 3 | 0.2 | 1.38 | 5 | 0.43 | 2.3 | 5 | 0.39 | 2.24 | 5 | 0.38 | 2.19 | 4 | 0.29 | 1.83 |
| Scenario 7 | 11 | 0.84 | 3.27 | 10 | 0.7 | 2.94 | 7 | 0.45 | 2.12 | 9 | 0.68 | 2.9 | 8 | 0.5 | 2.32 | 7 | 0.44 | 2.1 | 14 | 1.16 | 4.1 | 12 | 1.03 | 3.81 | 11 | 0.93 | 3.59 | 9 | 0.67 | 2.82 |
| Scenario 8 | 28 | 3.39 | 8.87 | 26 | 2.92 | 7.88 | 21 | 1.88 | 5.38 | 26 | 3.13 | 8.82 | 22 | 2.13 | 6.32 | 21 | 1.91 | 5.64 | 35 | 5.41 | 12.79 | 31 | 4.84 | 12.81 | 30 | 4.04 | 10.44 | 26 | 2.86 | 7.84 |
| Scenario 9 | 8 | 0.6 | 2.72 | 7 | 0.5 | 2.43 | 5 | 0.32 | 1.77 | 7 | 0.48 | 2.31 | 6 | 0.37 | 2.01 | 5 | 0.33 | 1.79 | 10 | 0.82 | 3.35 | 9 | 0.74 | 3.26 | 8 | 0.68 | 3.05 | 7 | 0.49 | 2.37 |
| Scenario 10 | 35 | 19.1 | 48.78 | 33 | 15.93 | 42.82 | 30 | 10.11 | 29.68 | 33 | 17.6 | 46.3 | 31 | 13.05 | 37.57 | 30 | 10.62 | 31.07 | 36 | 23.42 | 57.68 | 35 | 22.76 | 56.66 | 34 | 22.05 | 55.89 | 32 | 15.69 | 43.1 |
| Scenario 11 | 47 | 21.9 | 51.31 | 44 | 17.63 | 43.53 | 39 | 11.65 | 31.57 | 44 | 19.68 | 47.57 | 41 | 14.33 | 38.72 | 39 | 11.59 | 31.4 | 51 | 28.33 | 61.79 | 49 | 26.31 | 59.2 | 47 | 25.09 | 57.93 | 43 | 18.03 | 45.44 |
| Scenario 12 | 17 | 1.9 | 6.49 | 15 | 1.6 | 5.75 | 12 | 0.99 | 3.92 | 15 | 1.54 | 5.48 | 13 | 1.24 | 5.28 | 12 | 1.03 | 4.13 | 19 | 2.56 | 8.02 | 18 | 2.5 | 9.13 | 17 | 2.06 | 7.33 | 15 | 1.54 | 5.7 |

Scenarios: 1 - Balanced Shifts with Fewer Operators; 2 - Balanced Shifts; 3 - Balanced Shifts with More Operators; 4 - Shifts with Low Imbalance and Fewer Operators; 5 - Shifts with Low Imbalance; 6 - Shifts with Low Imbalance and More Operators; 7 - Shifts with Medium Imbalance with Fewer Operators; 8 - Shifts with Medium Imbalance; 9 - Shifts with Medium Imbalance and More Operators; 10 - Shifts with High Imbalance with Fewer Operators; 11 - Shifts with High Imbalance; 12 - Shifts with High Imbalance and More Operators. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.



Table A.14: Ticket Scheduling with Uniform Distribution - 35.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 6 | 0.35 | 1.89 | 8 | 0.58 | 2.63 | 5 | 0.3 | 1.75 | 7 | 0.5 | 2.4 | 8 | 0.6 | 2.56 | 7 | 0.49 | 2.41 | 8 | 0.54 | 2.55 | 7 | 0.46 | 2.25 | 7 | 0.5 | 2.46 | 7 | 0.47 | 2.26 |
| Scenario 2 | 14 | 0.95 | 3.38 | 20 | 1.66 | 4.92 | 12 | 0.78 | 2.9 | 19 | 1.58 | 4.81 | 21 | 1.71 | 4.9 | 18 | 1.39 | 4.42 | 19 | 1.54 | 4.61 | 16 | 1.23 | 4.05 | 17 | 1.37 | 4.38 | 18 | 1.34 | 4.15 |
| Scenario 3 | 3 | 0.22 | 1.48 | 4 | 0.34 | 2.07 | 3 | 0.19 | 1.39 | 4 | 0.32 | 1.98 | 5 | 0.35 | 1.97 | 4 | 0.29 | 1.86 | 4 | 0.31 | 1.96 | 4 | 0.28 | 1.74 | 4 | 0.33 | 2.05 | 4 | 0.29 | 1.81 |
| Scenario 4 | 8 | 0.52 | 2.39 | 10 | 0.82 | 3.37 | 7 | 0.42 | 2.14 | 10 | 0.82 | 3.41 | 11 | 0.89 | 3.52 | 10 | 0.75 | 3.36 | 10 | 0.78 | 3.22 | 9 | 0.68 | 2.93 | 10 | 0.77 | 3.29 | 10 | 0.71 | 3.02 |
| Scenario 5 | 19 | 1.88 | 6.23 | 26 | 3.44 | 10.43 | 18 | 1.57 | 5.18 | 26 | 3.47 | 10.95 | 28 | 3.66 | 11.17 | 24 | 2.98 | 9.59 | 26 | 3.21 | 9.75 | 23 | 2.62 | 8.34 | 24 | 3.03 | 10.07 | 24 | 2.79 | 8.79 |
| Scenario 6 | 4 | 0.26 | 1.6 | 5 | 0.39 | 2.18 | 3 | 0.22 | 1.49 | 5 | 0.36 | 2.1 | 6 | 0.41 | 2.17 | 5 | 0.36 | 2.09 | 5 | 0.36 | 2.12 | 5 | 0.33 | 1.9 | 5 | 0.39 | 2.2 | 5 | 0.34 | 1.95 |
| Scenario 7 | 10 | 0.69 | 2.95 | 14 | 1.14 | 4.16 | 9 | 0.56 | 2.45 | 14 | 1.12 | 4.16 | 15 | 1.24 | 4.14 | 13 | 1 | 3.83 | 14 | 1.09 | 3.92 | 12 | 0.9 | 3.43 | 13 | 1.05 | 3.91 | 13 | 1 | 3.64 |
| Scenario 8 | 28 | 3.14 | 8.66 | 35 | 5.46 | 14.12 | 26 | 2.67 | 7.33 | 36 | 5.76 | 14.99 | 38 | 6.31 | 15.66 | 34 | 4.94 | 13.09 | 35 | 5.39 | 13.58 | 31 | 4.15 | 11.14 | 33 | 4.78 | 12.5 | 33 | 4.49 | 11.29 |
| Scenario 9 | 8 | 0.51 | 2.42 | 10 | 0.79 | 3.38 | 6 | 0.39 | 2.01 | 10 | 0.78 | 3.32 | 11 | 0.83 | 3.37 | 9 | 0.73 | 3.35 | 9 | 0.72 | 3.14 | 9 | 0.63 | 2.84 | 9 | 0.72 | 3.14 | 9 | 0.69 | 2.96 |
| Scenario 10 | 34 | 18.08 | 47.4 | 37 | 26.87 | 62.87 | 33 | 15.5 | 41.7 | 37 | 27.64 | 64.42 | 38 | 27.98 | 65.35 | 36 | 24.3 | 59.01 | 37 | 24.63 | 59.19 | 36 | 23.42 | 57.5 | 36 | 25.88 | 61.61 | 36 | 22.44 | 55.5 |
| Scenario 11 | 45 | 20.29 | 48.49 | 52 | 32.35 | 67.55 | 44 | 17.81 | 44.53 | 52 | 34.58 | 72.2 | 54 | 34.63 | 71.46 | 50 | 29.14 | 63.67 | 52 | 29.15 | 62.54 | 49 | 26.99 | 60.39 | 50 | 29.13 | 63.46 | 50 | 25.61 | 57.65 |
| Scenario 12 | 16 | 1.66 | 6.23 | 21 | 3.07 | 10.53 | 14 | 1.3 | 4.75 | 20 | 3 | 10.41 | 21 | 2.99 | 10.43 | 19 | 2.43 | 8.89 | 20 | 2.62 | 8.85 | 18 | 2.29 | 8.45 | 19 | 2.61 | 9.24 | 19 | 2.4 | 8.55 |

Same scenarios and metrics as in Table A.13

Table A.15: Ticket Scheduling with Uniform Distribution - 40.000 tickets

| Distribution Scenarios | Run 1 | | | Run 2 | | | Run 3 | | | Run 4 | | | Run 5 | | | Run 6 | | | Run 7 | | | Run 8 | | | Run 9 | | | Run 10 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD | % | AVG WT | WT SD |
| Scenario 1 | 8 | 0.5 | 2.28 | 9 | 0.62 | 2.75 | 6 | 0.38 | 1.95 | 9 | 0.59 | 2.56 | 6 | 0.35 | 1.85 | 7 | 0.44 | 2.2 | 11 | 0.75 | 3.03 | 6 | 0.34 | 1.8 | 7 | 0.4 | 1.97 | 9 | 0.59 | 2.53 |
| Scenario 2 | 20 | 1.52 | 4.46 | 23 | 1.88 | 5.23 | 17 | 1.11 | 3.55 | 23 | 1.85 | 5.25 | 17 | 1.08 | 3.5 | 18 | 1.35 | 4.25 | 26 | 2.27 | 5.96 | 16 | 0.98 | 3.24 | 17 | 1.17 | 3.74 | 22 | 1.73 | 4.82 |
| Scenario 3 | 4 | 0.27 | 1.68 | 5 | 0.33 | 2 | 3 | 0.24 | 1.58 | 5 | 0.32 | 1.9 | 3 | 0.21 | 1.44 | 4 | 0.26 | 1.74 | 5 | 0.38 | 2.1 | 3 | 0.21 | 1.37 | 4 | 0.23 | 1.47 | 5 | 0.32 | 1.85 |
| Scenario 4 | 11 | 0.77 | 3 | 12 | 0.89 | 3.45 | 9 | 0.57 | 2.43 | 12 | 0.9 | 3.56 | 9 | 0.53 | 2.4 | 10 | 0.67 | 2.91 | 14 | 1.1 | 3.99 | 9 | 0.53 | 2.37 | 9 | 0.58 | 2.46 | 12 | 0.89 | 3.37 |
| Scenario 5 | 27 | 3.22 | 9.74 | 30 | 3.82 | 10.51 | 24 | 2.33 | 6.67 | 30 | 4.07 | 12.28 | 23 | 2.22 | 6.84 | 25 | 2.82 | 8.86 | 33 | 5.06 | 14.38 | 23 | 2.28 | 7.39 | 25 | 2.65 | 8.85 | 29 | 3.89 | 11.2 |
| Scenario 6 | 5 | 0.33 | 1.84 | 5 | 0.38 | 2.1 | 4 | 0.26 | 1.63 | 6 | 0.39 | 2.12 | 4 | 0.25 | 1.55 | 5 | 0.31 | 1.89 | 6 | 0.45 | 2.26 | 4 | 0.25 | 1.51 | 5 | 0.28 | 1.63 | 6 | 0.37 | 1.99 |
| Scenario 7 | 14 | 1.05 | 3.68 | 16 | 1.23 | 4.11 | 12 | 0.81 | 3.03 | 16 | 1.25 | 4.34 | 12 | 0.77 | 2.98 | 13 | 0.91 | 3.42 | 18 | 1.59 | 4.96 | 12 | 0.74 | 2.85 | 13 | 0.84 | 3.13 | 16 | 1.24 | 4.12 |
| Scenario 8 | 37 | 5.57 | 13.75 | 40 | 6.81 | 16.5 | 33 | 3.73 | 9.19 | 40 | 6.78 | 16.56 | 32 | 3.74 | 9.49 | 34 | 4.68 | 12.09 | 44 | 9.12 | 21.83 | 31 | 3.55 | 9.46 | 34 | 4.45 | 12.16 | 40 | 6.56 | 15.86 |
| Scenario 9 | 10 | 0.74 | 3.03 | 11 | 0.85 | 3.35 | 8 | 0.53 | 2.33 | 11 | 0.85 | 3.46 | 8 | 0.5 | 2.29 | 9 | 0.63 | 2.78 | 13 | 1.07 | 4.05 | 8 | 0.49 | 2.22 | 9 | 0.56 | 2.4 | 12 | 0.85 | 3.32 |
| Scenario 10 | 37 | 26.86 | 62.3 | 38 | 30.4 | 68.26 | 36 | 20.4 | 50.66 | 38 | 30.46 | 68.66 | 36 | 21.18 | 52.2 | 36 | 24.68 | 58.42 | 40 | 34.33 | 74.18 | 36 | 21.96 | 53.79 | 36 | 25.35 | 60.15 | 38 | 29.79 | 67.02 |
| Scenario 11 | 53 | 32.53 | 67.29 | 55 | 35.47 | 71.71 | 50 | 23.66 | 53.23 | 55 | 37.59 | 74.96 | 49 | 23.58 | 53.43 | 51 | 29 | 61.87 | 59 | 44.43 | 82.64 | 49 | 25.1 | 56.6 | 50 | 28.21 | 62.13 | 55 | 35.94 | 71.84 |
| Scenario 12 | 21 | 2.72 | 9.12 | 22 | 3.23 | 10.37 | 18 | 1.85 | 6.05 | 22 | 3.31 | 11.3 | 18 | 1.93 | 6.78 | 19 | 2.34 | 8.3 | 25 | 4.11 | 13.44 | 18 | 1.89 | 6.95 | 19 | 2.25 | 8.38 | 22 | 3.12 | 10.63 |

