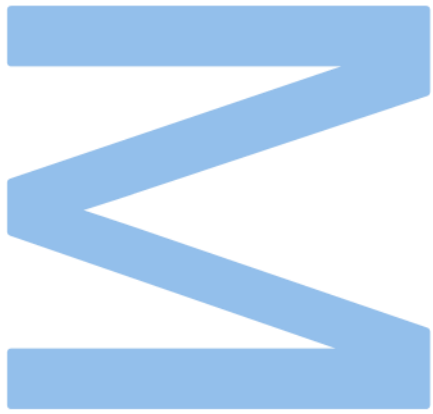
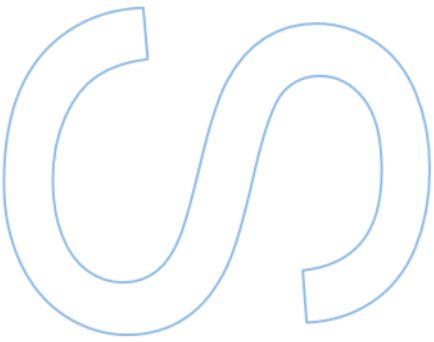
Optimizing Process Mining Algorithms: A Hyperparameter Tuning Approach



Simão Pedro Costa Cardoso
Master's Degree in Network and Information Systems Engineering
Department of Computer Science
2023

Supervisor

Ricardo Teixeira Sousa, Research Assistant, Laboratory of Artificial Intelligence and Decision Support at INESC TEC



Abstract

This thesis presents an in-depth exploration of hyperparameter tuning techniques applied to process mining algorithms, focusing on their application within the domain of document process management. The study addresses the pressing need for efficient and accurate process analysis and improvement in document-centric workflows.

The first part of the thesis delves into process mining fundamentals, covering essential topics such as event logs, process discovery, conformance checking, and process enhancement. Special emphasis is placed on the significance of process mining in the context of document management systems, highlighting the manual, time-consuming, and costly nature of traditional approaches.

The central contribution of this research lies in the comprehensive evaluation of hyperparameter tuning methods, including Genetic Algorithms, Random Search, and Bayesian Optimization, when applied to process mining algorithms. Through experimentation and analysis, we explore the impact of hyperparameter tuning on the performance of key process mining algorithms, namely the Inductive Miner and Heuristic Miner.

The findings of this thesis reveal that hyperparameter tuning significantly enhances the effectiveness of process mining algorithms in document process management. These results offer promising implications for organizations seeking to optimize their document-centric workflows, reduce operational costs, and enhance overall process efficiency.

Resumo

Esta tese apresenta uma exploração aprofundada das técnicas de ajuste de hiperparâmetros aplicadas a algoritmos de process mining, com foco na sua aplicação no domínio da gestão de processos documentais. O estudo aborda a necessidade de ótimização e análise de processos em fluxos de trabalho focados em documentos.

A primeira parte da tese explora os fundamentos de process mining, abrangendo tópicos essenciais como event logs, process discovery, conformance checking e process enhancement. É dada especial ênfase à importância de process mining no contexto dos sistemas de gestão documental, destacando a natureza manual, demorada e dispendiosa das abordagens tradicionais.

A contribuição central desta investigação reside na avaliação de métodos de ajuste de hiperparâmetros, incluindo *Genetic Algorithms*, *Random Search* e *Bayesian Optimization*, quando aplicados a algoritmos de process mining. Através de experimentação e análise, exploramos o impacto do ajuste de hiperparâmetros no desempenho de algoritmos de process mining, nomeadamente o *Inductive Miner* e o *Heuristic Miner*.

Os resultados desta tese revelam que o ajuste de hiperparâmetros melhora significativamente a eficácia dos algoritmos de process mining na gestão de processos documentais. Estes resultados oferecem implicações promissoras para organizações que procuram otimizar os seus fluxos de trabalho centrados em documentos, reduzir custos operacionais e aprimorar a eficiência global dos processos.

Agradecimentos

Chegar ao fim deste percurso na Faculdade de Ciências da Universidade do Porto representa o culminar de uma jornada repleta de desafios e aprendizagens. Neste momento especial, gostaria de expressar a minha gratidão a todas as pessoas que contribuíram para esta conquista.

À minha família, que sempre acreditou em mim, quero agradecer pelo apoio incondicional em todas as fases deste percurso académico. Em particular, à minha noiva Sónia, que esteve ao meu lado nos momentos mais desafiantes.

Agradecer também aos meus colegas e amigos de faculdade, em especial ao Ângelo e ao Eduardo, pelo apoio, pela aprendizagem e memórias que criámos ao longo destes anos.

Ao meu orientador, o Professor Ricardo Sousa, agradeço pela orientação valiosa e simpatia imudável.

À equipa da IPBrick, em especial ao Professor Raul Oliveira, agradeço pela colaboração, apoio e pela disponibilização dos dados cruciais para a realização deste trabalho.

Por fim, gostaria de agradecer a todos os meus colegas e professores da Faculdade de Ciências da Universidade do Porto. As vossas contribuições e ensinamentos foram fundamentais para o meu crescimento académico e pessoal.

Acknowledgements

This work was made in the context of project **SIGIPRO NORTE-01-0247-FEDER-069218**. All the data present in this work was provided by **IPBrick**.

Contents

List of Tables

List of Figures

Listings

Acronyms

BPMN Business Process Modeling Notation

FCUP Faculdade de Ciências da Universidade do Porto

DMS Document Management System

LIAAD Artificial Intelligence and Decision Support

XES Extensible Event Stream

WFN Workflow Net

Chapter 1

Introduction

1.1 Context

In the modern digital era, we have experienced a rapid growth in the volume of information, leading to an overwhelming influx of documents and files within today's systems. This abundance of data presents a significant challenge for companies when it comes to efficient management. Consequently, there is an increasing need to find swift and effective ways to handle the workflows and processes within organizations [2].

To address this challenge, many companies have shifted from a data-focused approach to a process-focused environment. This shift involves analyzing how an organization structures its work and allocates resources, including the sequence in which activities are carried out and the authorization of specific tasks to various groups of people [38].

As a result, precise process definitions have become essential for effectively organizing work within organizations. To aid in this endeavor, process-aware information systems, such as **workflow management systems**, have emerged. These systems play a crucial role in streamlining and optimizing the execution of tasks by automating processes and facilitating seamless collaboration among employees.

Process mining bridges the gap between the challenges posed by the exponential growth of information and the need for effective process management within organizations. It is an indispensable asset for workflow management systems, as it automates and streamlines a traditionally manual, time-consuming, and costly process. Without process mining, workflow management relies on manual intervention, making it a resource-intensive task that demands the expertise of specialists to design models effectively. By leveraging process mining, companies can gain valuable insights into their operational workflows and how information flows through various systems [2], enabling them to make data-driven decisions, streamline processes, and enhance overall productivity in today's dynamic and information-rich landscape.

By analyzing **event logs**, process mining techniques can reveal the underlying process flows,

bottlenecks, and inefficiencies, making it a crucial tool for process optimization and performance enhancement [8]. As organizations increasingly embrace digitalization and data-driven decision-making, process mining has gained significant traction in various industries, most notably healthcare [31] but also many other areas such as finance, manufacturing, supply chain and education [9] [18].

1.2 Motivation

Despite its promise, achieving accurate and actionable process mining results heavily depends on the appropriate selection of process mining algorithms and their **hyperparameters** [6]. Hyperparameters play a critical role in configuring the behavior and performance of process mining models, and determining factors such as model complexity, accuracy and generalization. However, finding the optimal hyperparameter values for process mining algorithms remains a challenging task due to the complexity and high-dimensional nature of event logs [4]. The traditional manual approach to hyperparameter tuning is often time-consuming and resource-intensive, limiting the scalability and efficiency of process mining applications.

To address these challenges, automated **hyperparameter tuning methods** have been proposed and successfully applied in various machine learning domains [14]. These methods, such as Bayesian Optimization and Grid Search, have demonstrated significant improvements in model performance and efficiency. However, the application of these techniques to the specific domain of process mining remains relatively under-explored.

This thesis aims to contribute to the field of process mining by exploring and evaluating effective hyperparameter tuning strategies for process mining algorithms. By automating and optimizing the hyperparameter selection process, we seek to enhance the accuracy and effectiveness of process mining results. The potential impact of this research extends beyond the academic realm, as the results and insights gained can be directly applied in practical settings.

This work was made possible thanks to the collaboration of Artificial Intelligence and Decision Support (LIAAD) of INESC TEC, Faculdade de Ciências da Universidade do Porto (FCUP) and IPBrick. Access to the iPortalDoc platform from IPBrick was essential. This platform is a document management system where it is possible to create manual workflows from a process.

This work was also made in the context of the **SIGIPRO** project, which aimed to leverage process mining techniques within a process management system to enhance the efficiency of handling processes within complex workflows.

1.3 Problem Statement

Process mining has emerged as a valuable approach to gain insights into complex business processes and improve organizational efficiency. By analyzing event logs, process mining algorithms can

automatically discover, monitor, and enhance process models based on real-world data. However, the accuracy and effectiveness of process mining heavily rely on the appropriate selection of hyperparameters for the underlying algorithms.

The **hyperparameter tuning** problem in process mining arises from the fact that different algorithms and techniques require specific parameter settings to achieve optimal performance (more generalized, simple and precise models) on different event logs. Selecting the right hyperparameter values is an important task due to the diversity and potentially high-dimensional nature of event logs.

The traditional manual approach to hyperparameter tuning often involves trial-and-error or expert knowledge, which is time-consuming and resource-intensive. Furthermore, the effectiveness of manually chosen hyperparameters is limited, as it may not be feasible to explore the entire hyperparameter space exhaustively. Consequently, sub-optimal hyperparameter configurations can lead to inaccurate process models, hindering the potential for valuable process insights and optimization opportunities. To address these challenges, there is a growing interest in automating the hyperparameter tuning process in process mining algorithms.

1.4 Dissertation Structure

This dissertation is structured into 5 chapters.

Chapter 1 - "Introduction": sets the stage by introducing the context and motivation, and problem statement behind the research. It outlines the significance of hyperparameter tuning for process mining algorithms and highlights the potential impact of automating this process.

Chapter 2 - "State of the art": presents an overview of relevant literature related to document process management, process mining, hyperparameter tuning, and their intersection. It discusses the existing approaches to hyperparameter tuning and how they can be implemented in process mining algorithms.

Chapter 3 - "Methodology": details the approach undertaken in this study. It outlines the process mining algorithms selected for investigation, the hyperparameters of interest, and the machine learning algorithms used for the automated tuning techniques. The chapter also describes the data used for experimentation and the evaluation metrics employed to assess the effectiveness of the proposed tuning strategies.

Chapter 4 - "Results": presents the results of the conducted experiments. It showcases the performance of the process mining approaches with different automated hyperparameter tuning algorithms. Additionally, this chapter discusses any challenges encountered during the tuning process and offers insights into the efficiency gains achieved.

Chapter 5 - "Conclusion": summarizes the results of the thesis and reiterates its significance in the field of process mining hyperparameter tuning. It reflects on the research objectives,

suggesting directions for future work.

Chapter 2

State of the art

This chapter will provide the main and fundamental concepts behind the research topics addressed in this dissertation in order to provide a clear understanding of the subjects that will be presented in the following chapters.