Scenarios: 1 - Balanced Shifts with Fewer Operators; 2 - Balanced Shifts; 3 - Balanced Shifts with More Operators; 4 - Shifts with Low Imbalance and Fewer Operators; 5 - Shifts with Low Imbalance; 6 - Shifts with Low Imbalance and More Operators; 7 - Shifts with Medium Imbalance with Fewer Operators; 8 - Shifts with Medium Imbalance; 9 - Shifts with Medium Imbalance and More Operators; 10 - Shifts with High Imbalance with Fewer Operators; 11 - Shifts with High Imbalance; 12 - Shifts with High Imbalance and More Operators. Metrics: % - Percentage of scheduled tickets for later treatment; AVG - Average; WT - Wait time; SD - Standard deviation.

## A.2 CROSS Performance

Table A.16: Action/Step TAU Analysis (all seeds)

| TAU | Run 1 (Seed 2) | | | | Run 2 (Seed 3) | | | | Run 3 (Seed 4) | | | | Run 4 (Seed 5) | | | | Run 5 (Seed 6) | | | | AVG | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D | A | B | C | D |
| 20 | 0.050 | 0.01 | 0.004 | 0.015 | 0.051 | 0.011 | 0.004 | 0.017 | 0.048 | 0.009 | 0.004 | 0.017 | 0.049 | 0.008 | 0.004 | 0.017 | 0.051 | 0.011 | 0.004 | 0.016 | 0.05 | 0.01 | 0.004 | 0.016 |
| 40 | 0.086 | 0.023 | 0.003 | 0.017 | 0.077 | 0.02 | 0.002 | 0.013 | 0.084 | 0.022 | 0.003 | 0.017 | 0.084 | 0.021 | 0.003 | 0.017 | 0.08 | 0.021 | 0.003 | 0.017 | 0.082 | 0.021 | 0.003 | 0.016 |
| 60 | 0.12 | 0.04 | 0.004 | 0.02 | 0.106 | 0.036 | 0.004 | 0.019 | 0.12 | 0.04 | 0.003 | 0.018 | 0.12 | 0.04 | 0.004 | 0.021 | 0.12 | 0.050 | 0.004 | 0.022 | 0.117 | 0.041 | 0.004 | 0.02 |
| 80 | 0.158 | 0.064 | 0.004 | 0.019 | 0.132 | 0.054 | 0.004 | 0.018 | 0.183 | 0.093 | 0.005 | 0.027 | 0.206 | 0.094 | 0.005 | 0.027 | 0.179 | 0.111 | 0.005 | 0.026 | 0.171 | 0.083 | 0.004 | 0.023 |
| 100 | 0.161 | 0.074 | 0.003 | 0.017 | 0.181 | 0.094 | 0.004 | 0.206 | 0.154 | 0.071 | 0.003 | 0.181 | 0.154 | 0.07 | 0.003 | 0.018 | 0.155 | 0.071 | 0.003 | 0.017 | 0.161 | 0.076 | 0.003 | 0.088 |
| 120 | 0.186 | 0.098 | 0.003 | 0.018 | 0.177 | 0.091 | 0.004 | 0.02 | 0.177 | 0.09 | 0.002 | 0.15 | 0.178 | 0.09 | 0.003 | 0.018 | 0.177 | 0.091 | 0.003 | 0.014 | 0.179 | 0.092 | 0.003 | 0.044 |
| 140 | 0.199 | 0.11 | 0.003 | 0.018 | 0.201 | 0.112 | 0.003 | 0.018 | 0.203 | 0.112 | 0.003 | 0.017 | 0.199 | 0.11 | 0.004 | 0.022 | 0.2 | 0.11 | 0.003 | 0.019 | 0.2 | 0.111 | 0.003 | 0.019 |
| 160 | 0.22 | 0.132 | 0.004 | 0.214 | 0.22 | 0.131 | 0.003 | 0.13 | 0.22 | 0.132 | 0.003 | 0.019 | 0.221 | 0.133 | 0.004 | 0.02 | 0.22 | 0.132 | 0.004 | 0.021 | 0.22 | 0.132 | 0.003 | 0.081 |
| 180 | 0.295 | 0.194 | 0.004 | 0.023 | 0.314 | 0.221 | 0.005 | 0.031 | 0.403 | 0.293 | 0.004 | 0.029 | 0.467 | 0.441 | 0.005 | 0.029 | 0.271 | 0.175 | 0.004 | 0.022 | 0.35 | 0.265 | 0.004 | 0.027 |
| 200 | 0.311 | 0.219 | 0.004 | 0.021 | 0.338 | 0.245 | 0.005 | 0.027 | 0.291 | 0.212 | 0.003 | 0.019 | 0.39 | 0.31 | 0.007 | 0.042 | 0.296 | 0.208 | 0.004 | 0.024 | 0.325 | 0.239 | 0.004 | 0.027 |
| 220 | 0.323 | 0.24 | 0.004 | 0.02 | 0.32 | 0.239 | 0.005 | 0.026 | 0.325 | 0.248 | 0.004 | 0.025 | 0.326 | 0.244 | 0.005 | 0.028 | 0.368 | 0.29 | 0.005 | 0.033 | 0.332 | 0.252 | 0.004 | 0.026 |
| 240 | 0.326 | 0.26 | 0.004 | 0.023 | 0.315 | 0.249 | 0.004 | 0.022 | 0.4 | 0.323 | 0.005 | 0.031 | 0.391 | 0.31 | 0.007 | 0.039 | 0.393 | 0.312 | 0.004 | 0.023 | 0.365 | 0.291 | 0.005 | 0.028 |

A - Average recommendation time; B - Recommendation time standard deviation; C - Average readaptation time; D - Readaptation time standard deviation time. All meatrics are measured in seconds.

Table A.17: Scalability Assessment

| Scenarios | Run 1 (Seed 2) | | | | | Run 2 (Seed 3) | | | | | Run 3 (Seed 4) | | | | | Run 4 (Seed 5) | | | | | Run 5 (Seed 6) | | | | | AVG | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | A | B | C | D | E | A | B | C | D | E | A | B | C | D | E | A | B | C | D | E | A | B | C | D | E |
| 6.000 tickets with 12 operators | 0.125 | 0.076 | 0.004 | 0.022 | 165.195 | 0.139 | 0.089 | 0.005 | 0.262 | 164.937 | 0.128 | 0.076 | 0.004 | 0.023 | 164.876 | 0.125 | 0.077 | 0.003 | 0.016 | 164.717 | 0.117 | 0.073 | 0.005 | 0.023 | 165.062 | 0.127 | 0.078 | 0.004 | 0.069 | 164.957 |
| 12.000 tickets with 12 operators | 0.113 | 0.065 | 0.004 | 0.019 | 196.15 | 0.118 | 0.068 | 0.005 | 0.024 | 196.11 | 0.116 | 0.067 | 0.004 | 0.02 | 196.167 | 0.113 | 0.064 | 0.003 | 0.018 | 196.059 | 0.111 | 0.063 | 0.005 | 0.024 | 196.025 | 0.114 | 0.065 | 0.004 | 0.021 | 196.102 |
| 12.000 tickets with 15 operators | 0.206 | 0.135 | 0.004 | 0.023 | 209.138 | 0.167 | 0.108 | 0.004 | 0.021 | 208.9 | 0.161 | 0.105 | 0.004 | 0.018 | 208.873 | 0.155 | 0.1 | 0.004 | 0.021 | 208.857 | 0.147 | 0.094 | 0.003 | 0.019 | 208.67 | 0.167 | 0.108 | 0.004 | 0.021 | 208.888 |
| 12.000 tickets with 18 operators | 0.169 | 0.108 | 0.004 | 0.022 | 222.474 | 0.169 | 0.108 | 0.004 | 0.025 | 222.022 | 0.193 | 0.129 | 0.006 | 0.031 | 222.14 | 0.187 | 0.12 | 0.005 | 0.026 | 222.125 | 0.183 | 0.117 | 0.005 | 0.028 | 221.969 | 0.18 | 0.116 | 0.005 | 0.026 | 222.146 |
| 12.000 tickets with 21 operators | 0.23 | 0.15 | 0.006 | 0.028 | 237.303 | 0.219 | 0.141 | 0.006 | 0.029 | 236.623 | 0.22 | 0.141 | 0.006 | 0.028 | 236.939 | 0.207 | 0.134 | 0.005 | 0.025 | 236.894 | 0.202 | 0.129 | 0.004 | 0.019 | 236.94 | 0.216 | 0.139 | 0.005 | 0.026 | 236.94 |
| 18.000 tickets with 12 operators | 0.121 | 0.069 | 0.004 | 0.02 | 213.756 | 0.109 | 0.061 | 0.005 | 0.023 | 213.44 | 0.108 | 0.06 | 0.004 | 0.019 | 213.566 | 0.109 | 0.061 | 0.003 | 0.017 | 213.671 | 0.114 | 0.064 | 0.005 | 0.023 | 213.644 | 0.112 | 0.063 | 0.004 | 0.02 | 213.615 |
| 24.000 tickets with 12 operators | 0.11 | 0.061 | 0.004 | 0.019 | 254.036 | 0.11 | 0.06 | 0.005 | 0.022 | 253.439 | 0.117 | 0.066 | 0.004 | 0.022 | 253.278 | 0.116 | 0.064 | 0.003 | 0.019 | 253.318 | 0.119 | 0.067 | 0.005 | 0.024 | 253.23 | 0.114 | 0.064 | 0.004 | 0.021 | 253.46 |

A - Average recommendation time; B - Recommendation time standard deviation; C - Average readaptation time; D - Readaptation time standard deviation time; E - Memory usage (MegaBytes). All meatrics, except the last, are measured in seconds.

Table A.18: Treatment with CROSS (all seeds)

| | **Seed 2** | | | | | | Seed 3 | | | | | | Seed 4 | | | | | | Seed 5 | | | | | | Seed 6 | | | | | | AVG | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ROAR | F | G | H | I | J | K | F | G | H | I | J | K | F | G | H | I | J | K | F | G | H | I | J | K | F | G | H | I | J | K | F | G | H | I | J | K |
| 100 | 6.25 | 52.79 | 9.92 | 2.91 | 0: 2916 | 2022-W35: 0.061, 2022-W36: -0.069, 2022-W37: 0.437, 2022-W38: 0.084, 2022-W39: 0.358, 2022-W40: -0.039, 2022-W41: -0.038, 2022-W42: 0.315, 2022-W43: 0.063, 2022-W44: 0.074, 2022-W45: 0.062, 2022-W46: 0.11, 2022-W47: -0.088, 2022-W48: 0.416, 2022-W49: -0.135, 2022-W50: 0.0, 2022-W51: 0.158, 2022-W52: 0.124, 2023-W1: -0.128, 2023-W2: 0.244, 2023-W3: -0.114, 2023-W4: -0.017 | 6.47 | 54.04 | 10.04 | 3.07 | 0: 2916 | 2022-W35: 0.643, 2022-W36: 0.421, 2022-W37: 0.216, 2022-W38: 0.168, 2022-W39: 0.004, 2022-W40: 0.082, 2022-W41: -0.083, 2022-W42: 0.161, 2022-W43: 0.281, 2022-W44: 0.331, 2022-W45: 0.202, 2022-W46: 0.125, 2022-W47: -0.133, 2022-W48: 0.16, 2022-W49: 0.159, 2022-W50: 0.112, 2022-W51: -0.007, 2022-W52: -0.031, 2023-W1: 0.174, 2023-W2: 0.236, 2023-W3: -0.271, 2023-W4: 0.039 | 5.47 | 54.04 | 9.95 | 2.92 | 0: 2916 | 2022-W35: 0.411, 2022-W36: 0.14, 2022-W37: -0.077, 2022-W38: 0.238, 2022-W39: 0.144, 2022-W40: -0.032, 2022-W41: 0.131, 2022-W42: 0.089, 2022-W43: 0.049, 2022-W44: 0.026, 2022-W45: 0.045, 2022-W46: 0.016, 2022-W47: 0.167, 2022-W48: -0.08, 2022-W49: -0.003, 2022-W50: -0.087, 2022-W51: 0.174, 2022-W52: -0.035, 2023-W1: -0.043, 2023-W2: 0.021, 2023-W3: 0.057, 2023-W4: 0.172 | 6.47 | 54.04 | 10.04 | 2.98 | 0: 2916 | 2022-W35: 0.205, 2022-W36: -0.158, 2022-W37: 0.143, 2022-W38: 0.354, 2022-W39: 0.049, 2022-W40: 0.234, 2022-W41: 0.342, 2022-W42: 0.145, 2022-W43: 0.103, 2022-W44: 0.095, 2022-W45: -0.058, 2022-W46: 0.027, 2022-W47: 0.093, 2022-W48: -0.081, 2022-W49: 0.059, 2022-W50: 0.096, 2022-W51: 0.25, 2022-W52: 0.083, 2023-W1: -0.036, 2023-W2: 0.105, 2023-W3: 0.175, 2023-W4: -0.014 | 6.14 | 52.43 | 9.94 | 2.98 | 0: 2916 | 2022-W35: 0.079, 2022-W36: 0.228, 2022-W37: 0.279, 2022-W38: 0.311, 2022-W39: 0.21, 2022-W40: -0.037, 2022-W41: 0.072, 2022-W42: 0.206, 2022-W43: 0.085, 2022-W44: -0.003, 2022-W45: 0.043, 2022-W46: 0.06, 2022-W47: 0.137, 2022-W48: 0.012, 2022-W49: -0.212, 2022-W50: 0.27, 2022-W51: -0.221, 2022-W52: 0.041, 2023-W1: 0.162, 2023-W2: 0.094, 2023-W3: -0.143, 2023-W4: -0.157 | 6.16 | 53.47 | 9.98 | 2.97 | 0: 2916 | 2022-W35: 0.28, 2022-W36: 0.11, 2022-W37: 0.2, 2022-W38: 0.23, 2022-W39: 0.15, 2022-W40: 0.04, 2022-W41: 0.08, 2022-W42: 0.18, 2022-W43: 0.12, 2022-W44: 0.1, 2022-W45: 0.06, 2022-W46: 0.07, 2022-W47: 0.04, 2022-W48: 0.09, 2022-W49: -0.03, 2022-W50: 0.08, 2022-W51: 0.97, 2022-W52: 0.04, 2023-W1: 0.03, 2023-W2: 0.14, 2023-W3: -0.06, 2023-W4: 0 |
| 80 | 6.25 | 52.79 | 10.01 | 2.98 | 0: 2886, 1: 27, 2: 3 | 2022-W35: 6.468, 2022-W36: 4.897, 2022-W37: 5.337, 2022-W38: 4.432, 2022-W39: 4.082, 2022-W40: 4.536, 2022-W41: 2.716, 2022-W42: 3.815, 2022-W43: 2.923, 2022-W44: 2.92, 2022-W45: 3.697, 2022-W46: 2.971, 2022-W47: 3.801, 2022-W48: 4.134, 2022-W49: 2.934, 2022-W50: 2.52, 2022-W51: 5.259, 2022-W52: 2.579, 2023-W1: 3, 2023-W2: 4.108, 2023-W3: 3.343, 2023-W4: 2.755 | 6.25 | 52.79 | 10.02 | 3.06 | 0: 2888, 1: 25, 2: 2 | 2022-W35: 7.912, 2022-W36: 6.711, 2022-W37: 5.425, 2022-W38: 5.976, 2022-W39: 4.634, 2022-W40: 3.413, 2022-W41: 4.874, 2022-W42: 3.791, 2022-W43: 4.951, 2022-W44: 4.838, 2022-W45: 4.522, 2022-W46: 5.902, 2022-W47: 5.95, 2022-W48: 3.248, 2022-W49: 4.631, 2022-W50: 5.144, 2022-W51: 2.34, 2022-W52: 6.289, 2023-W1: 4.288, 2023-W2: 4.181, 2023-W3: 3.85, 2023-W4: 5.143 | 6.47 | 54.04 | 10.02 | 3 | 0: 2885, 1: 28, 2: 2, 3: 1 | 2022-W35: 7.867, 2022-W36: 5.576, 2022-W37: 4.179, 2022-W38: 5.423, 2022-W39: 4.639, 2022-W40: 3.667, 2022-W41: 5.25, 2022-W42: 5.495, 2022-W43: 2.641, 2022-W44: 2.945, 2022-W45: 2.77, 2022-W46: 3.688, 2022-W47: 5.605, 2022-W48: 3.983, 2022-W49: 2.54, 2022-W50: 4.354, 2022-W51: 6.127, 2022-W52: 3.895, 2023-W1: 2.473, 2023-W2: 4.749, 2023-W3: 3.632, 2023-W4: 3.984 | 6.28 | 53.07 | 10.05 | 3.01 | 0: 2889, 1: 25, 3: 1 | 2022-W35: 7.419, 2022-W36: 4.943, 2022-W37: 4.391, 2022-W38: 4.142, 2022-W39: 4.975, 2022-W40: 2.798, 2022-W41: 3.223, 2022-W42: 2.923, 2022-W43: 2.06, 2022-W44: 2.97, 2022-W45: 3.002, 2022-W46: 2.994, 2022-W47: 4.926, 2022-W48: 3.589, 2022-W49: 3.013, 2022-W50: 3.227, 2022-W51: 4.458, 2022-W52: 4.719, 2023-W1: 2.336, 2023-W2: 3.847, 2023-W3: 4.358, 2023-W4: 4.427 | 6.14 | 52.43 | 9.99 | 2.98 | 0: 2893, 1: 19, 2: 4 | 2022-W35: 7.847, 2022-W36: 4.579, 2022-W37: 4.905, 2022-W38: 4.917, 2022-W39: 3.586, 2022-W40: 4.312, 2022-W41: 2.917, 2022-W42: 5.068, 2022-W43: 4.094, 2022-W44: 2.388, 2022-W45: 4.662, 2022-W46: 5.116, 2022-W47: 2.603, 2022-W48: 4.049, 2022-W49: 3.489, 2022-W50: 4.793, 2022-W51: 4.537, 2022-W52: 3.714, 2023-W1: 3.129, 2023-W2: 6.016, 2023-W3: 4.412, 2023-W4: 3.292 | 6.28 | 53.02 | 10.02 | 3.01 | 0: 2888, 1: 25, 2: 2 | 2022-W35: 7.5, 2022-W36: 5.34, 2022-W37: 4.85, 2022-W38: 4.98, 2022-W39: 4.38, 2022-W40: 3.75, 2022-W41: 3.8, 2022-W42: 4.22, 2022-W43: 3.33, 2022-W44: 3.21, 2022-W45: 3.73, 2022-W46: 4.11, 2022-W47: 4.58, 2022-W48: 3.8, 2022-W49: -0.03, 2022-W50: 0.08, 2022-W51: 0.97, 2022-W52: 0.04, 2023-W1: 0.03, 2023-W2: 0.14, 2023-W3: -0.06, 2023-W4: 0 |
| 60 | 6.25 | 52.79 | 10.09 | 3.11 | 0: 2793, 1: 79, 2: 26, 3: 12 | 2022-W35: 6.265, 2022-W36: 5.564, 2022-W37: 4.021, 2022-W38: 5.457, 2022-W39: 4.193, 2022-W40: 3.839, 2022-W41: 3.633, 2022-W42: 3.649, 2022-W43: 2.963, 2022-W44: 3.719, 2022-W45: 2.394, 2022-W46: 3.834, 2022-W47: 4.235, 2022-W48: 4.786, 2022-W49: 4.438, 2022-W50: 4.305, 2022-W51: 4.799, 2022-W52: 4.323, 2023-W1: 3.917, 2023-W2: 5.36, 2023-W3: 4.73, 2023-W4: 3.99 | 6.25 | 52.79 | 10.3 | 3.28 | 0: 2779, 1: 102, 2: 24, 3: 7 | 2022-W35: 6.682, 2022-W36: 5.388, 2022-W37: 5.129, 2022-W38: 5.205, 2022-W39: 4.501, 2022-W40: 3.308, 2022-W41: 5.129, 2022-W42: 4.136, 2022-W43: 4.797, 2022-W44: 3.898, 2022-W45: 3.008, 2022-W46: 5.828, 2022-W47: 4.626, 2022-W48: 3.436, 2022-W49: 3.992, 2022-W50: 5.433, 2022-W51: 3.131, 2022-W52: 4.94, 2023-W1: 2.943, 2023-W2: 5.489, 2023-W3: 4.082, 2023-W4: 4.107 | 6.47 | 54.04 | 10.08 | 3.18 | 0: 2800, 1: 80, 2: 22, 3: 8, 4:1 | 2022-W35: 7.769, 2022-W36: 5.678, 2022-W37: 3.809, 2022-W38: 4.824, 2022-W39: 4.464, 2022-W40: 4.187, 2022-W41: 4.007, 2022-W42: 4.561, 2022-W43: 4.202, 2022-W44: 3.598, 2022-W45: 3.427, 2022-W46: 4.696, 2022-W47: 4.484, 2022-W48: 6.167, 2022-W49: 3.246, 2022-W50: 3.901, 2022-W51: 5.791, 