This chapter will start by covering the domain in which we are applying Process Mining to - document process management (Section 2.1), followed by Process Mining itself (Section 2.2). Within Process Mining we will look into Event Logs (Section 2.2.1) which are used as input for the different process mining techniques: discovery (Section 2.2.2), conformance (Section 2.2.3) and enhancement (Section 2.2.4). We will also explore the perspectives (Section 2.2.5), notations (Section 2.2.6), and evaluation metrics (Section 2.2.7) of Process Mining.

To finalize the State of the Art chapter there is going to be an overview of hyperparameter tuning in machine learning (Section 2.2.1), followed by the exploration of the hyperparameters of process mining discovery algorithms (Section 2.2.1), and a synthesis of the whole State of The Art (Section 2.5).

To retrieve the content needed to formulate this chapter it was performed keyword-based searches for papers, articles, and books from databases such as Engineering Village and Mendeley. Some of the keywords used were: ”process mining”, “process enhancement”, ”document and process management”, ”process models” and ”hyperparameter tuning”.

2.1 Document Process Management

The circulation of documents, as well as the processing of information, takes time and is prone to mistakes, leading to the loss of documents and, as a result, inefficiencies in the processes. A **Document Management System (DMS)** can help you avoid these problems [21].

The deployment of a DMS is increasingly required for a company’s proper functioning. This DMS helps businesses in managing information in a structured and centralized way, reducing losses of time and information. Work instructions can be obtained from processes that have

been recorded and used to instruct additional employees. Furthermore, proper documentation enables the analysis of modeled processes, allowing for the identification of bottlenecks and further optimization opportunities. Based on the process models, it is also possible to allocate roles and plan resources.

Companies who invest in a DMS get a return on investment in the form of increased productivity results due to improved document and process management efficiency [21]. Nonetheless, a traditional DMS is manual, making it a resource-intensive task that relies on a specialist to design models. To enhance the potential of a DMS we can employ the usage of process mining as it automates a traditionally slow and manual process.

2.2 Process Mining

Processes are an integral part of our daily routines, each designed with a specific purpose in mind. Understanding how these processes operate within a company serves as the initial step toward enhancing their efficiency [17]. However, grasping the intricacies of these processes and pinpointing areas of inefficiency can be quite a challenge. This is where process mining comes into play, offering a data-driven approach to dissecting and improving these workflows.

In many organizations, computer systems play a pivotal role in managing administrative tasks. These systems meticulously document how these tasks are executed and store this valuable information in databases [31], which are then transformed into event logs (2.2.1). Process mining harnesses the power of these event logs to unveil the inner workings of a company's operations, essentially turning raw data into a comprehensible narrative.

By utilizing event log data, process mining aligns operational activities with real-world practices, identifies opportunities for enhancement, and facilitates well-informed decision-making based on concrete evidence. This approach takes the hidden insights tucked away in event logs and presents them in a user-friendly format.

Process mining is a multidisciplinary blend of data science and process management techniques, designed to extract valuable insights and knowledge from datasets and **event logs**, ultimately resulting in the creation of **process models** (2.2.6). These tools and methodologies empower organizations to discover, model, monitor, and optimize their processes, thereby strengthening their control over these critical workflows [2].

Process mining bridges the gap by establishing connections between actual processes, recorded data, and process models [1]. This organized perspective on processes serves as an abstraction and simplification of the real processes and participant behaviors, making it easier to gather insights and make informed decisions. It all begins with the event log, which acts as the cornerstone of process mining. Most information systems store this data in an unstructured manner, but the feasibility of sequentially recording events associated with specific activities and process instances is key. These event logs often encompass a wealth of additional data, such as the

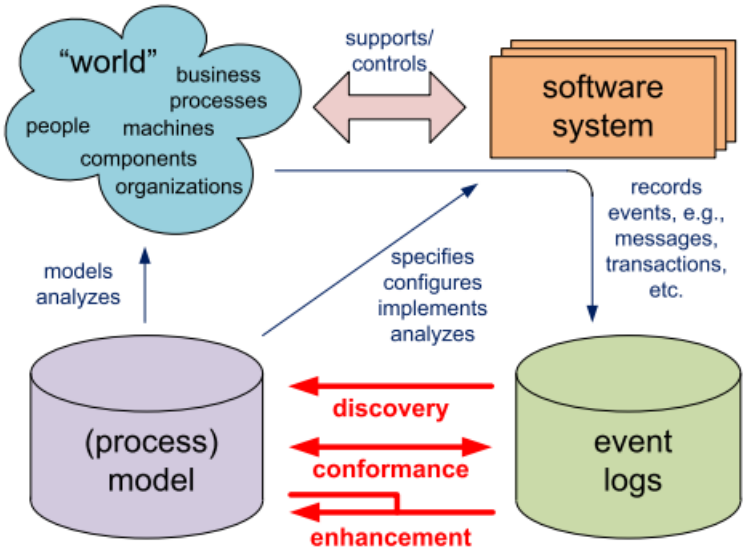


Figure 2.1: Process Mining (based on [4],[31])

resource handling the activity and timestamps, which process mining approaches frequently leverage to gain a comprehensive understanding of the workflows.

Event logs allow for the application of the recognized three types of process mining. **Process Discovery** (Section 2.2.2) involves creating a process model from an event log without any prior model. **Conformance Checking** (Section 2.2.3) examines the alignment between a process model and an event log, uncovering discrepancies. **Process Enhancement** (Section 2.2.4) focuses on refining an existing process model, aiming to improve it using both event logs and the original model. This work primarily centers on **Process Discovery**, while acknowledging the complementary nature of other process mining techniques such as Enhancement and Conformance Checking which derive substantial benefits from the insights gained through discovery.

2.2.1 Event Logs

An event log serves as a chronological record of events occurring within a process[4]. Each event signifies an action taken during a real-world process, resulting in changes within an information system. The widely accepted format for event logs in process mining is known as **Extensible Event Stream (XES)**. In 2016, it achieved the status of an official IEEE standard [30].

Consider the scenario where events can be recorded in sequence, with each event linked to a specific activity that belongs to a certain **case** (i.e., a process instance). This method creates a **sequence of events** for each process instance and is usually referred to as a *trace*. These events are denoted by **activity names**. Event logs typically encompass more than just activity names; they often include additional data like the resource, that indicates the person or system responsible for the execution of the action, the event's start and end timestamps, and other related information.

Crucially, in the realm of process mining, the event log is the primary input. The accuracy of this log significantly influences the outcome of the process mining resulting model. A poor-quality event log can lead to a convoluted and challenging-to-interpret model or even a model that fails to faithfully represent the true behavior of the business process [25]. In essence, a well-maintained and precise event log forms the foundation for effective process mining.

In process mining discovery, an event log typically requires three essential variables to capture and analyze process-related data effectively [2]:

1. **Case Identifier**: The Case Identifier is a unique identifier associated with each instance or case in your process. It allows you to group related events together and understand the flow of activities within a specific case.
2. **Activity Name**: This variable represents the name or label of the specific activity or event that occurred within the process. In the context of a DMS, an activity could correspond to actions like ”document creation,” ”review,” ”approval,” or ”archiving.” These labels provide insights into the different steps or actions taken in your process.
3. **Timestamp**: The Timestamp variable records the date and time when each activity or event occurred. It enables you to establish the chronological order of activities within a case and helps analyze the time taken for each step. This information is crucial for understanding process bottlenecks, delays, and performance metrics.

Consider Table 2.1, which depicts a portion of an event log, to help explain the concept of an event log. Note that all activities are already grouped per case. There are just two traces displayed, one with five occurrences and one with 3 occurrences.

	Case ID	Activity	Timestamp
0	17362	Aprovar encomenda a fornecedor	2017-01-27 12:28:19
1	17362	Reformular encomenda	2017-01-27 12:45:57
2	17362	Aprovar encomenda a fornecedor	2017-01-27 12:46:53
3	17362	Enviar email a fornecedor	2017-01-27 12:47:42
4	17362	Aguardar receçao de bens/ serviços	2017-01-27 12:47:52
5	17794	Aprovar encomenda a fornecedor	2017-03-16 17:02:42
6	17794	Enviar email a fornecedor	2017-03-16 17:03:21
7	17794	Aguardar receçao de bens/ serviços	2017-03-16 17:03:44

Table 2.1: Event log example.

2.2.2 Process Discovery

Process discovery is the first type of process mining. A discovery approach involves the automatic extraction of process models from event logs without requiring any prior knowledge. These

models enable organizations to visualize their processes and identify bottlenecks, inefficiencies, and deviations from the intended workflow, ultimately paving the way for process optimization and improvement initiatives. In essence, process discovery serves as the essential first step in the journey toward enhancing operational efficiency and achieving better process management.

There are several well-known and researched algorithms, such as **Alpha-Miner** [27], **Heuristic Miner** [37] and **Inductive Miner** [22]. These techniques take an event log and generate a process model that explains the behavior observed in that same log [4].

This section explores these process discovery techniques, their strengths, limitations, and the challenges associated with this essential step.

2.2.2.1 Challenges

There are some challenges that process discovery needs to face. The ability to deal with these challenges is essential to ensure the effective application of process mining within organizations. Here, we explore the key challenges that organizations may encounter during the process discovery phase:

- **Event Log Data Collection and Transformation:** A fundamental challenge lies in capturing event data within the DMS context and transforming it into a compatible event log format suitable for process mining. The events should be sequentially linked so that each event corresponds to an activity (a clearly defined step in some process), and is associated with a specific case, forming process instances that can be analyzed effectively [36].
- **Data Complexity and Quality:** One of the foremost challenges in process discovery is dealing with the complexity and quality of event log data. Real-world processes often generate vast amounts of data, which may be unstructured or poorly recorded. Ensuring data accuracy, completeness, and consistency of the event log directly impacts the reliability of the process models generated [36].
- **Overfitting and Underfitting:** Balancing the complexity of process models is critical. Overfitting, where a model is overly complex and fits the training data too closely, can result in models that do not generalize well to new data. Conversely, underfitting, where a model is too simplistic and fails to capture essential patterns, can lead to inaccurate representations of processes [1]. Striking the right balance between these extremes is a challenge in process discovery.
- **Unsupervised learning:** Process discovery presents an inherent challenge as it operates within the more complex realm of unsupervised learning. Event logs typically lack information about unsuccessful or prevented state transitions, adding to the intricacy of the unsupervised learning setting [19].

2.2.2.2 Techniques for Process Discovery

Process discovery is not a one-size-fits-all endeavor; there are many techniques, each with its own approach, strengths, and limitations. Understanding these techniques is essential for organizations seeking to uncover the hidden dynamics of their processes. Here, we delve into some of the primary process discovery techniques and shed light on their characteristics:

- **Alpha Algorithm:** This algorithm extracts process models by identifying dependencies between activities or events in the event logs. It looks for **frequent patterns** of activities occurring in a sequence and records these dependencies. This technique is renowned for its **simplicity** and effectiveness in revealing straightforward process flows with sequential dependencies.

  While it excels in scenarios where processes follow a linear sequence of activities with minimal branching or loops, the Alpha Algorithm may encounter challenges when dealing with more intricate and complex workflows. In such cases, it may struggle to accurately represent parallel activities or decision points, potentially leading to oversimplified process models [3].

  The Alpha-Miner algorithm is suitable for introductory process discovery tasks or when dealing with relatively simple and well-structured processes. However, for more complex processes, other algorithms like the Inductive Miner or Heuristic Miner may be preferred as they can capture a wider range of process behaviors.

- **Heuristic Miner**: The Heuristic Miner, which builds upon the foundations set by the Alpha Algorithm, relies on predefined heuristics to uncover process models. It excels at identifying frequently occurring behaviors within event logs, making it a valuable tool in process mining [4].

  Unlike the Alpha-Miner, which primarily focuses on capturing directly-follows dependencies, the Heuristic Miner goes a step further. It takes various types of dependencies between activities into account and assesses their significance based on frequency and importance [29]. By doing so, it can construct more accurate process models that are capable of capturing complex process behaviors.

  One notable strength of the Heuristic Miner lies in its ability to handle concurrency and occasional deviations in process execution. It prioritizes the identification of common patterns while eliminating less frequent ones, which makes it resilient against noise and incomplete logs. However, it's important to note that the Heuristic Miner doesn't guarantee process soundness. The Heuristic Miner's flexibility allows it to capture a wide range of process structures, including parallelism, loops, and more complex arrangements. This makes it particularly well-suited for processes with diverse and intricate behaviors [38].

- **Inductive Miner**: The Inductive Miner process begins with the abstraction of the event log into a Directly-Follows Graph (DFG). This graph represents the sequential relationships between activities in the log, revealing which activities often follow one another.

Next, the algorithm identifies specific points within the DFG where it can "cut" the graph into different sections. These cuts can represent various process patterns, such as sequences, concurrency, loops, and choices [29].

Once the cuts are defined, the Inductive Miner recursively applies itself to each section of the DFG, effectively creating sublogs for each section. Each sublog is treated as an independent process to be discovered (*divide and conquer strategy*). This recursive process continues until a base case is reached. In other words, the algorithm keeps dividing the process until it cannot be further divided or until specific stopping criteria are met [29].

Finally, the Inductive Miner integrates the process models created for each sublog into a single, comprehensive process model. This model reflects the overall behavior of the process as captured from the event log [4]. The Inductive Miner's strength lies in its ability to effectively capture different process patterns. However, it may produce more complex models in some cases.

2.2.3 Conformance Checking

While process discovery creates models solely from event logs, conformance checking takes a different route. It operates by comparing an existing process model with the observed behavior captured in an event log, to see how well the model aligns with actual process executions [3].

As processes grow in complexity and change more often, there is a pressing need to ensure that the models employed accurately represent current operations. However, models often diverge from reality or become outdated, leading to inefficiencies and misalignments [5, 32, 33]. This is precisely where conformance checking can be applied.

Conformance checking serves as a quality control mechanism. It assesses whether the recorded reality, as documented in the event log, lines up with the constructed process model and vice versa [4]. This examination helps uncover discrepancies, revealing where the model doesn't quite capture what's going on in the real world [13]. It's not just about finding differences; it also helps us understand how big these differences are and why they happen. This knowledge empowers organizations to identify and fix issues, ensuring that their processes stay efficient, effective, and in sync with their intended models.

Conformance checking compares a process model with the event log with four main quality dimensions in mind: **fitness**, **precision**, **simplicity** and **generalization**. These quality dimensions are addressed in the chapter 2.2.7.

2.2.4 Process Enhancement

The third type of process mining is process enhancement. The primary objective here is to extend or enhance an existing process model, drawing upon valuable insights from an event log that captures the real process in action.

Unlike conformance checking, which primarily focuses on ensuring that the model aligns with what's happening in reality, this form of process mining takes a proactive approach. It seeks to modify or expand the initial model, making it even more representative of the actual processes at play [3].

On one hand process enhancement allows us to **repair** the model. It involves making adjustments to the model to bring it closer to representing correctly reality. On the other hand, allows us to **extend** the model. This provides a fresh perspective on the process model by cross-referencing it with the event log [4]. For example, if we include timestamps in the event log, we can expand the process model to uncover bottlenecks and frequency patterns. This additional layer of data enriches the model, allowing organizations to gain deeper insights and identify areas for improvement.

2.2.5 Perspectives in Process Mining

Alongside the three different types of mining (discovery, conformance, and enhancement), there are also four **perspectives** that are usually identified [4]:

- The **control-flow perspective** is concerned with the ordering of activities, often known as control-flow. Finding an accurate description of all potential pathways is the aim of this perspective.
- The **organizational perspective** is concerned with the information about the resources that is concealed in the log, i.e., the agents (such as people or roles) that are involved and their connections. It's possible to obtain a depiction of the organization's structure by grouping people into roles or departments, or even the underlying social network.
- In the **case perspective**, case characteristics are emphasized. Specific values of cases can also be used to describe processes and influence a particular decision.
- The **time perspective** focuses on the timing and frequency of events. Events with timestamps allow for the detection of bottlenecks, measure service levels, track resource use, and estimate how long cases still in progress will take to be processed.

Process models might only show the control-flow. However, when extending process models, additional perspectives are added.

2.2.6 Process Model Notations

Process mining enables us to have a deeper knowledge of how a process is carried out. Therefore, in order to comprehend processes and reason about them, we need a way to communicate them, that is, a way to demonstrate a process that allows a consensus on how it should be carried out. To achieve this, process models are fundamental [17].

Process models help with complexity management by offering insight and documenting methods. Information systems require accurate configuration and operation. As a result, process models are now frequently employed in businesses [2].

There are several process modeling notations. As a result, we present only a few fundamental notations and do not intend to offer a comprehensive review of all existing process modeling notations. Each state has a unique label and transitions are represented by arcs.

- **Transition Systems**: The most basic process modeling notation is a Transition System [2]. A Transition System contains *states* and *transitions*.

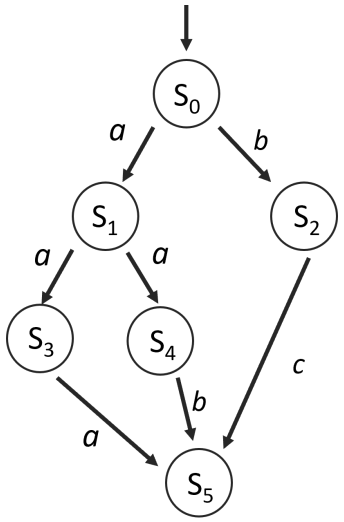


Figure 2.2: Transition System

A Transition System is a triplet $TS = (S, A, T)$ where S is the set of *states*, A is the set of *activities*, and T is the set of *transitions*. S^{start} is the set of initial states, and S^{end} is the set of final states.

The Transition System shown in **Fig. 2.2** can be described as follows:
$S = \{S_0, S_1, S_2, S_3, S_4, S_5\}$, $S^{start} = \{S_0\}$, $S^{end} = \{S_5\}$, $A = \{a, b, c\}$, and $T = \{(S_0, a, S_1), (S_0, b, S_2), (S_1, a, S_3), (S_1, a, S_4), (S_3, a, S_5), (S_4, b, S_5), (S_2, c, S_5)\}$.

Many Transition System concepts may be readily translated to higher-level languages like a Petri net. Transition systems are simple, but they have difficulty describing *concurrency* concisely. A Petri net with a large number of simultaneous activities would be significantly more compact than a corresponding Transition System. Given the concurrent nature of business processes, more expressive models such as Petri nets are required to appropriately describe the outputs of process mining [2].

- **Business Process Modeling Notation**: The Business Process Modeling Notation (BPMN) has become one of the most widely used languages to model business processes.

Figure 2.3 shows a subset of all notation elements. *Tasks* are what we call activities. There are different forms of split and join gateways: AND, XOR, and OR. The splits

are determined by the data parameters. An *event* is equivalent to a place in a Petri net although the semantics are significantly different. A *start* event has one outgoing arc and a *end* event has one incoming arc. Events with multiple incoming or outgoing arcs are not possible, as opposed to Petri nets. Gateways are required for splitting and joining.

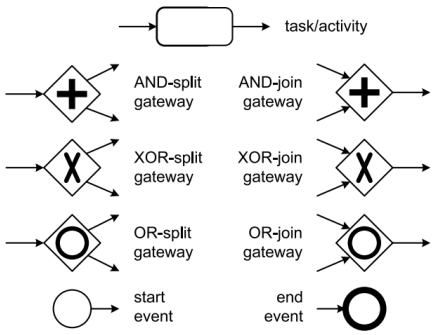


Figure 2.3: **BPMN** notation

- **Petri Nets**: Petri nets stand as the most venerable and extensively examined formalism for process modeling, enabling the representation of *concurrent* processes.

  A Petri net is a graph made up of *places* that may hold *tokens*. The presence of tokens in the different places define the net's state. The *transitions* affect the net's state by consuming and creating tokens from and to linked places [23]. The net's state is referred to as its *marking* and it corresponds to a multi-set of tokens.

  A Petri net is a triplet $N = (P, T, F)$ where P is a finite set of *places*, T is a finite set of *transitions*, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs that link *places* and *transitions* [20].

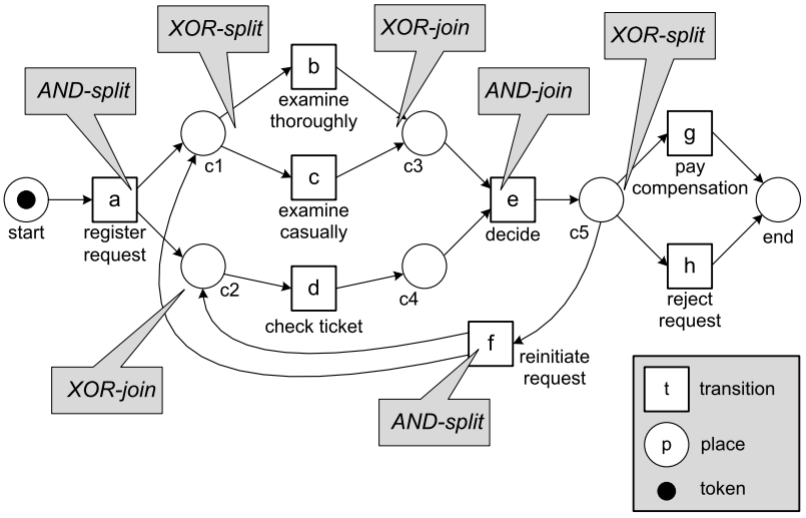


Figure 2.4: Petri net

The Petri net shown in **Fig. 2.4** can be described as follows:

$P = \{start, p_1, p_2, p_3, p_4, end\}$, $T = \{a, b, c, d, e\}$, and $F = \{(start, a),\ (a, p_1),\ (a, p_2),\ (p_1, b),\ (p_1, c),\ (p_2, d),\ (b, p_3),\ (c, p_3),\ (d, p_4),\ (p_3, e),\ (p_4, e)\ (e, end)\}$.

The marking shown in **Fig. 2.4** is $[start]$. The **firing rule** defines the flow of a Petri net. If each of its input places has a token, the transition is enabled. An enabled transition can fire, consuming one token from each input place and creating one token for each output place. As a result, in the Fig. 2.4 (with the marking [start]) only the transition a is enabled. When you fire a, you get the marking $[c1, c2]$. Transition a is no longer enabled. Transitions b, c, and d, on the other hand, have become available. Now, firing b resulted in marking $[c2, c3]$. d is still enabled here, but b and c are not.

- **Workflow Net**: In the context of Petri net based business process modeling, it is common to delve into a specific subclass of Petri nets referred to as Workflow Net (WFN). In a WFN, there is a single *source* with no incoming transitions (the start of the process) and a single *sink* with no outgoing transitions (the end of the process), and each place and transition is on an ordered path between these two places [23]. The petri net presented in Fig. 2.4 is also a WFN.