2022-W52: 6.721, 2023-W1: 3.674, 2023-W2: 4.964, 2023-W3: 3.586, 2023-W4: 5.439 | 6.28 | 53.07 | 10.26 | 3.18 | 0: 2787, 1: 87, 2: 28, 3: 9 | 2022-W35: 7.368, 2022-W36: 5.392, 2022-W37: 3.926, 2022-W38: 4.013, 2022-W39: 5.964, 2022-W40: 2.79, 2022-W41: 2.935, 2022-W42: 3.456, 2022-W43: 2.828, 2022-W44: 3.157, 2022-W45: 4.355, 2022-W46: 3.27, 2022-W47: 4.86, 2022-W48: 3.645, 2022-W49: 4.186, 2022-W50: 2.822, 2022-W51: 4.489, 2022-W52: 5.069, 2023-W1: 2.475, 2023-W2: 3.581, 2023-W3: 5.204, 2023-W4: 4.534 | 5.95 | 51.42 | 10.04 | 3.13 | 0: 2819, 1: 64, 2: 23, 3: 8 | 2022-W35: 7.405, 2022-W36: 5.215, 2022-W37: 5.085, 2022-W38: 4.433, 2022-W39: 4.856, 2022-W40: 3.888, 2022-W41: 3.002, 2022-W42: 4.93, 2022-W43: 5.51, 2022-W44: 4.072, 2022-W45: 4.309, 2022-W46: 5.379, 2022-W47: 4.491, 2022-W48: 5.535, 2022-W49: 5.902, 2022-W50: 5.924, 2022-W51: 5.281, 2022-W52: 5.384, 2023-W1: 3.266, 2023-W2: 5.395, 2023-W3: 4.797, 2023-W4: 6.056 | 6.24 | 52.82 | 10.15 | 3.18 | 0: 2796, 1: 82, 2: 25, 3: 9 | 2022-W35: 7.1, 2022-W36: 5.45, 2022-W37: 4.39, 2022-W38: 4.79, 2022-W39: 4.8, 2022-W40: 3.6, 2022-W41: 3.74, 2022-W42: 4.15, 2022-W43: 4.06, 2022-W44: 3.69, 2022-W45: 3.5, 2022-W46: 4.6, 2022-W47: 4.54, 2022-W48: 4.71, 2022-W49: 4.35, 2022-W50: 4.48, 2022-W51: 4.7, 2022-W52: 5.29, 2023-W1: 3.26, 2023-W2: 4.96, 2023-W3: 4.48, 2023-W4: 4.83 |
| 40 | 6.06 | 51.79 | 10.33 | 3.61 | 0: 2636, 1: 147, 2: 70, 3: 42, 4: 2, 5:1 | 2022-W35: 6.631, 2022-W36: 5.645, 2022-W37: 5.433, 2022-W38: 3.994, 2022-W39: 4.301, 2022-W40: 4.39, 2022-W41: 4.256, 2022-W42: 4.904, 2022-W43: 3.432, 2022-W44: 3.989, 2022-W45: 4.349, 2022-W46: 4.564, 2022-W47: 5.111, 2022-W48: 5.891, 2022-W49: 4.9, 2022-W50: 4.031, 2022-W51: 5.569, 2022-W52: 5.077, 2023-W1: 4.889, 2023-W2: 6.46, 2023-W3: 4.699, 2023-W4: 3.634 | 6.06 | 51.79 | 10.47 | 3.59 | 0: 2623, 1: 160, 2: 71, 3: 43, 4: 1 | 2022-W35: 6.209, 2022-W36: 5.83, 2022-W37: 5.927, 2022-W38: 5.307, 2022-W39: 4.578, 2022-W40: 3.07, 2022-W41: 5.654, 2022-W42: 3.674, 2022-W43: 5.562, 2022-W44: 5.227, 2022-W45: 3.982, 2022-W46: 4.591, 2022-W47: 4.825, 2022-W48: 2.153, 2022-W49: 3.13, 2022-W50: 5.156, 2022-W51: 3.476, 2022-W52: 6.446, 2023-W1: 2.226, 2023-W2: 6.041, 2023-W3: 5.179, 2023-W4: 6.193 | 6.47 | 54.04 | 10.42 | 3.55 | 0: 2588, 1: 190, 2: 70, 3: 38, 4: 9 | 2022-W35: 7.495, 2022-W36: 5.624, 2022-W37: 5.022, 2022-W38: 4.302, 2022-W39: 5.015, 2022-W40: 3.847, 2022-W41: 4.975, 2022-W42: 4.612, 2022-W43: 3.808, 2022-W44: 3.532, 2022-W45: 4.783, 2022-W46: 5.395, 2022-W47: 6.216, 2022-W48: 5.508, 2022-W49: 4.559, 2022-W50: 5.635, 2022-W51: 5.587, 2022-W52: 6.537, 2023-W1: 3.988, 2023-W2: 4.488, 2023-W3: 3.751, 2023-W4: 4.956 | 6.28 | 53.07 | 10.58 | 3.55 | 0: 2590, 1: 189, 2: 83, 3: 35, 4: 1 | 2022-W35: 6.108, 2022-W36: 5.682, 2022-W37: 4.057, 2022-W38: 4.931, 2022-W39: 4.598, 2022-W40: 3.927, 2022-W41: 4.095, 2022-W42: 3.34, 2022-W43: 3.372, 2022-W44: 2.98, 2022-W45: 5.079, 2022-W46: 5.786, 2022-W47: 6.067, 2022-W48: 3.874, 2022-W49: 3.354, 2022-W50: 3.872, 2022-W51: 5.055, 2022-W52: 5.17, 2023-W1: 3.753, 2023-W2: 3.88, 2023-W3: 2.768, 2023-W4: 5.141 | 6.14 | 52.43 | 10.43 | 3.49 | 0: 2659, 1: 146, 2: 68, 3: 34, 4: 2 | 2022-W35: 6.734, 2022-W36: 5.197, 2022-W37: 5.249, 2022-W38: 5.372, 2022-W39: 4.804, 2022-W40: 3.477, 2022-W41: 4.571, 2022-W42: 4.016, 2022-W43: 5.613, 2022-W44: 4.943, 2022-W45: 5.855, 2022-W46: 5.855, 2022-W47: 1.499, 2022-W48: 5.867, 2022-W49: 3.076, 2022-W50: 4.024, 2022-W51: 7.423, 2022-W52: 3.264, 2023-W1: 2.634, 2023-W2: 4.985, 2023-W3: 4.192, 2023-W4: 3.898 | 6.20 | 52.62 | 10.45 | 3.56 | 0: 2619, 1: 166, 2: 72, 3: 38, 4:3 | 2022-W35: 6.64, 2022-W36: 5.6, 2022-W37: 5.14, 2022-W38: 4.78, 2022-W39: 4.66, 2022-W40: 3.74, 2022-W41: 4.71, 2022-W42: 4.51, 2022-W43: 4.36, 2022-W44: 4.13, 2022-W45: 4.37, 2022-W46: 5.24, 2022-W47: 4.74, 2022-W48: 4.66, 2022-W49: 3.8, 2022-W50: 4.54, 2022-W51: 5.42, 2022-W52: 5.3, 2023-W1: 3.5, 2023-W2: 5.17, 2023-W3: 4.12, 2023-W4: 4.76 |
| 20 | 6.06 | 51.79 | 11.35 | 4.38 | 0: 2201, 1: 279, 2: 235, 3: 137, 4: 26 | 2022-W35: 6.077, 2022-W36: 5.69, 2022-W37: 5.249, 2022-W38: 5.358, 2022-W39: 5.965, 2022-W40: 4.645, 2022-W41: 4.283, 2022-W42: 3.352, 2022-W43: 4.809, 2022-W44: 4.059, 2022-W45: 3.42, 2022-W46: 4.956, 2022-W47: 3.718, 2022-W48: 6.05, 2022-W49: 3.222, 2022-W50: 5.333, 2022-W51: 5.269, 2022-W52: 3.986, 2023-W1: 7.558, 2023-W2: 4.901, 2023-W3: 4.95, 2023-W4: 6.275 | 6.06 | 51.79 | 11.49 | 4.46 | 0: 2155, 1: 144, 2: 231, 3: 144, 4: 22 | 2022-W35: 6.533, 2022-W36: 6.423, 2022-W37: 6.076, 2022-W38: 6.297, 2022-W39: 3.798, 2022-W40: 4.535, 2022-W41: 6.038, 2022-W42: 4.584, 2022-W43: 3.973, 2022-W44: 4.9, 2022-W45: 5.02, 2022-W46: 6.281, 2022-W47: 5.86, 2022-W48: 4.176, 2022-W49: 4.665, 2022-W50: 6.736, 2022-W51: 2.13, 2022-W52: 3.718, 2023-W1: 5.271, 2023-W2: 4.588, 2023-W3: 4.186, 2023-W4: 4.377 | 6.28 | 53.07 | 11.3 | 4.33 | 0: 2162, 1: 319, 2: 288, 3: 150, 4: 18 | 2022-W35: 7.299, 2022-W36: 6.058, 2022-W37: 4.407, 2022-W38: 5.592, 2022-W39: 4.624, 2022-W40: 4.166, 2022-W41: 3.038, 2022-W42: 5.68, 2022-W43: 5.553, 2022-W44: 3.303, 2022-W45: 4.759, 2022-W46: 6.296, 2022-W47: 5.402, 2022-W48: 7.437, 2022-W49: 6.486, 2022-W50: 2.357, 2022-W51: 4.635, 2022-W52: 7.998, 2023-W1: 4.447, 2023-W2: 6.033, 2023-W3: 5.982, 2023-W4: 4.691 | 6.25 | 52.79 | 11.24 | 4.32 | 0: 2243, 1: 277, 2: 198, 3: 134, 4: 17 | 2022-W35: 6.746, 2022-W36: 6.012, 2022-W37: 3.7, 2022-W38: 5.789, 2022-W39: 4.005, 2022-W40: 3.7, 2022-W41: 3.926, 2022-W42: 3.418, 2022-W43: 3.203, 2022-W44: 5.004, 2022-W45: 4.923, 2022-W46: 3.843, 2022-W47: 3.516, 2022-W48: 4.284, 2022-W49: 5.292, 2022-W50: 3.236, 2022-W51: 6.581, 2022-W52: 7.907, 2023-W1: 6.01, 2023-W2: 5.35, 2023-W3: 6.929, 2023-W4: 3.627 | 5.95 | 51.42 | 11.14 | 4.33 | 0: 2274, 1: 280, 2: 176, 3: 130, 4: 17 | 2022-W35: 7.141, 2022-W36: 4.881, 2022-W37: 5.204, 2022-W38: 4.969, 2022-W39: 5.885, 2022-W40: 3.482, 2022-W41: 4.424, 2022-W42: 5.874, 2022-W43: 5.197, 2022-W44: 6.449, 2022-W45: 5.285, 2022-W46: 4.737, 2022-W47: 4.784, 2022-W48: 4.624, 2022-W49: 2.761, 2022-W50: 4.669, 2022-W51: 6.386, 2022-W52: 6.348, 2023-W1: 2.842, 2023-W2: 5.651, 2023-W3: 6.592, 2023-W4: 6.595 | 6.12 | 52.17 | 11.3 | 4.36 | 0: 2207, 1: 296, 2: 214, 3: 139, 4: 20 | 2022-W35: 6.76, 2022-W36: 5.81, 2022-W37: 4.93, 2022-W38: 5.6, 2022-W39: 4.86, 2022-W40: 4.11, 2022-W41: 4.34, 2022-W42: 4.58, 2022-W43: 4.55, 2022-W44: 4.74, 2022-W45: 4.68, 2022-W46: 5.22, 2022-W47: 4.66, 2022-W48: 5.31, 2022-W49: 4.49, 2022-W50: 4.47, 2022-W51: 5, 2022-W52: 5.99, 2023-W1: 5.23, 2023-W2: 5.3, 2023-W3: 5.73, 2023-W4: 5.11 |