  Not every WFN represents a correct process. A process depicted using a WFN can exhibit errors such as deadlocks, activities that can never become active and livelocks. To avoid those, there is a correctness criterion that helps us to assure the model is faultless named ***soundness***. It is an aggregation of the following properties [2]:

  - *Safeness*: Each place can only hold a single token at the same time;
  - *Proper completion*: If the sink place is marked, then all the other places must not contain a token;
  - *Option to complete*: It must always be possible for the sink place to be marked.
  - *Absence of dead parts*: For any transition in the model, there must always be a sequence of events that lead to it occurring.

  The WFN presented in Fig. 2.4 is **sound**.

In this work, we took the sensible approach of selecting the most common and well accepted process model notation for our goal, **Petri net**, specifically, WFN. Petri nets are backed up by a multitude of techniques and tools. Many additional process modeling languages can also be translated/converted into Petri nets and vice versa. As a result, Petri nets are effective representations of current process model notations.

2.2.7 Process Mining Evaluation

There are different **quality measures** to assess the process models resulting from a process mining execution [4]. The four widely accepted quality measures are:

- **Fitness**: Fitness measures how well the discovered model fits the actual event log. It **quantifies** the extent to which the model can reproduce the **observed behavior**. This metric is calculated with the following formula [26]:

$$f(\sigma, N) = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}\right)$$

  Assuming N is a WFN, Σ represents the summation of all tokens produced, consumed, missing, and remaining, applying the same formula. Let $p_{N,\sigma}$ denote the count of tokens produced when replaying σ on N. The $m_{N,\sigma}$ stands for the number of missing tokens, representing an alternative duplicate task that is never repeated within the same sequence. The $c_{N,\sigma}$ represents the count of consumed tokens, while $r_{N,\sigma}$ denotes the count of remaining tokens. When the fitness is equal to 1, it signifies that the discovery process model can successfully replay all traces in the event log [26].

- **Precision**: Precision deals with the problem of **underfitting** a model. It assesses the accuracy of the model in predicting the occurrence of specific events or transitions. It calculates the ratio of correctly predicted events to the total predicted events. This metric is calculated with the following formula [26]:

$$\alpha_B = 1 - \frac{\sum_{i=1}^{k} n_i x_i}{(m-1)\sum_{i=1}^{k} n_i}$$

  Let k represent the number of distinct traces within the event log. For each trace within the log, denoted as $1 \leq i \leq k$, n_i signifies the count of process instances aggregated into the current trace. Additionally, x_i represents the average number of enabled transitions during log replay. It's important to note that m denotes the count of labeled tasks (excluding invisible tasks), and we assume that $m > 1$ [26].

- **Simplicity**: Simplicity measures the complexity of the discovered model. It **evaluates** how easily the model can be **understood** and **maintained**. This metric is calculated with the following formula [26]:

$$\partial s = \frac{|T| - |T_{DT}| + |T_{IT}|}{T}$$

  Let T denote the total number of transitions in the model. $T_{DT} \subseteq T$ represents the count of alternative duplicate tasks, while $T_{IT} \subseteq T$ represents the count of redundant invisible tasks. Alternative duplicate tasks refer to tasks that are never repeated simultaneously within a single sequence. On the other hand, redundant invisible tasks are tasks that can be safely eliminated from the model without altering its behavior [26].

- **Generalization**: Generalization deals with the problem of **overfitting** a model. It evaluates the **ability** of the model to **generalize** from the observed data to potential future instances. It assesses whether the model can capture the underlying process patterns

without overfitting the specific event log. This metric is calculated with the following formula [26]:

$$Q_g = 1 - \frac{\sum_{nodes}(\sqrt{\#execution})^{-}1}{'node\_in\_model'}$$

The square root of the number of executions is utilized because it accounts for the fact that the impact of having 10 executions, as opposed to just 1, is perceived as a more substantial improvement than the transition from 10 to 100 executions. Subsequently, each of these values is raised to the power of -1 to standardize them within the range of 0 to 1. These standardized values are then summed and divided by the total number of nodes in the tree to obtain the average for the entire tree [26].

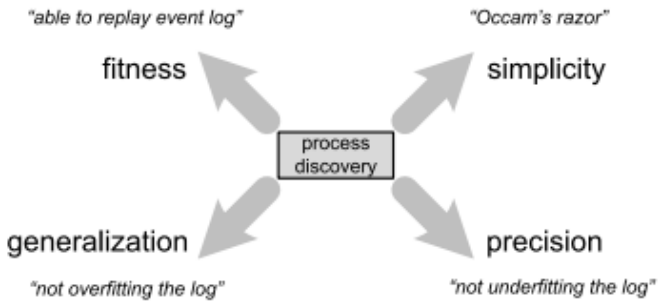


Figure 2.5: Quality Dimensions [4]

A model with high **fitness** effectively captures the majority of behaviors found in the event log. When a model can accurately replicate all sequences of events from start to finish in the log, it achieves optimal fitness. Naturally, the ideal model is the **simplest** model that can describe the behavior observed in the log. Occam's Razor is the name given to this notion.

However, assessing the quality of a process model requires more than just evaluating fitness and simplicity. For instance, it's possible to discover a model that not only replicates all traces in an event log but also works for any other event log with the same set of events. Conversely, a model that merely mimics the exact behaviors recorded in the event log may fall short, as the log might not encompass all potential workflows.

Precision in a model implies that it should exclude undesired behaviors. When a model overly simplifies the behavior seen in the log, it's referred to as underfitting. In contrast, a model should also **generalize** appropriately and not be excessively constrained by the examples in the log. Overfitting occurs when an overly specific model is constructed, disregarding the fact that the log represents only sample behaviors and not the entirety of potential workflows [3].

2.3 Hyperparameter Tuning in Machine Learning

Machine learning models are powerful tools for making predictions and extracting insights from data. However, the effectiveness of these models heavily depends on their hyperparameters. Hyperparameters are configuration settings that are not learned from the data but are crucial in determining the model's performance.

Hyperparameters are essential in shaping the behavior and performance of machine learning models. Finding the right hyperparameters can be challenging, as they often interact in complex ways, and their optimal values can vary depending on the dataset and problem at hand.

Hyperparameters can be configured through various methods when working with a specific dataset. One approach involves manual configuration, where hyperparameters are initially set, and their performance is evaluated accordingly. Subsequently, different hyperparameter values are tested, and for each adjustment, the corresponding performance is assessed. However, manually fine-tuning hyperparameter values in this trial-and-error manner can be a laborious and time-consuming process. Another method for determining suitable hyperparameter configurations is to opt for default values recommended by the software packages used in the implementation. While default values may perform well for certain datasets, it does not guarantee optimal results in all cases [12]. Alternatively, we can use hyperparameter optimization strategies. These strategies are data-dependent optimization algorithms, designed to minimize the expected error of a machine learning model over the hyperparameter search space of considered candidate configurations.

There are many methods that have been proposed to tackle the challenge of hyperparameter tuning with the aim of automating the process of finding the optimal hyperparameters.

- **Grid Search**: Grid search represents a straightforward yet efficient hyperparameter tuning approach. It entails defining a set of values for each hyperparameter and systematically evaluating all conceivable combinations. Although it guarantees a comprehensive exploration of possibilities, it may become computationally demanding, particularly for models featuring numerous hyperparameters.

  In this technique, a grid is constructed encompassing all potential combinations of the provided hyperparameter values, and each model's performance is assessed accordingly. Ultimately, the Grid Search algorithm identifies the configurations that yield the highest performance during the validation process. The selected hyperparameter values from the Grid Search are subsequently employed in the actual model. However, it can encounter challenges related to rapid convergence and high dimensionality [12].

- **Random Search**: Random Search offers an alternative to grid search by randomly sampling hyperparameters from predefined distributions. This approach can exhibit greater efficiency compared to grid search as it covers a wider spectrum of hyperparameter values and frequently converges towards effective solutions more swiftly.

  In Random Search, the total number of evaluations must be determined before commencing

the hyperparameter optimization process. Nevertheless, a notable limitation of the Random Search algorithm lies in its lack of utilization of information from previous trials to guide the selection of subsequent configurations. Additionally, it does not employ any predictive strategy for the next trial [12].

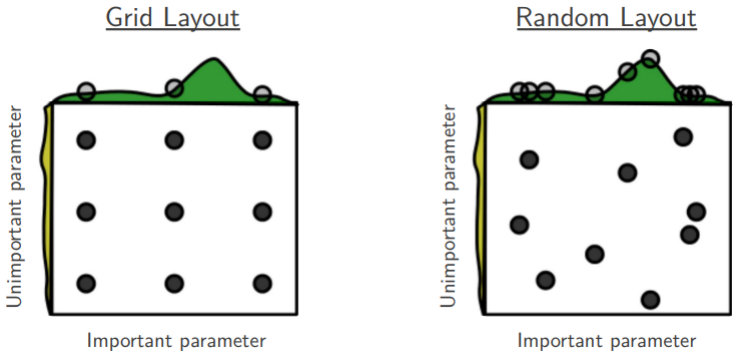


Figure 2.6: Grid Search vs Random Search [7]

- **Bayesian Optimization**: Bayesian Optimization is a powerful technique that harnesses probabilistic models to optimize the hyperparameters of machine learning models. It operates by dynamically selecting and evaluating different hyperparameter settings based on the predictions made by these probabilistic models. This approach is particularly effective in scenarios where the hyperparameter search space is extensive or evaluating each set of hyperparameters is computationally expensive.

  Bayesian Optimization stands out as an informed search algorithm. This means that it learns from previous iterations and leverages the results of one iteration to inform the next. While it shares some similarities with the Random Search method in that it also samples a subset of hyperparameter combinations, they differ significantly in how these combinations are chosen [12].

  In the context of Bayesian Optimization, the hyperparameter tuning process is framed as the optimization of a surrogate model to approximate the objective function. Among the various surrogate models available, the Gaussian Process is a commonly used choice. The process begins with a random selection of a few hyperparameter combinations, which are then evaluated to produce an initial model of the objective function. As Bayesian Optimization is an iterative process, it proceeds by exploring another subset of the hyperparameter search space in each subsequent iteration. The goal is to locate additional optimal points within this space based on the information obtained from previous evaluations. This strategy allows it to make informed decisions about which combinations to test next, significantly reducing the number of evaluations required [12].

  By iteratively refining its surrogate model and exploring the most promising regions of the hyperparameter space, Bayesian Optimization efficiently converges towards optimal hyperparameter configurations.

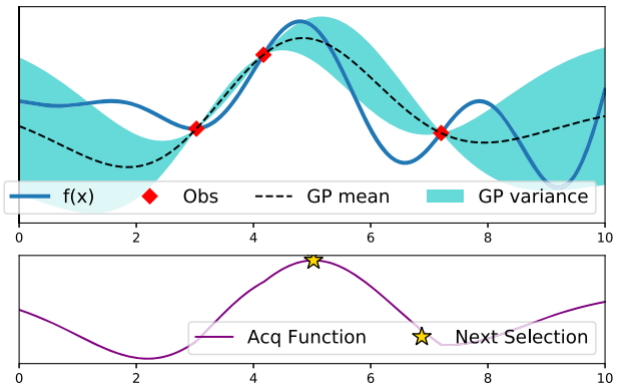


Figure 2.7: Example of Bayesian Optimization suggesting the next optimal point to evaluate [28]

- **Genetic Algorithms**: Genetic Algorithms are optimization techniques inspired by the principles of natural selection and evolution. In the context of hyperparameter tuning, Genetic Algorithms are employed to evolve a population of hyperparameter configurations iteratively. This process involves selecting the best-performing configurations and introducing variations through mutation and crossover operations to efficiently search for optimal hyperparameters [35].

  At the core of Genetic Algorithms lies the concept of evolution, where individuals with advantageous traits are more likely to survive and pass on their traits to the next generation. In the context of hyperparameter tuning, these traits correspond to the hyperparameter values. As the algorithm progresses through generations, it strives to produce configurations with improved performance while discarding less promising ones [39].

  In the application of Genetic Algorithms to hyperparameter optimization, several key operations are performed [35]:

  - **Selection**: Configurations with the best performance are selected to form the basis for the next generation. Selection is biased towards favoring configurations with superior results, increasing their likelihood of being passed to the next generation.
  - **Crossover**: Crossover is a genetic operation that combines characteristics of two or more configurations to generate new ones. In the context of hyperparameter tuning, crossover involves swapping portions of hyperparameter values between different configurations. This process encourages the mixing of solutions within the search space.
  - **Mutation**: Mutation is another operation that introduces randomness by modifying one or more hyperparameter values within a configuration. This randomness ensures diversity in the population and allows the algorithm to explore different regions of the search space.

  These operations work together to create new configurations in each generation, inheriting beneficial traits from their predecessors while introducing variations to explore uncharted regions of the hyperparameter space.

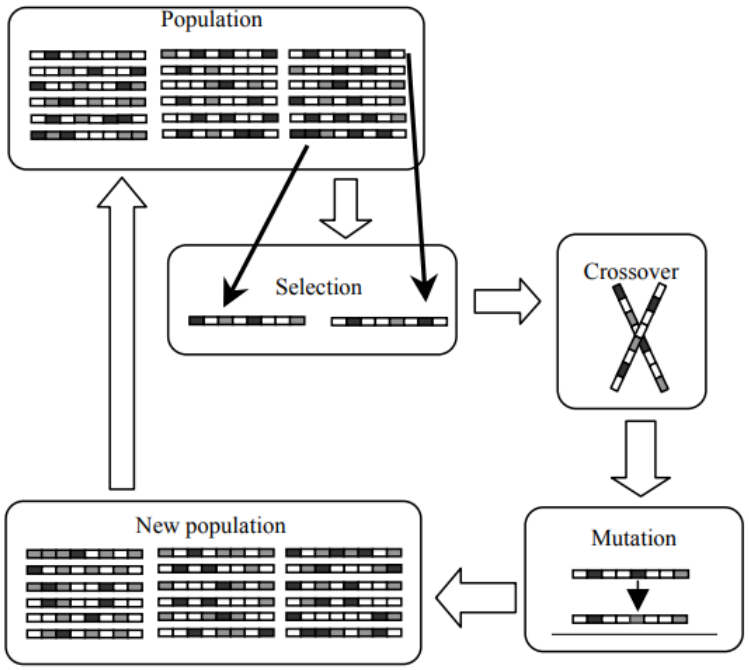


Figure 2.8: Genetic Algorithm flow [24]

Hyperparameter tuning is a critical step in the machine learning pipeline that can significantly impact model performance. Different methods offer various trade-offs in terms of computational cost and effectiveness.

2.4 Process Mining Algorithms and Hyperparameters

Process mining is a powerful approach for analyzing and improving business processes by extracting valuable insights from event data. To make the most of process mining, it is essential to understand the intricacies of the underlying algorithms and their hyperparameters. In this subchapter, we will explore the popular process mining algorithms available in the pm4py Python package. Namely, the Alpha-Miner, Inductive Miner, and Heuristic Miner, along with their associated hyperparameters [17].

- **Alpha-Miner**: The Alpha-Miner is one of the foundational process discovery algorithms in process mining. Being known for its simplicity, in the pm4py python package, this algorithm has no parameters that can be tuned with hyperparameter tuning.
- **Inductive Miner**: The Inductive Miner is another widely used process discovery algorithm that aims to capture both control flow and concurrency in process models. It offers several hyperparameters for customization, such as:
  - Noise Threshold: Determines the level of tolerance for considering activities as noise in a process model. Activities that occur less frequently than this threshold are treated

as noise and may not be included in the resulting process model. This parameter helps to filter out infrequent or outlier activities, focusing on the most relevant and recurring patterns in the process.

- **Heuristic Miner**: The Heuristic Miner is a heuristic-based process discovery algorithm that focuses on extracting process models from event data. It is known for its ability to handle noise and uncertainty in event logs. Some of the hyperparameters associated with the Heuristic Miner include:
  - Dependency Threshold: Sets a threshold for the level of dependency between activities in the model. Activities with lower dependencies may be considered parallel or independent.
  - AND Threshold: Sets the minimum occurrence frequency for a set of activities to be considered as an AND-split in the process model. If a set of activities occurs less frequently than this threshold, they are not treated as an AND-split and may be represented differently in the mined model.
  - Loop two threshold: Specifies the minimum number of times a loop with two activities must occur in the process data for it to be considered as a loop in the mined process model.

Understanding the hyperparameters of process mining algorithms is crucial for tailoring the analysis to the unique characteristics of your event data. By adjusting these hyperparameters, you can influence the resulting process models' complexity, accuracy, and level of detail.

2.5 Synthesis of the State of the Art

In this state of the art, we have explored the intersection of process mining and document management systems. Event logs can be used to conduct **three types of process mining** as shown in Figure 2.1 [34]:

- **Discovery** is the initial type of process mining. Without an a priori model, process discovery creates a process model from the event log.
- **Conformance checking** performs a comparison between a process model and a corresponding event log to discover similarities and differences between the two.
- **Process enhancement** takes the existing process model and attempts to extend or improve it. Given an event log and a process model as input, it returns an improved process model.

However, optimizing the performance of process mining algorithms remains a critical challenge. This is where the intersection with machine learning and hyperparameter tuning becomes

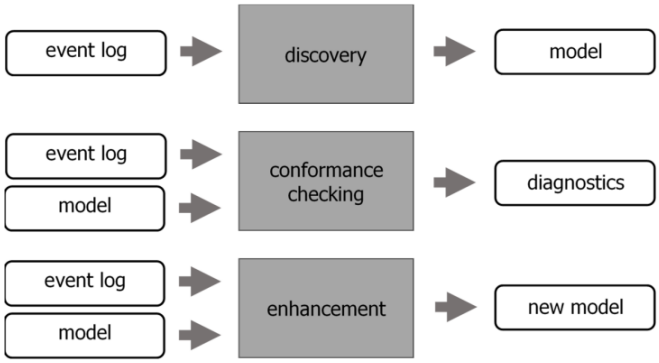


Figure 2.9: Process mining (based on [31])

crucial. Leveraging techniques from the field of machine learning, hyperparameter tuning offers a systematic approach to fine-tuning process mining algorithms, enhancing their efficiency.

Within this broader context, three hyperparameter tuning algorithms take center stage: **Genetic Algorithm**, **Random Search**, and **Bayesian Optimization**.

Genetic Algorithm, inspired by the principles of **natural selection**, employs a population-based approach to iteratively evolve hyperparameter configurations.

On the other hand, **Random Search**, an alternative to exhaustive grid search, offers an efficient approach by **randomly selecting hyperparameters** from predefined distributions. This method explores a broader range of hyperparameter values, often converging to promising solutions more swiftly.

Bayesian Optimization, harnesses **probabilistic models** to estimate the performance of different hyperparameter settings. Operating as an informed search algorithm, Bayesian Optimization dynamically selects hyperparameters based on predictions made by these probabilistic models.

By subjecting these three hyperparameter tuning algorithms to testing, this study aims to uncover the most effective and efficient approach to fine-tune process mining algorithms within the context of document management systems, which is still an under-explored field since there is no literature on the optimization of process mining using hyperparameter tuning methods.

Chapter 3

Methodology

3.1 Overview

In the context of this thesis, our **primary objective** is to leverage **hyperparameter tuning** techniques to systematically **optimize process models** derived from real-world event data. Process models play a pivotal role in understanding, analyzing, and enhancing complex workflows, making their accuracy and interpretability critical. The aim is to utilize hyperparameter tuning models to refine the performance of process discovery algorithms in terms of precision, simplicity, generalization, and replay fitness.

Our **secondary goal** involves a **comparative analysis**. We seek to evaluate the efficacy of these tuned models by rigorously benchmarking them against the baseline "raw" models generated using default process mining algorithms. This comparative assessment will provide valuable insights into the extent of improvement achieved through hyperparameter tuning. Ultimately, our research endeavors to contribute to the evolution of process mining practices, enhancing their efficiency and applicability in document-centric domains and beyond. The Figure 3.1 shows a diagram with a summary of the structure of the objectives.

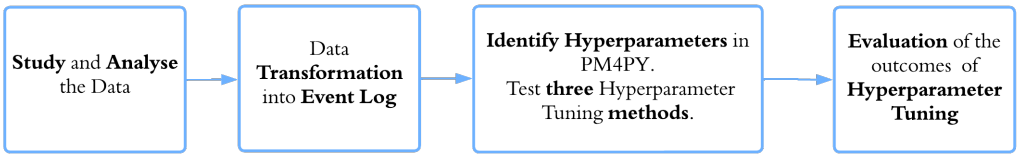


Figure 3.1: Summary of the structure of the objectives.

First, a collaborative effort was established with IPBrick to access the essential data required for this research. Initially, the data was accessed through DBeaver software for data analysis, facilitating the preliminary stages of the project.

As the project progressed, IPBrick opted to provide the data in a JSON format, with each file corresponding to a specific process. These event logs served as the primary input for the process mining discovery algorithms, forming the foundation for creating comprehensive process models.

Subsequent to an in-depth study of the dataset and the necessary data transformations, the focus shifted towards identifying the hyperparameters within the process model algorithms from the PM4Py Python package. The aim was to fine-tune these algorithms using hyperparameter tuning techniques to optimize their performance.

Throughout the research, Python 3.9.2 served as the primary programming environment, and various Python packages were employed to aid in data analysis and modeling. Notably, the Process Mining for Python (PM4Py) library played a pivotal role in facilitating data manipulation, data importation, and the utilization of diverse process mining algorithms. Its contributions were instrumental in the successful execution of this research.

3.2 Data Analysis

The initial step involved accessing data from IPBrick using SQL queries with the Psycopg2 package. The data came from the iPortalDoc platform. iPortalDoc serves as an all-encompassing solution for managing documents and processes, utilizing workflow functionality. What sets it apart is its seamless integration with a Unified Communications Center, enabling the recording of calls, emails, and chat conversations within the document management system. These recorded interactions can then be linked to relevant documents and processes, making them easily accessible for reference whenever needed [21].

Accessing the data directly was crucial in helping us gain a comprehensive understanding of the data's structure and content.

The database at hand encompassed a total of 1063 tables. IPBrick provided us an overview of the most important tables in the iPortalDoc, in Figure 3.2 it is presented a scheme with the tables that describe the workflow of a process, the 8 selected tables, enhanced in the figure, are "accaorev," "estadorev," "workflowrev," "revisaodoc," "documento," "mailutilizador," "ligadocsec," and "seccao."