F - Average Wait Time of non-escalated tickets (seconds); G - Wait Time of non-escalated tickets standard deviation; H - Average response time of non-escalated tickets (minutes); I - Response Time of non-escalated tickets standard deviation; J - Readaptions; K - Time difference evolution between observed and estimated actions (when they are equal at a ROAR of 100 and different when ROAR != 100).

Table A.19: CROSS Acceptance Evolution (all seeds)

| ROAR | Seed | Week/Year | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2022-W35 | 2022-W36 | 2022-W37 | 2022-W38 | 2022-W39 | 2022-W40 | 2022-W41 | 2022-W42 | 2022-W43 | 2022-W44 | 2022-W45 | 2022-W46 | 2022-W47 | 2022-W48 | 2022-W49 | 2022-W50 | 2022-W51 | 2022-W52 | 2023-W1 | 2023-W2 | 2023-W3 | 2023-W4 |
| 80 | 2 | 79.1 | 94.12 | 87.59 | 92.91 | 93.55 | 94.37 | 90 | 94.56 | 92.41 | 95.04 | 91.27 | 96.58 | 94.7 | 90.91 | 96.88 | 92.48 | 96.09 | 96.3 | 95.61 | 95.11 | 96.69 | 92.14 |
| | 3 | 83.58 | 92.16 | 90.51 | 91.34 | 91.13 | 95.07 | 92.31 | 92.52 | 96.55 | 95.04 | 97.32 | 92.47 | 94.7 | 93.71 | 96.09 | 89.47 | 95.31 | 96.3 | 97.37 | 93.71 | 93.39 | 95.71 |
| | 4 | 77.61 | 90.2 | 90.51 | 92.13 | 91.13 | 96.48 | 96.15 | 93.88 | 92.41 | 95.75 | 95.3 | 93.84 | 96.97 | 92.31 | 96.09 | 93.99 | 96.09 | 92.59 | 93.86 | 94.41 | 90.08 | 93.57 |
| | 5 | 86.57 | 88.24 | 91.24 | 88.98 | 93.55 | 90.14 | 90.77 | 92.52 | 93.1 | 92.2 | 94.63 | 93.84 | 93.94 | 93.08 | 91.41 | 91.73 | 95.31 | 94.82 | 90.35 | 93.71 | 95.04 | 97.14 |
| | 6 | 83.56 | 80.39 | 79.78 | 90.55 | 94.36 | 94.37 | 93.08 | 93.88 | 92.41 | 96.45 | 92.62 | 95.21 | 93.18 | 93.01 | 91.41 | 94.74 | 93.75 | 96.3 | 92.98 | 97.2 | 97.52 | 92.86 |
| | **AVG** | 82.09 | 89.02 | 87.93 | 91.18 | 92.74 | 94.08 | 92.46 | 93.47 | 93.38 | 94.89 | 94.23 | 94.38 | 94.7 | 92.6 | 94.38 | 92.48 | 95.31 | 95.26 | 94.04 | 94.83 | 94.55 | 94.29 |
| 60 | 2 | 68.66 | 78.43 | 74.45 | 87.4 | 80.65 | 73.24 | 86.92 | 82.99 | 88.97 | 87.94 | 84.56 | 88.36 | 88.64 | 82.52 | 89.84 | 87.97 | 85.94 | 88.15 | 90.35 | 90.91 | 90.08 | 91.43 |
| | 3 | 61.19 | 82.35 | 81.02 | 77.95 | 82.26 | 84.51 | 88.46 | 85.03 | 83.45 | 89.36 | 86.58 | 90.41 | 85.61 | 83.92 | 87.5 | 84.96 | 89.06 | 91.85 | 86.84 | 86.01 | 85.95 | 83.57 |
| | 4 | 65.67 | 75.49 | 74.45 | 74.02 | 91.13 | 83.8 | 86.92 | 91.84 | 83.45 | 89.36 | 89.93 | 90.41 | 90.15 | 81.82 | 88.28 | 87.97 | 85.94 | 85.19 | 88.6 | 89.51 | 85.95 | 83.57 |
| | 5 | 68.66 | 74.51 | 75.91 | 81.1 | 86.29 | 83.1 | 80 | 86.4 | 86.9 | 90.07 | 87.25 | 89.73 | 85.61 | 88.81 | 84.38 | 93.23 | 87.5 | 86.67 | 78.07 | 85.32 | 83.47 | 87.86 |
| | 6 | 74.63 | 70.59 | 74.45 | 85.04 | 87.1 | 83.8 | 91.54 | 89.12 | 82.07 | 91.49 | 87.25 | 82.12 | 89.39 | 85.32 | 92.19 | 84.96 | 89.06 | 85.93 | 89.47 | 83.22 | 85.95 | 89.29 |
| | **AVG** | 67.76 | 76.27 | 76.06 | 81.1 | 85.48 | 81.69 | 86.77 | 87.08 | 84.97 | 89.65 | 87.11 | 88.2 | 87.88 | 84.48 | 88.44 | 87.82 | 87.5 | 87.56 | 86.67 | 86.99 | 86.28 | 87.14 |
| 40 | 2 | 41.79 | 59.8 | 67.88 | 63.78 | 75.81 | 66.9 | 79.23 | 72.79 | 80 | 73.05 | 82.55 | 82.88 | 87.88 | 80.42 | 84.38 | 81.29 | 79.69 | 81.48 | 89.47 | 86.01 | 81.82 | 81.43 |
| | 3 | 52.24 | 65.69 | 64.96 | 67.72 | 76.61 | 77.47 | 76.92 | 5.51 | 73.1 | 76.6 | 77.85 | 80.82 | 81.06 | 79.02 | 81.25 | 77.44 | 78.91 | 82.96 | 86.84 | 76.22 | 83.47 | 76.43 |
| | 4 | 52.24 | 52.94 | 58.39 | 67.72 | 76.61 | 73.94 | 76.15 | 75.51 | 74.48 | 72.34 | 85.24 | 82.88 | 74.24 | 79.72 | 81.25 | 78.2 | 85.94 | 76.3 | 80.7 | 78.32 | 83.47 | 75 |
| | 5 | 43.28 | 62.75 | 61.31 | 75.59 | 72.58 | 75.35 | 75.39 | 76.87 | 73.1 | 73.05 | 69.8 | 80.82 | 79.55 | 74.83 | 77.34 | 81.96 | 76.56 | 75.56 | 83.33 | 78.32 | 73.55 | 77.14 |
| | 6 | 47.76 | 61.77 | 62.74 | 70.87 | 80.65 | 75.35 | 78.46 | 74.83 | 73.1 | 80.14 | 77.85 | 78.08 | 76.52 | 86.71 | 89.06 | 82.71 | 79.69 | 78.52 | 84.21 | 76.92 | 76.03 | 78.57 |
| | **AVG** | 47.46 | 60.59 | 63.06 | 69.13 | 76.45 | 73.8 | 77.23 | 75.1 | 74.76 | 75.04 | 78.66 | 81.1 | 79.85 | 80.14 | 82.66 | 80.32 | 80.16 | 78.96 | 84.91 | 79.16 | 79.67 | 77.71 |
| 20 | 2 | 14.93 | 42.16 | 42.26 | 44.09 | 49.19 | 52.13 | 63.85 | 55.1 | 64.14 | 67.38 | 59.06 | 65.07 | 69.7 | 72.73 | 76.56 | 72.93 | 55.47 | 62.96 | 62.28 | 67.83 | 73.55 | 67.86 |
| | 3 | 37.31 | 32.35 | 37.96 | 38.58 | 46.77 | 50 | 43.08 | 48.3 | 50.35 | 46.81 | 59.73 | 67.12 | 65.91 | 58.04 | 64.84 | 58.65 | 67.19 | 71.11 | 68.42 | 58.04 | 61.98 | 59.29 |
| | 4 | 29.85 | 35.29 | 32.85 | 37.8 | 58.87 | 57.04 | 55.39 | 48.98 | 48.28 | 60.99 | 69.8 | 65.07 | 61.36 | 70.63 | 68.75 | 63.16 | 61.72 | 60 | 60.53 | 65.73 | 59.5 | 59.29 |
| | 5 | 22.39 | 37.26 | 37.23 | 48.03 | 59.68 | 52.82 | 50 | 61.91 | 65.52 | 68.09 | 55.71 | 61.64 | 66.67 | 71.33 | 64.06 | 73.68 | 73.44 | 69.63 | 70.18 | 66.43 | 52.07 | 68.57 |
| | 6 | 22.39 | 46.08 | 46.72 | 46.46 | 64.52 | 53.52 | 58.46 | 65.31 | 59.31 | 67.38 | 72.48 | 58.9 | 57.58 | 65.73 | 77.34 | 67.67 | 67.19 | 67.19 | 68.42 | 66.43 | 68.6 | 75 |
| | **AVG** | 25.37 | 38.63 | 39.4 | 42.99 | 55.81 | 53.1 | 54.15 | 55.92 | 57.52 | 62.13 | 63.36 | 63.56 | 64.24 | 67.69 | 70.31 | 67.22 | 65 | 66.18 | 65.96 | 64.9 | 63.14 | 66 |

ROAR - RecOmmendation Acceptance Rate.

## A.3 FEDOT AutoML Performance

Table A.20: FEDOT with two training years and seasonality (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | | Run 5 | | | | Run 6 | | | | Run 7 | | | | Run 8 | | | | Run 9 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 17.42 | 18.44 | 1.93 | 25.27 | 16.03 | 18.16 | 1.77 | 5.02 | 18.05 | 20.48 | 1.98 | 5.06 | 16.3 | 19 | 1.79 | 5.03 | 17.23 | 18.22 | 1.9 | 5.13 | 17.23 | 18.22 | 1.9 | 5.25 | 4.85 | 6.44 | 0.39 | 46.26 | 10.95 | 12.59 | 1.16 | 5.08 | 16.6 | 19.28 | 1.82 | 5.01 | 18.05 | 20.48 | 1.98 | 10.12 |
| 1 | 15.6 | 17.79 | 1.71 | 5.1 | 17.26 | 18.26 | 1.91 | 5.12 | 4.31 | 6.32 | 0.29 | 5.83 | 4.18 | 5.94 | 0.28 | 6.21 | 12.35 | 18.36 | 1.18 | 5.21 | 18.09 | 20.5 | 1.98 | 5.05 | 17.26 | 18.26 | 1.91 | 5.13 | 5.52 | 8.57 | 0.44 | 5.2 | 18.17 | 20.58 | 1.99 | 5.06 | 12.9 | 15.28 | 1.4 | 5.1 |
| 2 | 18.22 | 20.65 | 1.99 | 5.08 | 17.71 | 19.36 | 1.96 | 5.52 | 17.78 | 20.06 | 1.95 | 5.72 | 16.55 | 19.25 | 1.81 | 5.01 | 16.3 | 17.8 | 1.81 | 5.02 | 17.87 | 19.83 | 1.97 | 5.62 | 17.26 | 18.26 | 1.91 | 5.22 | 18.22 | 20.65 | 1.99 | 5.11 | 16.38 | 20.99 | 1.75 | 5.27 | 18.22 | 20.65 | 1.99 | 5.08 |
| 3 | 17.26 | 19.26 | 1.91 | 5.12 | 17.28 | 18.27 | 1.91 | 5.13 | 17.28 | 18.27 | 1.91 | 5.14 | 18.04 | 20.37 | 1.98 | 5.63 | 8.22 | 10.91 | 0.79 | 17.53 | 18.44 | 20.86 | 2.02 | 5.6 | 18.04 | 20.37 | 1.98 | 5.59 | 18.44 | 20.86 | 2.02 | 5.6 | 18.04 | 20.37 | 1.98 | 5.59 | 18.44 | 20.92 | 2.02 | 11.05 |
| 4 | 4.58 | 6.68 | 0.35 | 68.57 | 17.85 | 20.19 | 1.96 | 5.57 | 17.26 | 18.26 | 1.91 | 6 | 18.25 | 20.02 | 1.97 | 5.07 | 18.22 | 20.65 | 1.99 | 11.33 | 17.26 | 18.26 | 1.91 | 5.13 | 17.85 | 20.19 | 1.96 | 5.65 | 17.85 | 20.19 | 1.96 | 5.83 | 16.3 | 17.8 | 1.81 | 5.05 | 18.45 | 20.91 | 2.02 | 6.04 |
| 5 | 8.12 | 12.13 | 0.73 | 17.76 | 13.14 | 14.64 | 1.45 | 5.25 | 19.35 | 22.13 | 2.11 | 5.62 | 4.35 | 6.41 | 0.3 | 6.93 | 4.18 | 5.88 | 0.28 | 7.38 | 4.49 | 6.52 | 0.34 | 79 | 4.19 | 5.96 | 0.28 | 6.87 | 6.08 | 8 | 0.54 | 70.49 | 12.1 | 14.48 | 1.32 | 5.28 | 4.15 | 5.84 | 0.27 | 5.95 |
| 6 | 17.26 | 18.26 | 1.91 | 5.12 | 17.32 | 20.13 | 1.89 | 5.02 | 18.22 | 20.65 | 1.99 | 10.24 | 17.26 | 18.26 | 1.91 | 5.27 | 4.2 | 5.95 | 0.28 | 5 | 29.95 | 32.58 | 3.24 | 5.98 | 11.49 | 15.59 | 1.12 | 5.04 | 6.17 | 9.57 | 0.51 | 5.04 | 18.59 | 19.64 | 2.05 | 5.34 | 24.68 | 27.02 | 2.68 | 5.97 |
| 7 | 17.264 | 18.26 | 1.91 | 5.16 | 16.3 | 17.8 | 1.81 | 5.01 | 17.04 | 19.49 | 1.87 | 5.01 | 17.26 | 18.26 | 1.91 | 17.26 | 18.26 | 18.26 | 1.91 | 5.14 | 17.26 | 18.26 | 1.91 | 5.13 | 17.26 | 18.26 | 1.91 | 5.12 | 17.26 | 18.26 | 1.91 | 5.11 | 17.26 | 18.26 | 1.91 | 5.1 | 17.26 | 18.26 | 1.91 | 5.15 |
| 8 | 17.62 | 20.02 | 1.93 | 5.08 | 18.61 | 21.22 | 2.03 | 5.55 | 17.26 | 18.26 | 1.91 | 5.26 | 17.94 | 20.39 | 1.96 | 5.06 | 17.24 | 18.23 | 1.9 | 5.14 | 18.01 | 20.54 | 1.97 | 5.61 | 16.3 | 17.8 | 1.81 | 5.01 | 18.01 | 20.54 | 1.97 | 5.65 | 18.22 | 20.65 | 1.99 | 5.08 | 14.27 | 15.95 | 1.55 | 5.09 |
| 9 | 4.57 | 6.64 | 0.35 | 71.86 | 17.99 | 20.36 | 1.87 | 6.27 | 17.99 | 20.36 | 1.97 | 5.66 | 17.26 | 18.26 | 1.91 | 5.17 | 6.36 | 8.38 | 0.57 | 74.62 | 17.26 | 18.26 | 1.91 | 5.11 | 22.54 | 26.77 | 2.21 | 88.31 | 17.26 | 18.26 | 1.91 | 5.11 | 17.99 | 20.36 | 1.97 | 5.58 | 4.21 | 5.76 | 0.31 | 82 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