The combination of essential data resulted in the creation of a dataframe comprising 19 columns, providing a comprehensive description of the process workflow with valuable information. The table 3.1 is the 10 first records of the dataframe.

	idaccaorev	idestadorev	idaccao	realizado	idpessoa	datainicio	datarealizacao	descricao	idmultiplos	docid	limitdate	idworkrev	idestado	idwork	idrev	finalizado	utilizador	mail	codigo
0	38762	32913	578	False	1	2017-01-17 15:41:08	None	Email Reply	NaN	17217	2017-02-16 15:41:08	17279	649	128	18184	False	10284	bcastanheira@ucoip.pt	EI_5/2017
1	19315	13422	578	True	1	2016-05-16 10:03:41	2016-09-27 15:21:50+01:00	Email Reply	2155.0	8062	2016-06-15 10:03:41	8106	649	128	8767	True	10000	administrator@ucoip.pt	SM_3805/2016
2	19265	13397	578	True	1	2016-05-16 10:02:28	2016-09-27 15:21:51+01:00	Email Reply	2155.0	8037	2016-06-15 10:02:28	8081	649	128	8742	True	10000	administrator@ucoip.pt	SM_3780/2016
3	31325	19874	578	True	1	2016-07-18 10:21:22	2016-09-27 10:22:26+01:00	Email Reply	2155.0	14047	2016-08-17 10:21:22	14091	649	128	14790	True	10000	administrator@ucoip.pt	SM_9386/2016
4	31549	19986	578	True	1	2016-07-18 10:40:44	2016-09-27 10:22:34+01:00	Email Reply	2155.0	14159	2016-08-17 10:40:44	14203	649	128	14902	True	10000	administrator@ucoip.pt	SM_9497/2016
5	31577	20000	578	True	1	2016-07-18 10:42:39	2016-09-27 10:22:46+01:00	Email Reply	2155.0	14173	2016-08-17 10:42:39	14217	649	128	14916	True	10000	administrator@ucoip.pt	SM_9511/2016
6	31461	19942	578	True	1	2016-07-18 10:32:56	2016-09-27 10:22:47+01:00	Email Reply	2155.0	14115	2016-08-17 10:32:56	14159	649	128	14858	True	10000	administrator@ucoip.pt	SM_9454/2016
7	38763	32913	-10	False	1	2017-01-17 15:41:08	2017-01-17 15:41:08+00:00	Permissao de leitura	NaN	17217	2017-02-16 15:41:08	17279	649	128	18184	False	10284	bcastanheira@ucoip.pt	EI_5/2017
8	31505	19964	578	True	1	2016-07-18 10:37:14	2016-09-27 10:22:35+01:00	Email Reply	2155.0	14137	2016-08-17 10:37:14	14181	649	128	14880	True	10000	administrator@ucoip.pt	SM_9475/2016
9	25303	16457	578	True	1	2016-06-06 10:34:03	2016-09-27 13:31:21+01:00	Email Reply	2155.0	11026	2016-07-06 10:34:03	11071	649	128	11736	True	10000	administrator@ucoip.pt	SM_6694/2016

Table 3.1: First 10 records of the Dataframe returned from the SQL query.

In Figure 3.2 it is presented the description of the database using the Pandas tool, in terms of column name, type, null count, and number of entries. It resulted in 222477 entries, in which 'datarealizacao' and 'idmultiplos' have null values, 35432 and 206352, respectively. We can conclude that it entailed with ten numeric (integers) columns, two booleans, two date types and four categorical.

Column	Non-Null Count	Dtype
idaccaorev	222477	int64
idestadorev	222477	int64
idaccao	222477	int64
realizado	222477	bool
idpessoa	222477	int64
datainicio	222477	datetime64[ns]
datarealizacao	187045	object
descricao	222477	object
idmultiplos	16125	float64
docid	222477	int64
limitdate	222477	datetime64[ns]
idworkrev	222477	int64
idestado	222477	int64
idwork	222477	int64
idrev	222477	int64
finalizado	222477	bool
utilizador	222477	int64
mail	222477	object
codigo	222477	object
Total	222476	

Table 3.2: DataFrame Structure

As previously discussed in Section 2.2.1, a minimum of three variables is essential for the application of process mining algorithms. However, in our specific context, we opted for the inclusion of four variables out of the available 19. These four variables were deemed crucial for the successful application of these algorithms and analysis of processes. The selected variables were: timestamp ('datainicio'), activity ('descricao'), case identifier ('idworkrev'), and process

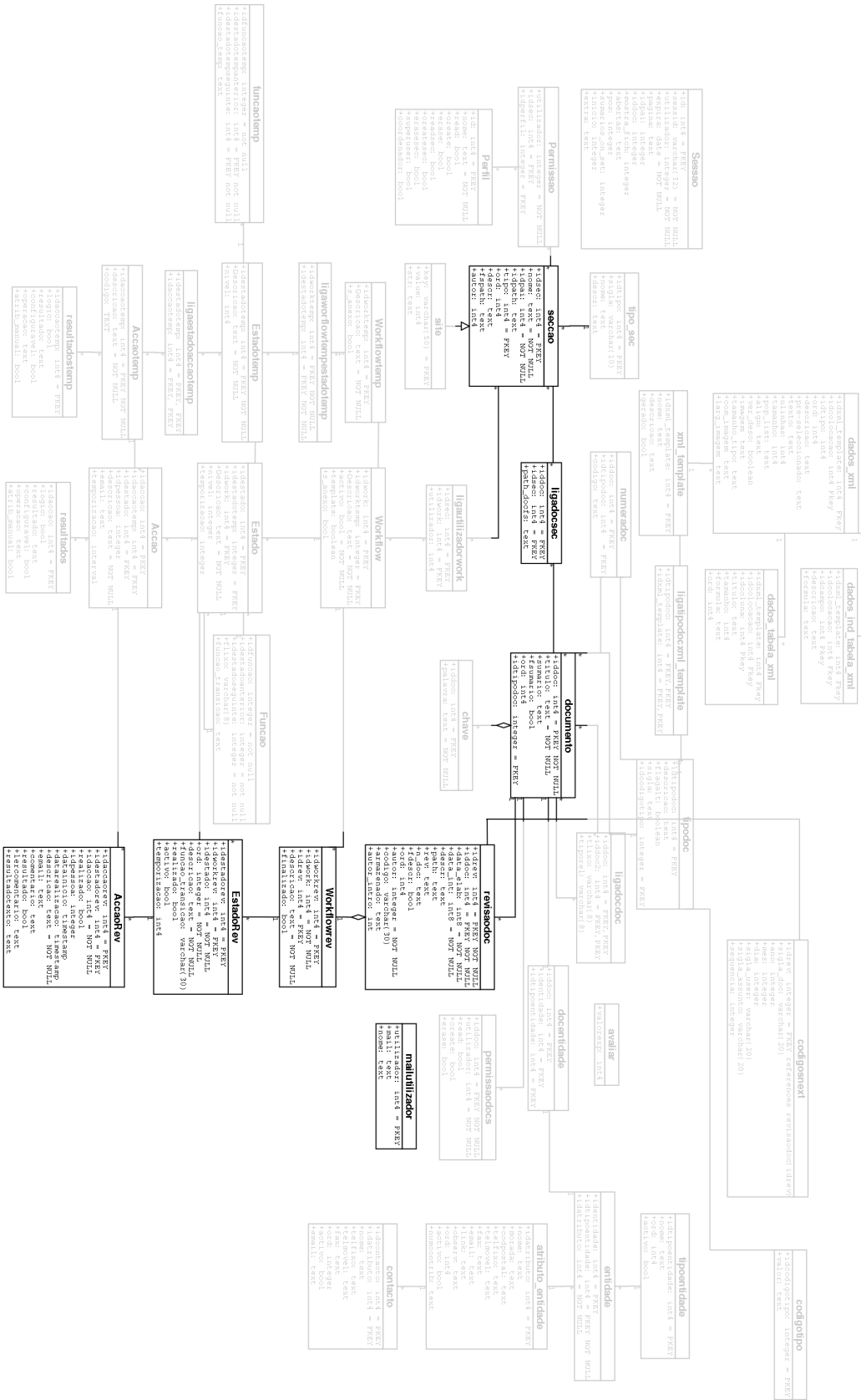


Figure 3.2: Database Structure

identifier ('workid'). In the Table 3.3 we can see the first 10 records of the trimmed dataframe ordered by date.

	datainicio	descricao	idworkrev	idwork
174224	2016-01-20 10:31:48	Email Reply	3669	128
198915	2016-01-20 15:34:09	Email Reply	3670	128
198918	2016-01-21 15:55:17	Encaminhar Circular Informativa para assinatura	3671	195
198919	2016-01-21 15:56:32	Permissao de leitura	3671	195
862	2016-01-21 15:57:06	Divulgar Circular Informativa	3671	195
174229	2016-01-21 15:57:47	Tomar conhecimento de Circular Informativa	3671	195
1216	2016-01-21 15:57:49	Permissao de leitura	3671	195
161860	2016-01-25 18:01:52	Email Reply	3674	128
174230	2016-01-25 18:29:31	Email Reply	3676	128
14259	2016-01-26 15:23:26	Email Reply	3681	128

Table 3.3: First 10 records of the Dataframe after trimming.

The remaining variables, while informative, provide supplementary details about the processes and their instances. The supplementary variables furnish valuable insights, including information about task performers and code, the potential involvement of multiple individuals in a task and task completion status, this extra information in the context of this thesis was an invaluable resource, since our main goal is hyperparameter tuning of the discovery algorithms.

In our pursuit of a more profound comprehension of the distribution of activities and instances within processes, we undertook the creation of two informative tables. The first table was generated to provide insights into the count of activities per process, while the second table offered a glimpse into the frequency of instances per process.

This approach involved the systematic compilation of unique process identifiers into separate lists for each table. This method allowed us to methodically organize and quantify the occurrences of activities and instances across a spectrum of processes, thereby facilitating a comprehensive analysis of their distribution patterns. These tables serve as valuable tools for extracting meaningful insights and patterns from the data, enhancing our ability to identify trends and anomalies within the document management processes under investigation. The data generated can be seen in the tables 3.4 and 3.5 (containing the first 10 records).

	Process	Number of Activities
0	128	11
1	250	15
2	268	17
3	194	2
4	165	20
5	175	20
6	234	26
7	198	10
8	177	10
9	264	30

Table 3.4: First 10 records of the number of activities per process.

	Process	Number of Instances
0	128	43234
1	250	205
2	268	156
3	194	6
4	165	34
5	175	93
6	234	270
7	198	18
8	177	30
9	264	263

Table 3.5: First 10 records of the number of instances per process.

Employing the Pandas package significantly enriched our analysis by providing valuable insights into the distribution of activities and processes. The tool used in this analysis was the describe() function, an integral feature of Pandas.

This function allowed us to quickly obtain essential statistics, such as mean, standard deviation, and quartile values, enabling a comprehensive assessment of data distribution patterns. In the resulted dataframe the values were rounded off to 3 decimal places. Table 3.6 represents the output of this tool.

	Number of Activities	**Number of Instances**
Count	69	69
Mean	11.971	677.29
Standard Deviation	7.683	5199.25
Minimum	1	1
25th Percentile (Q1)	6	6
50th Percentile (Median, Q2)	11	18
75th Percentile (Q3)	17	56
Maximum	30	43234

Table 3.6: Summary Statistics for Number of Activities and Number of Instances

The table presents a summary of key statistics related to the number of activities and instances observed within a dataset containing 69 distinct processes. After analyzing each parameter, we can conclude that:

- **Number of Activities:**
  - **Count:** The dataset comprises a total of 69 instances of processes, each featuring varying numbers of activities.
  - **Mean:** On average, each process includes approximately 11.971 activities.
  - **Standard Deviation:** The standard deviation of 7.683 indicates the degree of variability in the number of activities among the processes.

  - **Minimum:** The minimum number of activities in a single process is 1, signifying that some processes involve relatively few activities.
  - **25th Percentile (Q1):** At the 25th percentile, approximately 6 activities are observed in processes, suggesting that a quarter of the processes have fewer activities.
  - **50th Percentile (Median, Q2):** The median number of activities, representing the middle value when the data is sorted, is approximately 11 activities.
  - **75th Percentile (Q3):** At the 75th percentile, approximately 17 activities are present in processes, indicating that a quarter of the processes have more activities.
  - **Maximum:** The dataset's maximum number of activities in a single process is 30, showcasing the highest observed activity count among the processes.

- **Number of Instances:**
  - **Count:** Similar to the number of activities, the dataset contains 69 instances of processes, each characterized by a varying number of instances.
  - **Mean:** On average, each process encompasses approximately 677.29 instances, reflecting substantial variation across the dataset.
  - **Standard Deviation:** The standard deviation of 5199.25 underscores the wide-ranging distribution of instances among the processes.
  - **Minimum:** The minimum number of instances within a single process is 1, indicating that some processes involve very few instances.
  - **25th Percentile (Q1):** At the 25th percentile, processes typically encompass approximately 6 instances, representing the lower end of the instance count spectrum.
  - **50th Percentile (Median, Q2):** The median number of instances is approximately 18, highlighting the middle value when sorted.
  - **75th Percentile (Q3):** At the 75th percentile, processes tend to have around 56 instances, illustrating a moderate instance count for a quarter of the processes.
  - **Maximum:** The dataset's maximum number of instances within a single process is notably high at 43234, indicating significant variability and the presence of processes with an extensive number of instances.

After grasping the data's central tendencies and distribution, a bar plot was created, shown in the Figure 3.3 to have a summary and visual representation of the statistics that were analysed.

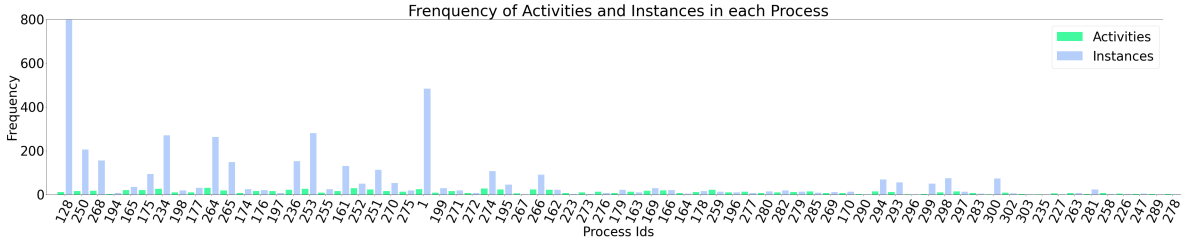


Figure 3.3: Bar Plot of the Frequency of Activities and Instances in each Process

In the Bar Plot, Figure 3.3, that was generated, we can conclude, that Process 128 is an outlier. It has a very high discrepancy in the number of instances compared to all of the other processes, entailing exactly 43234, as can be checked in the Table 3.5.

This comprehensive data analysis provides a foundational understanding of the dataset's characteristics, setting the stage for subsequent stage, being, the data conversion.

3.3 Data Manipulation

The transformation of raw data, provided by IPBrick, into an event log compatible with the PM4Py framework, was an important step in the exploration of our document-centric processes. This chapter unveils the details of this crucial data conversion process.

The data initially arrived in JSON format, containing a lot of information about document management processes. To unlock its potential, the first task was to normalize the JSON file into a structured dataframe, which was made possible by the Pandas package. In the table 3.7 its an example, including only the first 10 records, of a received JSON after the conversion into a dataframe.

	idworkrev	descricao	datarealizacao	idmultiplostemp	resultado	idpessoa
0	16376	Encaminhar Circular Informativa para assinatura	None	3464	P/ Assinar	10227
1	16376	Encaminhar Circular Informativa para assinatura	None	3463	Divulgar	10227
2	16387	Encaminhar Circular Informativa para assinatura	2016-09-13 11:45:58+01	3464	P/ Assinar	10227
3	16387	Encaminhar Circular Informativa para assinatura	2016-09-13 11:45:58+01	3463	Divulgar	10227
4	16387	Divulgar Circular Informativa	2016-09-13 11:47:15+01	4707	P/ correcao	10227
5	16387	Divulgar Circular Informativa	2016-09-13 11:47:15+01	4708	Divulgar	10227
6	16387	Tomar conhecimento de Circular Informativa	2016-09-13 11:47:42+01	3468	Tomei conhecimento	10227
7	16529	Encaminhar Circular Informativa para assinatura	2016-10-11 18:14:47+01	3464	P/ Assinar	10354
8	16529	Encaminhar Circular Informativa para assinatura	2016-10-11 18:14:47+01	3463	Divulgar	10354
9	18248	Divulgar Comunicado	None	3464	P/ Assinar	10219

Table 3.7: First 10 records of the normalized JSON.

With this data in hand, identifying the key elements that define a process was paramount. After analyzing the data contents, it became apparent that:

- **'idworkrev'**: provided a unique trail to trace each process instance, therefore, being the case identifier.
- **'descricao'** and **'resultado'**: describe the activities that the process entailed, serving as the activity.
- **'datarealizacao'**: marked the chronological evolution of processes over time, therefore, being the timestamp.
- **'idmultiplostemp'** and **'idpessoa'** : add additional information to the process, that in this case, didn't add much value regarding process mining discovery and the subsequent hyperparameter tuning.

After obtaining the three essential variables required for the creation of a PM4Py event object: the case ID ('idworkrev'), activity ('descricao' and 'resultado'), and timestamp ('datarealizacao'), we proceed to generate an XES file. EXtensible Event Stream, XES, conforms to the IEEE Standard, designed to facilitate interoperability in event logs and event streams. The standard defines XES as "a language to transport, store, and exchange event data" [11].

Subsequently, a PM4Py log is constructed, with the XES file serving as a crucial component in its creation. This log object enables us to construct various models, such as Petri nets, heuristic nets, directly-follow graphs, or process trees, utilizing PM4Py functions.

Figure 3.4 showcases the transformed JSON file, containing process information after its conversion into a dataframe and subsequent generation into an XES file (the Figure 3.4 doesn't show the full XES file, only a portion). Each trace corresponds to a specific project instance, with each activity identified as an individual event. Therefore, a trace constitutes a collection of events, and the log is composed of an assemblage of traces.

```
<?xml version="1.0" encoding="utf-8" ?>
<log xes.version="1849-2016" xes.features="nested-attributes" xmlns="http://www.xes-standard.org/">
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext" />
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext" />
  <string key="origin" value="csv" />
  <trace>
    <string key="concept:name" value="11920" />
    <event>
      <string key="idworkrev" value="11920" />
      <string key="descricao" value="Encaminhar Circular Informativa para assinatura _ P/ Assinar" />
      <date key="datarealizacao" value="2016-06-20T10:23:19+00:00" />
      <string key="concept:name" value="Encaminhar Circular Informativa para assinatura _ P/ Assinar" />
      <date key="time:timestamp" value="2016-06-20T10:23:19+00:00" />
      <int key="@@index" value="64" />
    </event>
    <event>
      <string key="idworkrev" value="11920" />
      <string key="descricao" value="Encaminhar Circular Informativa para assinatura _ Divulgar" />
      <date key="datarealizacao" value="2016-06-20T10:23:19+00:00" />
      <string key="concept:name" value="Encaminhar Circular Informativa para assinatura _ Divulgar" />
      <date key="time:timestamp" value="2016-06-20T10:23:19+00:00" />
      <int key="@@index" value="65" />
    </event>
    <event>
      <string key="idworkrev" value="11920" />
      <string key="descricao" value="Divulgar Circular Informativa _ P/ correção" />
      <date key="datarealizacao" value="2016-06-20T10:28:16+00:00" />
      <string key="concept:name" value="Divulgar Circular Informativa _ P/ correção" />
      <date key="time:timestamp" value="2016-06-20T10:28:16+00:00" />
      <int key="@@index" value="66" />
    </event>
```

Figure 3.4: Portion of XES file example.

This phase marks the transition from raw data into an event log compatible with the Pm4py framework. The event log, now a repository of process instances and activities, sets the stage for the subsequent phases of process discovery and hyperparameter tuning.

3.4 Process Mining Algorithms Selection

In this chapter, we discuss the selection of process mining algorithms for evaluation and the preferred modeling notation. After a comprehensive analysis of several process mining algorithms available in the Pm4py framework, the decision was made to focus on testing two key algorithms: Inductive Miner and Heuristic Miner.

This decision was driven by the fact that both algorithms offer hyperparameters that can be fine-tuned during the hyperparameter tuning phase of the thesis. In Table 3.8 is presented a brief description of the selected algorithms from the Pm4py framework [15] [16]:

Algorithm	**Description**
Inductive Miner	• Detects various 'cuts' in the log (e.g., sequential cut, parallel cut, concurrent cut, and loop cut). • Recurs on sublogs identified by applying the cut until a base case is found. • Utilizes hidden transitions extensively, especially for handling skipping/looping within the model. • Assigns unique labels to each visible transition in the model.
Heuristic Miner	• Operates on the Directly-Follows Graph to handle noise and identify common constructs such as dependencies between activities and AND-logic. • Outputs an Heuristics Net, containing activities and their relationships. • The Heuristics Net can be converted into a Petri net.

Table 3.8: Description of the selected Process Mining Algorithms

To make the most of these algorithms and tailor them to our specific needs, it's essential to understand the parameters available for configuration. In Table 3.9 we provide an overview of the parameters that can be fine-tuned when using PM4Py's Inductive Miner and Heuristic Miner algorithms. These parameters enable us to customize the mining process and optimize the results according to our research objectives.

These selected algorithms offer promising avenues for experimentation and evaluation within

Parameter	Inductive Miner	Heuristic Miner
log	event log / Pandas dataframe / typed DFG	event log / Pandas dataframe
noise_threshold (float)	noise threshold (default: 0.0)	-
dependency_threshold (float)	-	dependency threshold (default: 0.5)
and_threshold (float)	-	AND threshold (default: 0.65)
loop_two_threshold (float)	-	loop two threshold (default: 0.5)
Return type	Tuple[PetriNet, Marking, Marking]	Tuple[PetriNet, Marking, Marking]

Table 3.9: Parameters for PM4Py's Inductive Miner and Heuristic Miner Algorithms

hyperparameter tuning, providing opportunities for optimization and fine-tuning. In the subsequent chapters, we delve into the details of applying these algorithms to process mining tasks and their performance.

3.4.1 Modeling Notation Selection

In this work, Petri nets emerge as a central focal point for our exploration into process mining. As one of the most widely used notations in the field, Petri nets offer a powerful framework for modeling, visualizing, and analyzing complex processes. We leverage this versatile notation to construct our process models using the JSON files containing information about several processes provided by IPBrick. This data originates from the IPBrick IPortalDoc, a pivotal resource in their operational framework. Figure 3.5 illustrates the structure and nature of the models that this platform provides.