Table A.21: FEDOT with three training years and seasonality (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | | Run 5 | | | | Run 6 | | | | Run 7 | | | | Run 8 | | | | Run 9 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 3.97 | 5.77 | 0.28 | 263.2 | 3.97 | 5.77 | 0.28 | 238.71 | 3.97 | 5.77 | 0.28 | 227.4 | 3.97 | 5.77 | 0.28 | 243.1 | 3.97 | 5.77 | 0.28 | 243.1 | 3.97 | 5.77 | 0.28 | 243.1 | 3.97 | 5.77 | 0.28 | 243.1 | 3.97 | 5.77 | 0.28 | 243.1 | 3.97 | 5.77 | 0.28 | 243.1 | 3.97 | 5.77 | 0.28 | 243.1 |
| 1 | 3.97 | 5.5 | 0.27 | 74.59 | 3.97 | 5.5 | 0.27 | 72.62 | 3.97 | 5.5 | 27 | 72.91 | 3.97 | 5.5 | 0.27 | 72.85 | 3.97 | 5.5 | 0.27 | 73.24 | 3.97 | 5.5 | 0.27 | 73.24 | 3.97 | 5.5 | 0.27 | 73.24 | 3.97 | 5.5 | 0.27 | 73.24 | 3.97 | 5.5 | 0.27 | 73.24 | 3.97 | 5.5 | 0.27 | 73.24 |
| 2 | 5.99 | 8.09 | 0.35 | 5.08 | 5.99 | 8.09 | 0.35 | 5.02 | 5.97 | 8.05 | 0.35 | 5 | 5.99 | 8.09 | 0.35 | 5.02 | 5.99 | 8.09 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5 | 4.09 | 5.78 | 0.31 | 117.66 | 4.09 | 5.78 | 0.31 | 123.16 |
| 3 | 4.98 | 6.6 | 0.37 | 5.02 | 4.98 | 6.6 | 0.37 | 5.02 | 4.98 | 6.6 | 0.37 | 5 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.02 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.01 |
| 4 | 4.11 | 5.83 | 0.31 | 413.23 | 4.11 | 5.83 | 0.31 | 363.77 | 4.11 | 5.83 | 0.31 | 361.6 | 4.11 | 5.83 | 0.31 | 379.53 | 4.11 | 5.83 | 0.31 | 379.53 | 4.11 | 5.83 | 0.31 | 379.53 | 4.11 | 5.83 | 0.31 | 379.53 | 4.11 | 5.83 | 0.31 | 379.53 | 4.11 | 5.83 | 0.31 | 379.53 | 4.11 | 5.83 | 0.31 | 379.53 |
| 5 | 4.98 | 6.6 | 0.37 | 6.4 | 4.98 | 6.6 | 0.37 | 5.82 | 4.98 | 6.6 | 0.37 | 5.83 | 4.98 | 6.6 | 0.37 | 5.83 | 4.98 | 6.6 | 0.37 | 5.85 | 4.98 | 6.6 | 0.37 | 6.01 | 4.98 | 6.6 | 0.37 | 5.77 | 4.98 | 6.6 | 0.37 | 5.8 | 4.98 | 6.6 | 0.37 | 5.79 | 4.98 | 6.6 | 0.37 | 5.86 |
| 6 | 4.66 | 6.3 | 0.34 | 5.01 | 4.66 | 6.3 | 0.33 | 5.02 | 4.71 | 6.34 | 0.34 | 5.01 | 4.61 | 6.25 | 0.33 | 5.02 | 4.67 | 6.31 | 0.34 | 5.02 | 4.74 | 6.38 | 0.34 | 5.01 | 4.61 | 6.25 | 0.33 | 5.01 | 4.68 | 6.32 | 0.34 | 5.02 | 4.69 | 6.33 | 0.34 | 5.02 | 4.62 | 6.25 | 0.33 | 5.01 |
| 7 | 4.98 | 6.6 | 0.37 | 9.54 | 4.98 | 6.6 | 0.37 | 9.7 | 4.98 | 6.6 | 0.37 | 9.43 | 4.98 | 6.6 | 0.37 | 9.43 | 4.98 | 6.6 | 0.37 | 9.59 | 4.98 | 6.6 | 0.37 | 9.67 | 4.98 | 6.6 | 0.37 | 9.73 | 4.98 | 6.6 | 0.37 | 9.33 | 4.98 | 6.6 | 0.37 | 9.5 | 4.98 | 6.6 | 0.37 | 9.62 |
| 8 | 4.11 | 5.81 | 0.31 | 82.22 | 4.11 | 5.81 | 0.31 | 79.72 | 4.11 | 5.81 | 0.31 | 76.75 | 4.11 | 5.81 | 0.31 | 96.2 | 4.11 | 5.81 | 0.31 | 79.59 | 4.11 | 5.81 | 0.31 | 79.61 | 4.11 | 5.81 | 0.31 | 79.66 | 4.11 | 5.81 | 0.31 | 79.5 | 4.11 | 5.81 | 0.31 | 79.91 | 4.11 | 5.81 | 0.31 | 79.6 |
| 9 | 4.98 | 6.6 | 0.37 | 159.15 | 4.98 | 6.6 | 0.37 | 159.23 | 6.27 | 8.18 | 0.5 | 159.19 | 5.99 | 8.09 | 0.35 | 5.02 | 6.27 | 8.18 | 0.5 | 119.93 | 4.98 | 6.6 | 0.37 | 176.44 | 4.98 | 6.6 | 0.37 | 169 | 4.98 | 6.6 | 0.37 | 164 | 4.98 | 6.6 | 0.37 | 193.88 | 4.98 | 6.6 | 0.37 | 196.31 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

Table A.22: FEDOT with four training years and seasonality (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | | Run 5 | | | | Run 6 | | | | Run 7 | | | | Run 8 | | | | Run 9 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 6.27 | 8.18 | 0.5 | 5.16 | 5.97 | 8.05 | 0.35 | 5.03 | 5.97 | 8.05 | 0.35 | 5.02 | 5.97 | 8.05 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5.01 | 3.97 | 5.77 | 0.28 | 244.14 | 3.97 | 5.77 | 0.28 | 230.78 | 3.97 | 5.77 | 0.28 | 230.42 | 3.97 | 5.77 | 0.28 | 230.45 |
| 1 | 5.95 | 8 | 0.36 | 5.02 | 5.99 | 8.09 | 0.35 | 5.01 | 5.97 | 8.05 | 0.35 | 5 | 3.97 | 5.5 | 0.27 | 80.33 | 3.97 | 5.5 | 0.27 | 80.33 | 3.97 | 5.5 | 0.27 | 77.92 | 3.97 | 5.5 | 0.27 | 99.79 | 3.97 | 5.5 | 0.27 | 73.22 | 3.97 | 5.5 | 0.27 | 72.79 | 3.97 | 5.5 | 0.27 | 76.55 |
| 2 | 5.99 | 8.09 | 0.35 | 5.02 | 4.09 | 5.78 | 0.31 | 115.38 | 4.09 | 5.78 | 0.31 | 125.02 | 4.09 | 5.78 | 0.31 | 117.07 | 4.09 | 5.78 | 0.31 | 125.13 | 4.09 | 5.78 | 0.31 | 130.32 | 4.09 | 5.78 | 0.31 | 117.78 | 4.09 | 5.78 | 0.31 | 118.68 | 4.09 | 5.78 | 0.31 | 142.21 | 4.09 | 5.78 | 0.31 | 119.3 |
| 3 | 4.98 | 6.6 | 0.37 | 5.02 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.02 | 4.98 | 6.6 | 0.37 | 5.01 | 4.98 | 6.6 | 0.37 | 5.01 |
| 4 | 4.11 | 5.83 | 0.31 | 376.91 | 4.11 | 5.83 | 0.31 | 388.82 | 5.95 | 8 | 0.36 | 5.03 | 5.99 | 8.09 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5 | 5.99 | 8.09 | 0.35 | 5.02 | 5.99 | 8.09 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5.01 | 4.11 | 5.83 | 0.31 | 429.02 | 4.11 | 5.83 | 0.31 | 382.49 |
| 5 | 4.98 | 6.6 | 0.37 | 6.15 | 4.98 | 6.6 | 0.37 | 5.83 | 4.98 | 6.6 | 0.37 | 5.79 | 4.98 | 6.6 | 0.37 | 5.81 | 4.98 | 6.6 | 0.37 | 5.77 | 4.98 | 6.6 | 0.37 | 5.83 | 4.98 | 6.6 | 0.37 | 5.79 | 4.98 | 6.6 | 0.37 | 5.78 | 4.98 | 6.6 | 0.37 | 5.88 | 4.98 | 6.6 | 0.37 | 5.78 |
| 6 | 4.64 | 6.28 | 0.33 | 5.02 | 4.64 | 6.28 | 0.33 | 5.02 | 4.61 | 6.25 | 0.33 | 5.02 | 4.61 | 6.24 | 0.33 | 5.02 | 4.64 | 6.28 | 0.33 | 5.02 | 4.61 | 6.24 | 0.33 | 5.02 | 5.7 | 7.5 | 0.44 | 5 | 4.69 | 6.34 | 0.34 | 5.03 | 4.64 | 6.28 | 0.33 | 5.01 | 4.65 | 6.29 | 0.33 | 5.01 |
| 7 | 4.98 | 6.6 | 0.37 | 9.55 | 4.98 | 6.6 | 0.37 | 9.55 | 4.98 | 6.6 | 0.37 | 9.43 | 4.98 | 6.6 | 0.37 | 9.51 | 4.98 | 6.6 | 0.37 | 9.65 | 4.98 | 6.6 | 0.37 | 9.36 | 4.98 | 6.6 | 0.37 | 9.41 | 4.98 | 6.6 | 0.37 | 9.46 | 4.98 | 6.6 | 0.37 | 9.58 | 4.98 | 6.6 | 0.37 | 9.53 |
| 8 | 4.11 | 5.81 | 0.31 | 94.86 | 4.11 | 5.81 | 0.31 | 71.76 | 5.99 | 8.09 | 0.35 | 5.02 | 5.99 | 8.09 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5.02 | 5.97 | 8.05 | 0.35 | 5.02 | 5.99 | 8.09 | 0.35 | 5.01 | 5.99 | 8.09 | 0.35 | 5 | 4.11 | 5.81 | 0.31 | 89.16 | 4.11 | 5.81 | 0.31 | 79.88 |
| 9 | 4.98 | 6.6 | 0.37 | 158.4 | 4.98 | 6.6 | 0.37 | 172.05 | 4.98 | 6.6 | 0.37 | 187.12 | 4.98 | 6.6 | 0.37 | 172.52 | 4.98 | 6.6 | 0.37 | 172.52 | 4.98 | 6.6 | 0.37 | 172.52 | 4.98 | 6.6 | 0.37 | 172.52 | 4.98 | 6.6 | 0.37 | 172.52 | 4.98 | 6.6 | 0.37 | 172.52 | 4.98 | 6.6 | 0.37 | 172.52 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

Table A.23: FEDOT with two training years and increased tendency (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 5.39 | 6.81 | 0.13 | 5.71 | 5.33 | 6.65 | 0.13 | 5.02 | 5.33 | 6.65 | 0.13 | 5.01 | 5.39 | 6.81 | 0.13 | 5.78 | 5.33 | 6.65 | 0.13 | 5.01 |
| 1 | 5.33 | 6.65 | 0.13 | 5.02 | 5.33 | 6.65 | 0.13 | 5.01 | 5.64 | 7 | 0.15 | 5.01 | 5.64 | 7 | 0.15 | 5.01 | 5.64 | 7 | 0.15 | 5 |
| 2 | 5.33 | 6.65 | 0.13 | 5.01 | 5.32 | 6.64 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5.01 | 5.33 | 6.65 | 0.13 | 5.01 | 5.33 | 6.65 | 0.13 | 5.01 |
| 3 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5.01 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5.01 |
| 4 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5.01 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5.01 | 5.33 | 6.65 | 0.13 | 5.01 |
| 5 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5.02 | 5.33 | 6.65 | 0.13 | 5.01 |
| 6 | 5.33 | 6.65 | 0.13 | 5 | 5.33 | 6.65 | 0.13 | 5.01 | 5.59 | 6.95 | 0.15 | 5.01 | 5.33 | 6.65 | 0.13 | 5.01 | 5.59 | 6.95 | 0.15 | 5.01 |
| 7 | 5.33 | 6.65 | 0.13 | 5 | 5.61 | 6.97 | 0.15 | 5.01 | 5.33 | 6.65 | 0.13 | 5.01 | 5.64 | 7 | 0.15 | 5 | 5.32 | 6.64 | 0.13 | 5.02 |
| 8 | 5.32 | 6.64 | 0.13 | 5.01 | 5.32 | 6.64 | 0.13 | 5 | 5.32 | 6.64 | 0.13 | 5 | 5.32 | 6.64 | 0.13 | 5.01 | 5.61 | 6.97 | 0.15 | 5 |
| 9 | 5.32 | 6.64 | 0.13 | 5.01 | 5.65 | 7.1 | 0.15 | 5.35 | 5.65 | 7.1 | 0.15 | 5.07 | 5.65 | 7.1 | 0.15 | 5.17 | 5.65 | 7.1 | 0.15 | 5.1 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

Table A.24: FEDOT with three training years and increased tendency (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 5.52 | 7 | 0.13 | 7.32 | 5.52 | 7 | 0.13 | 7.19 | 5.52 | 7 | 0.13 | 7.16 | 5.52 | 7 | 0.13 | 7.19 | 5.52 | 7 | 0.13 | 7.17 |
| 1 | 5.43 | 6.81 | 0.14 | 5.08 | 5.43 | 6.81 | 0.14 | 4.91 | 5.43 | 6.81 | 0.14 | 4.88 | 5.43 | 6.81 | 0.14 | 4.88 | 5.43 | 6.81 | 0.14 | 4.89 |
| 2 | 5.54 | 6.93 | 0.14 | 7.83 | 5.54 | 6.93 | 0.14 | 8.09 | 5.54 | 6.93 | 0.14 | 8.01 | 5.54 | 6.93 | 0.14 | 8.04 | 5.54 | 6.93 | 0.14 | 8.01 |
| 3 | 5.31 | 6.61 | 0.14 | 6.38 | 5.31 | 6.61 | 0.14 | 6.19 | 5.31 | 6.61 | 0.14 | 6.2 | 5.31 | 6.61 | 0.14 | 6.18 | 5.31 | 6.61 | 0.14 | 6.18 |
| 4 | 5.57 | 6.97 | 0.15 | 269.33 | 5.57 | 6.97 | 0.15 | 253.62 | 5.57 | 6.97 | 0.15 | 279 | 5.57 | 6.97 | 0.15 | 267.32 | 5.57 | 6.97 | 0.15 | 267.32 |
| 5 | 5.52 | 6.98 | 0.14 | 140.43 | 5.52 | 6.98 | 0.14 | 139 | 5.52 | 6.98 | 0.14 | 129.34 | 5.52 | 6.98 | 0.14 | 136.26 | 5.52 | 6.98 | 0.14 | 136.26 |
| 6 | 5.54 | 6.93 | 0.14 | 6.36 | 5.54 | 6.93 | 0.14 | 5.98 | 5.54 | 6.92 | 0.14 | 5.9 | 5.54 | 6.92 | 0.14 | 5.99 | 5.54 | 6.92 | 0.14 | 5.93 |
| 7 | 5.52 | 6.9 | 0.14 | 4.98 | 5.52 | 6.9 | 0.14 | 4.96 | 5.52 | 6.9 | 0.14 | 4.97 | 5.52 | 6.9 | 0.14 | 4.95 | 5.52 | 6.9 | 0.14 | 4.95 |
| 8 | 5.59 | 6.93 | 0.15 | 5 | 5.59 | 6.93 | 0.15 | 5.02 | 5.59 | 6.93 | 0.15 | 5.01 | 5.59 | 6.93 | 0.15 | 5 | 5.59 | 6.93 | 0.15 | 5.01 |
| 9 | 5.54 | 6.93 | 0.14 | 154.12 | 5.54 | 6.93 | 0.14 | 176.5 | 5.54 | 6.93 | 0.14 | 174.44 | 5.54 | 6.93 | 0.14 | 168.35 | 5.54 | 6.93 | 0.14 | 168.35 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

Table A.25: FEDOT with four training years and increased tendency (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 5.56 | 7.07 | 0.13 | 30.17 | 5.56 | 7.07 | 0.13 | 25.2 | 5.56 | 7.07 | 0.13 | 25.18 | 5.56 | 7.07 | 0.13 | 24 | 5.56 | 7.07 | 0.13 | 25.24 |
| 1 | 5.47 | 6.83 | 0.14 | 7.56 | 5.47 | 6.83 | 0.14 | 7.4 | 5.47 | 6.83 | 0.14 | 7.44 | 5.47 | 6.83 | 0.14 | 7.41 | 5.47 | 6.83 | 0.14 | 7.46 |
| 2 | 5.39 | 6.72 | 0.14 | 29.2 | 5.39 | 6.72 | 0.14 | 29.71 | 5.39 | 6.72 | 0.14 | 28.66 | 5.39 | 6.72 | 0.14 | 29.19 | 5.39 | 6.72 | 0.14 | 29.19 |
| 3 | 5.29 | 6.62 | 0.13 | 5.05 | 5.29 | 6.62 | 0.13 | 5.01 | 5.29 | 6.62 | 0.13 | 5 | 5.29 | 6.62 | 0.13 | 5.02 | 5.29 | 6.62 | 0.13 | 5.02 |
| 4 | 5.44 | 6.81 | 0.14 | 804 | 5.44 | 6.81 | 0.14 | 829 | 5.44 | 6.81 | 0.14 | 816.5 | 5.44 | 6.81 | 0.14 | 816.5 | 5.44 | 6.81 | 0.14 | 816.5 |
| 5 | 5.77 | 7.33 | 0.14 | 5.55 | 5.77 | 7.33 | 0.14 | 4.81 | 5.77 | 7.33 | 0.14 | 4.81 | 5.77 | 7.33 | 0.14 | 4.93 | 5.77 | 7.33 | 0.14 | 5.09 |
| 6 | 5.41 | 6.77 | 0.14 | 5.02 | 5.44 | 6.79 | 0.14 | 5.02 | 5.4 | 6.75 | 0.14 | 5.02 | 5.45 | 6.8 | 0.14 | 185.43 | 5.45 | 6.8 | 0.14 | 187.23 |
| 7 | 5.48 | 6.85 | 0.14 | 26.69 | 5.48 | 6.85 | 0.14 | 26.14 | 5.48 | 6.85 | 0.14 | 26.09 | 5.48 | 6.85 | 0.14 | 25 | 5.48 | 6.85 | 0.14 | 29.99 |
| 8 | 5.52 | 6.86 | 0.14 | 11.43 | 5.52 | 6.86 | 0.14 | 15.89 | 5.52 | 6.86 | 0.14 | 17.2 | 5.52 | 6.86 | 0.14 | 17.57 | 5.52 | 6.86 | 0.14 | 2.51 |
| 9 | 5.39 | 6.72 | 0.14 | 396.94 | 5.39 | 6.72 | 0.14 | 378.49 | 5.39 | 6.72 | 0.14 | 387.15 | 5.39 | 6.72 | 0.14 | 387.15 | 5.39 | 6.72 | 0.14 | 387.15 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

Table A.26: FEDOT with two training years, seasonality and increased tendency (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 13.37 | 19.04 | 0.47 | 6.44 | 13.37 | 19.04 | 0.47 | 6.21 | 18.85 | 21.91 | 0.75 | 5.91 | 13.37 | 19.04 | 0.47 | 6.25 | 13.37 | 19.04 | 0.47 | 5.79 |
| 1 | 33.36 | 39.13 | 1.78 | 5.36 | 17.25 | 20.28 | 0.73 | 84.1 | 29.22 | 37.15 | 1.48 | 5.17 | 12.74 | 17.37 | 0.52 | 12.24 | 36.79 | 40.76 | 1.84 | 6.62 |
| 2 | 35.12 | 44.74 | 1.92 | 6.1 | 32.23 | 41.4 | 1.67 | 5.47 | 18.56 | 22.48 | 0.79 | 6.15 | 34.09 | 43.85 | 1.81 | 5.54 | 34.09 | 43.85 | 1.81 | 5.6 |
| 3 | 13.49 | 19.12 | 0.47 | 5.96 | 33.37 | 38.05 | 1.57 | 5.78 | 92.9 | 111.97 | 5.47 | 51.05 | 13.49 | 19.12 | 0.47 | 6.36 | 32.43 | 41.75 | 1.61 | 5.18 |
| 4 | 32.19 | 41.4 | 1.67 | 5.6 | 93.35 | 11.39 | 5.21 | 5.9 | 14.86 | 19.75 | 0.54 | 6.1 | 93.55 | 113.65 | 5.22 | 5.92 | 128.02 | 158.65 | 6.66 | 6.71 |
| 5 | 32.59 | 42.13 | 1.72 | 5.54 | 34.82 | 44.21 | 1.82 | 73.62 | 29.09 | 30.61 | 1.37 | 5.13 | 25.21 | 32.98 | 2.48 | 43.22 | 29.09 | 30.61 | 1.37 | 5.13 |
| 6 | 28.25 | 38.95 | 1.24 | 10.15 | 23.13 | 32.24 | 1.18 | 11.93 | 33.51 | 40.91 | 1.97 | 7.73 | 87 | 110.84 | 6.37 | 5.59 | 33.72 | 42.15 | 1.77 | 5.23 |
| 7 | 32.78 | 42.01 | 1.69 | 5.9 | 36.33 | 46.11 | 1.95 | 5.5 | 32.26 | 41.45 | 1.67 | 5.36 | 27.33 | 31.35 | 1.32 | 6.53 | 32.26 | 41.45 | 1.67 | 5.38 |
| 8 | 12.27 | 15.89 | 0.36 | 7.87 | 21.4 | 23.88 | 0.91 | 5.59 | 32.25 | 41.44 | 1.68 | 5.37 | 35.11 | 44.69 | 1.85 | 5.62 | 19.28 | 22.18 | 0.8 | 29.45 |
| 9 | 146.49 | 184.88 | 7.48 | 5.66 | 146.49 | 184.88 | 7.48 | 5.68 | 146.49 | 184.88 | 7.48 | 5.74 | 146.49 | 184.88 | 7.48 | 5.69 | 146.49 | 184.88 | 7.48 | 5.62 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

Table A.27: FEDOT with three training years, seasonality and increased tendency (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 18.53 | 25.4 | 0.61 | 7.43 | 18.53 | 25.4 | 0.61 | 7.25 | 18.53 | 25.4 | 0.61 | 7.31 | 18.53 | 25.4 | 0.61 | 7.28 | 18.53 | 25.4 | 0.61 | 7.27 |
| 1 | 14.32 | 18 | 0.63 | 6.3 | 14.32 | 18 | 0.63 | 6.61 | 14.31 | 17.98 | 0.63 | 6.69 | 14.31 | 17.79 | 0.63 | 6.65 | 14.3 | 18.02 | 0.62 | 6.24 |
| 2 | 10.28 | 14.06 | 0.31 | 8.81 | 10.28 | 14.06 | 0.31 | 10.05 | 10.28 | 14.06 | 0.31 | 9.77 | 10.28 | 14.06 | 0.31 | 7.41 | 10.28 | 14.06 | 0.31 | 7.38 |
| 3 | 10.72 | 14.6 | 0.31 | 226.73 | 10.45 | 14.32 | 0.32 | 35.34 | 10.45 | 14.32 | 0.32 | 34.11 | 10.72 | 14.6 | 0.31 | 194.01 | 10.72 | 14.6 | 0.31 | 226.36 |
| 4 | 9.45 | 12.59 | 0.26 | 757.28 | 9.45 | 12.59 | 0.26 | 778.62 | 9.45 | 12.59 | 0.26 | 767.95 | 9.45 | 12.59 | 0.26 | 767.95 | 9.45 | 12.59 | 0.26 | 767.95 |
| 5 | 10.2 | 13.54 | 0.03 | 5.28 | 10.2 | 13.54 | 0.3 | 138.68 | 10.2 | 13.54 | 0.3 | 71.98 | 10.2 | 13.54 | 0.03 | 71.98 | 10.2 | 13.54 | 0.03 | 71.98 |
| 6 | 14.32 | 18.01 | 0.63 | 7.3 | 14.32 | 18.01 | 0.63 | 5.87 | 14.32 | 18.01 | 0.63 | 6.07 | 14.32 | 18.01 | 0.63 | 6.04 | 14.32 | 18.01 | 0.63 | 6.12 |
| 7 | 13.91 | 17.42 | 0.65 | 6.44 | 13.91 | 17.42 | 0.65 | 6.24 | 13.91 | 17.42 | 0.65 | 6.25 | 13.91 | 17.42 | 0.65 | 6.21 | 13.91 | 17.42 | 0.65 | 6.27 |
| 8 | 11.25 | 14.92 | 0.31 | 15.92 | 11.25 | 14.92 | 0.31 | 15.93 | 11.25 | 14.92 | 0.31 | 15.99 | 11.25 | 14.92 | 0.31 | 15.99 | 11.25 | 14.92 | 0.31 | 16.06 |
| 9 | 13.42 | 19.02 | 0.35 | 250.93 | 13.42 | 19.02 | 0.35 | 322 | 13.42 | 19.02 | 0.35 | 286.47 | 13.42 | 19.02 | 0.35 | 286.47 | 13.42 | 19.02 | 0.35 | 286.47 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

Table A.28: FEDOT with four training years, seasonality and tendency (all seeds and runs)

| Seed | Run 0 | | | | Run 1 | | | | Run 2 | | | | Run 3 | | | | Run 4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q | N | O | P | Q |
| 0 | 15.67 | 19.55 | 0.72 | 27.89 | 15.67 | 19.55 | 0.72 | 21.52 | 15.67 | 19.55 | 0.72 | 21.24 | 15.67 | 19.55 | 0.72 | 21.22 | 15.67 | 19.55 | 0.72 | 21.2 |
| 1 | 15.57 | 19.39 | 0.71 | 7.74 | 15.57 | 19.39 | 0.71 | 7.63 | 15.57 | 19.39 | 0.71 | 7.72 | 15.57 | 19.39 | 0.71 | 7.67 | 15.57 | 19.39 | 0.71 | 7.74 |
| 2 | 11.9 | 15.88 | 0.31 | 25.27 | 11.9 | 15.88 | 0.31 | 30.05 | 11.9 | 15.88 | 0.31 | 41.66 | 11.9 | 15.88 | 0.31 | 27.5 | 11.9 | 15.88 | 0.31 | 25.96 |
| 3 | 9.34 | 13.61 | 0.31 | 5.03 | 9.37 | 13.98 | 0.3 | 35.74 | 9.42 | 13.57 | 0.32 | 5.02 | 9.37 | 13.98 | 0.3 | 35.76 | 15.68 | 19.55 | 0.72 | 20.39 |
| 4 | 9.61 | 12.73 | 0.28 | 1733 | 9.61 | 12.73 | 0.28 | 1712 | 9.61 | 12.73 | 0.28 | 1722.5 | 9.61 | 12.73 | 0.28 | 1722.5 | 9.61 | 12.73 | 0.28 | 1722.5 |
| 5 | 15.61 | 19.5 | 0.71 | 9.55 | 15.61 | 19.5 | 0.71 | 8.98 | 15.61 | 19.5 | 0.71 | 8.21 | 15.61 | 19.5 | 0.71 | 8.65 | 15.61 | 19.5 | 0.71 | 7.5 |
| 6 | 18.13 | 23.22 | 0.86 | 13.74 | 18.13 | 23.22 | 0.86 | 13.49 | 18.13 | 23.22 | 0.86 | 13.24 | 18.13 | 23.22 | 0.86 | 13.41 | 18.13 | 23.22 | 0.86 | 13.19 |
| 7 | 15.18 | 18.82 | 0.74 | 21.94 | 15.18 | 18.82 | 0.74 | 24.4 | 15.18 | 18.82 | 0.74 | 26.96 | 15.18 | 18.82 | 0.74 | 36.22 | 15.18 | 18.82 | 0.74 | 30.02 |
| 8 | 13.21 | 17.71 | 0.33 | 17.66 | 13.21 | 17.71 | 0.33 | 16.89 | 13.21 | 17.71 | 0.33 | 16.44 | 13.21 | 17.71 | 0.33 | 16.7 | 13.21 | 17.71 | 0.33 | 16.44 |
| 9 | 10.74 | 14.05 | 0.3 | 628.15 | 10.74 | 14.05 | 0.3 | 686 | 10.74 | 14.05 | 0.3 | 657 | 10.74 | 14.05 | 0.3 | 657 | 10.74 | 14.05 | 0.3 | 657 |

N - MAE; O - RMSE; P - MAPE; Q - Time (seconds).

## A.4 CRYSTAL Decisions

Table A.29: Low imbalanced shifts

| Seed | April | May | June | July | August | September | October | November | December | January | February | March |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | D:6 | D:6 | D:6 | D:6 | D:9 | D:6 | D:9 | D:9 | D:9 | D:9 | D:9 | D:6 |
| **1** | D:7 | D:7 | D:7 | D:7 | D:6 | D:6 | D:8 | D:9 | D:9 | D:9 | D:9 | D:7 |
| **3** | D:6 | D:6 | D:6 | D:6 | D:6 | D:6 | D:9 | D:9 | D:9 | D:9 | D:9 | D:6 |
| **4** | D:6 | D:6 | D:6 | D:6 | D:6 | D:6 | D:9 | D:9 | D:9 | D:9 | D:9 | D:6 |
| **5** | D:6 | D:6 | D:6 | D:6 | D:6 | D:6 | D:9 | D:9 | D:9 | D:9 | D:9 | D:6 |
| **8** | D:6 | D:6 | D:6 | D:6 | D:9 | D:6 | D:9 | D:9 | D:9 | D:9 | D:9 | D:6 |
| **MV with AVG** | D:6 | D:6 | D:6 | D:6 | D:7 | D:6 | D:9 | D:9 | D:9 | D:9 | D:9 | D:6 |
| **Observed** | D:6 | D:7 | D:6 | D:6 | D:6 | D:6 | D:9 | D:9 | D:9 | D:9 | D:9 | D:6 |

D - Dismiss; H - Hire; M - Maintain; MV - Majority Voting; AVG - Average

Table A.30: Scenario 13

| Seed | April | May | June | July | August | September | October | November | December | January | February | March |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | H:3 | H:3 | H:3 | H:3 | M | H:3 | M | M | M | M | M | H:3 |
| **1** | H:3 | H:3 | H:3 | H:3 | H:3 | H:3 | M | M | M | M | M | H:3 |
| **3** | H:3 | H:3 | H:3 | H:3 | H:3 | H:3 | M | M | M | M | M | H:3 |
| **4** | H:3 | H:3 | H:3 | H:3 | H:3 | H:3 | M | M | M | M | M | H:3 |
| **5** | H:3 | H:3 | H:3 | H:3 | H:3 | H:3 | M | M | M | M | M | H:3 |
| **8** | H:3 | H:3 | H:3 | H:3 | H:3 | H:3 | M | M | M | M | M | H:3 |
| **MV with AVG** | H:3 | H:3 | H:3 | H:3 | H:3 | H:3 | M | M | M | M | M | H:3 |
| **Observed** | H:3 | H:2 | H:3 | H:3 | H:3 | H:3 | M | M | M | M | M | H:3 |

Scenario 13 - Three operators, one per shift. D - Dismiss; H - Hire; M - Maintain; MV - Majority Voting; AVG - Average

Table A.31: Scenario 14

| Seed | April | May | June | July | August | September | October | November | December | January | February | March |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | H:2 | H:2 | H:2 | H:2 | D:1 | H:2 | D:1 | D:1 | D:1 | D:1 | D:1 | H:2 |
| **1** | H:2 | H:2 | H:2 | H:2 | H:2 | H:2 | D:1 | D:1 | D:1 | D:1 | D:1 | H:2 |
| **3** | H:2 | H:2 | H:2 | H:2 | H:2 | H:2 | D:1 | D:1 | D:1 | D:1 | D:1 | H:2 |
| **4** | H:2 | H:2 | H:2 | H:2 | H:2 | H:2 | D:1 | D:1 | D:1 | D:1 | D:1 | H:2 |
| **5** | H:2 | H:2 | H:2 | H:2 | H:2 | H:2 | D:1 | D:1 | D:1 | D:1 | D:1 | H:2 |
| **8** | H:2 | H:2 | H:2 | H:2 | H:2 | H:2 | D:1 | D:1 | D:1 | D:1 | D:1 | H:2 |
| **MV with AVG** | H:2 | H:2 | H:2 | H:2 | H:2 | H:2 | D:1 | D:1 | D:1 | D:1 | D:1 | H:2 |
| **Observed** | H:2 | H:1 | H:2 | H:2 | H:2 | H:2 | D:1 | D:1 | D:1 | D:1 | D:1 | H:2 |

Scenario 14 - Shift 0 and 1 have one operator each while shift 2 has two. D - Dismiss; H - Hire; M - Maintain; MV - Majority Voting; AVG - Average

Table A.32: Scenario 15

| Seed | April | May | June | July | August | September | October | November | December | January | February | March |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | M | M | M | M | D:3 | M | D:3 | D:3 | D:3 | D:3 | D:3 | M |
| **1** | M | M | M | M | M | M | D:3 | D:3 | D:3 | D:3 | D:3 | M |
| **3** | M | M | M | M | M | M | D:3 | D:3 | D:3 | D:3 | D:3 | M |
| **4** | M | M | M | M | M | M | D:3 | D:3 | D:3 | D:3 | D:3 | M |
| **5** | M | M | M | M | M | M | D:3 | D:3 | D:3 | D:3 | D:3 | M |
| **8** | M | M | M | M | M | M | D:3 | D:3 | D:3 | D:3 | D:3 | M |
| **MV with AVG** | M | M | M | M | M | M | D:3 | D:3 | D:3 | D:3 | D:3 | M |
| **Observed** | M | D:2 | M | M | M | M | D:3 | D:3 | D:3 | D:3 | D:3 | M |

Scenario 15 - Shift 0 has one operator, shift 1 has two, and shift 2 has three operators. D - Dismiss; H - Hire; M - Maintain; MV - Majority Voting; AVG - Average