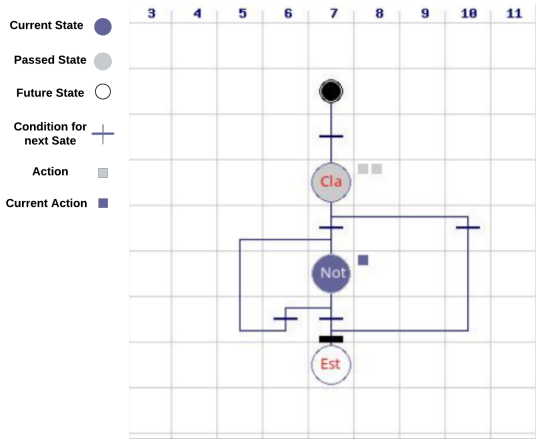


Figure 3.5: IPortal Doc Model Example.

In the visualization and analysis of Petri nets, we make extensive use of the Pm4py and Graphviz packages.

3.5 Hyperparameter Tuning Algorithm Selection

In this work, we explore the selection and application of three distinct hyperparameter tuning algorithms: Random Search, Bayesian Optimization, and Genetic Algorithms. Each of these techniques offers unique characteristics and advantages.

In Chapter 2.3 of the State of the Art is a deeper description and exploration about how these algorithms work, however, in the next subsections 3.5.1, 3.5.2 and 3.5.3 we will provide a brief overview of the key characteristics and the Python packages needed to make use of these algorithms.

3.5.1 Random Search

Random Search is a simple yet effective hyperparameter tuning method. It randomly samples hyperparameters from predefined search spaces.

Key Characteristics:

- **Exploration**: Random Search explores the hyperparameter space through random sampling.
- **Ease of Use**: It is straightforward to implement and computationally less expensive.
- **No Prior Information**: Random Search does not require prior knowledge about the problem.

Package Used: In the implementation of Random Search, the built-in Python "random" package was utilized.

3.5.2 Bayesian Optimization

Bayesian Optimization employs probabilistic models to guide the search for optimal hyperparameters. It builds a surrogate model to predict promising hyperparameter configurations.

Key Characteristics:

- **Exploitation**: Bayesian Optimization balances exploration and exploitation using a probabilistic model.
- **Efficiency**: It converges to optimal hyperparameters with fewer iterations compared to Random Search.
- **Prior Information**: Bayesian Optimization can incorporate prior knowledge about the problem.

Package Used: Bayesian Optimization was implemented using the "gp_minimize" function from the "scikit-optimize (skopt)" library.

3.5.3 Genetic Algorithm

Genetic Algorithms are population-based optimization techniques inspired by natural selection. They maintain a population of potential solutions and evolve them over multiple generations.

Key Characteristics:

- **Population-based**: Genetic Algorithms work with a population of individuals, evolving them over time.
- **Crossover and Mutation**: They use genetic operations like crossover and mutation to create new individuals.
- **Multi-Objective**: Genetic Algorithms can handle optimization tasks with multiple objectives.

Package Used: The implementation of Genetic Algorithms utilized the DEAP (Distributed Evolutionary Algorithms in Python) library, which provides tools for evolutionary algorithms, including custom operator and fitness function definitions.

In the following Chapter, we will delve into the specific details of our experiments, highlighting the difference in the results for choosing these algorithms and their performance in optimizing hyperparameters of the process mining algorithms in case.

Chapter 4

Results and Discussion

4.1 Overview

In this chapter, we evaluate and scrutinize the performance of three hyperparameter tuning algorithms: Random Search, Bayesian Optimization, and Genetic Algorithm. These algorithms play a pivotal role in the quest for enhancing the efficiency and effectiveness of process mining, a domain with diverse applications in the analysis and optimization of business processes.

Our evaluation focuses on two fundamental process mining techniques, the Inductive Miner and the Heuristic Miner. These techniques are essential tools for extracting valuable insights from event logs.

The primary objective of this chapter is twofold: first, to rigorously compare the results obtained through the application of each hyperparameter tuning algorithm, and second, to discern which process mining algorithm exhibits the most consistent and robust performance.

Our assessment is guided by a set of meticulously chosen metrics, which serve as the criteria for evaluating the effectiveness of the hyperparameter models. These metrics encompass the four crucial quality dimensions: precision, fitness, generalization, and simplicity. They enable us to comprehensively analyze the efficacy of each configuration and, ultimately, make informed decisions regarding their suitability for real-world process mining applications.

The outcomes of this exhaustive evaluation will shed light on critical questions within the realm of hyperparameter tuning and process mining.

4.2 Experimentations

This section begins by examining two distinct processes and presenting their model outputs along with key statistics. We then apply hyperparameter tuning algorithms, exploring both visual and statistical differences they introduce to the models. This combined qualitative and quantitative

approach aims to provide a comprehensive understanding of hyperparameter tuning's impact on process mining.

4.2.1 First Experiment - Case 'Faturas Fornecedor'

The process under examination is known as 'Faturas Fornecedor' which involves tasks such as classification, validation, payment processing, and adding or reading specific comments. Before applying the process mining algorithms, it was crucial to transform the process data from a JSON format into a suitable event log. To facilitate this conversion, we initially structured the data into a DataFrame, as illustrated in Table 4.1.

	idworkrev	descricao	datarealizacao	idmultiplostemp	resultado	idpessoa
0	19486	Classificar Fatura de Fornecedor	2017-10-16 16:17:25+01	4316	Cancelar	10167
1	19486	Classificar Fatura de Fornecedor	2017-10-16 16:17:25+01	4315	P/ validacao	10167
2	19486	Validar Fatura	2017-10-16 16:17:33+01	4385	Validar	10167
3	19486	Validar Fatura	2017-10-16 16:17:33+01	4309	Devolver a Fornecedor	10167
4	19486	Introduzir documento	2017-10-16 16:18:48+01	None	Introduzido	10167
5	19486	Devolver a Fornecedor	2017-10-16 16:19:42+01	4314	Devolver	10167
6	18181	Classificar Fatura de Fornecedor	2017-04-27 10:51:40+01	4316	Cancelar	10167
7	18181	Classificar Fatura de Fornecedor	2017-04-27 10:51:40+01	4315	P/ validacao	10167
8	18181	Validar Fatura	2017-04-27 10:53:45+01	4385	Validar	10167
9	18181	Validar Fatura	2017-04-27 10:53:45+01	4309	Devolver a Fornecedor	10167

Table 4.1: Faturas Fornecedor JSON after being converted into DataFrame

4.2.1.1 Heuristic Miner

As discussed in Section 3.3, the process of transforming the DataFrame into an event log has been detailed. Our exploration begins with the **Heuristic Miner**, employing the 'pm4py.discover_petri_net_heuristics' package. For this initial examination, we opted to utilize default hyperparameter values, including 'dependency_thresh', 'and_threshold', and 'loop_two_threshold'. The resulting model is visually represented in Figure A.1.

In the Table 4.2, its presented the output of the statistics provided by each evaluation metric, utilizing the 'pm4py.algo.evaluation' package.

Metric	Value
Fitness	0.922
Precision	0.993
Simplicity	0.517
Generalization	0.470

Table 4.2: Summary of Evaluation Metrics for Heuristic Miner

When analyzing both the visualization and results of the four quality dimensions, the fitness score is high at **0.922**, indicating that the heuristic miner produced model fits quite well with the event data in terms of capturing the observed behavior. The precision score is also high at **0.993**, suggesting that the model predicted the actual outcomes based on the event log.

The simplicity score is comparatively lower at **0.517**, indicating that the generated Petri net has a higher level of complexity in terms of its structure and transitions. The generalization score is moderate at **0.470**, indicating a reasonable level of generalization capability in capturing underlying patterns in the data.

In the visualization of the Petri net, Figure A.1, it's evident that the complexity of the model corresponds to the lower simplicity value. The visually complex Petri net may have numerous transitions and places, making it challenging to interpret and analyze. While the model performs well in terms of fitness and precision, its complexity should be considered when assessing its practicality for process mining and analysis.

4.2.1.2 Inductive Miner

Introducing now the results obtained using the **Inductive Miner** algorithm. To explore the quality dimensions of the process model discovered through this method, we employed the 'pm4py.discover_petri_net_inductive' package. For this initial exploration, we opted for the default value of the hyperparameter 'noise_threshold.' In Table 4.3 below, we present the evaluation metrics that resulted from the process model discovered through the Inductive Miner algorithm.

Metric	Value
Fitness	0.999
Precision	0.226
Simplicity	0.617
Generalization	0.700

Table 4.3: Summary of Evaluation Metrics for Inductive Miner.

In terms of fitness, the Inductive Miner achieved a near-perfect score of **0.999**, indicating a high degree of alignment between the discovered process model and the actual process behavior. However, the precision score is notably lower at **0.226**, suggesting that the model may include some false positives.

Simplicity and generalization metrics for the Inductive Miner yielded values of **0.617** and **0.700**, respectively. These values indicate a more straightforward model structure and better generalization capabilities compared to the Heuristic Miner results.

In comparing these results with the earlier Heuristic Miner results, we observe that the Inductive Miner generally outperforms the Heuristic Miner in terms of fitness, generalization,

and simplicity. However, it lags behind in precision.

This suggests that, while the Inductive Miner may perform well in fitness and gather better results in model simplicity and generalization, it may require some fine-tuning to improve precision. In the Figure A.2, is presented the resulted model.

In the next subsections, we will test the three hyperparameter algorithms, Random Search, Bayesian Optimization, and Genetic Algorithm, and see the impact it has on the obtained models.

4.2.1.3 Random Search

In this section, **Random Search** is employed to optimize the hyperparameters of the **Heuristic Miner** algorithm and **Inductive Miner**. These hyperparameters include **'dependency_thresh'**, **'and_threshold'**, and **'loop_two_threshold'** for Heuristic Miner and **'noise_threshold'** for Inductive Miner, which influence the behavior of the process discovery algorithms. The objective is to identify the combination of hyperparameters that maximizes the algorithm's performance.

The process begins by randomly sampling values within the [0.0, 1.0] range for each of these hyperparameters over 20 iterations. This random selection allows for comprehensive exploration of the hyperparameter space.

The Heuristic Miner or Inductive Miner algorithm is then applied to the event log using the current set of hyperparameters. This application results in the discovery of a Petri net model that represents the underlying process.

Several evaluation metrics are considered to evaluate the discovered model's quality, including **fitness**, **precision**, **simplicity**, and **generalization**. These metrics provide insights into how well the process model aligns with the observed data.

The average evaluation score, which combines these metrics, is calculated for each set of hyperparameters. This score provides a measure of the overall performance of the Heuristic Miner under different configurations.

Throughout 20 iterations, the **Random Search** algorithm systematically explores various combinations of **'dependency_thresh'**, **'and_threshold'**, and **'loop_two_threshold'** for Heuristic Miner and **'noise_threshold'** for Inductive Miner. The algorithm keeps track of the best-performing hyperparameters and their associated average evaluation scores.

If, during any iteration, a set of hyperparameters yields a better average evaluation score than the previously recorded best score, the best hyperparameters and their corresponding score are updated.

Heuristic Miner

In the Table 4.4, is it presented the result of running **Random Search** for the Heuristic Miner, and the before and after scores when applying the hyperparameters that were chosen by the algorithm. The hyperparameters values that were provided resulted in **'dependency_thresh'** with **0.926**, **'and_threshold'** with **0.307**, and **'loop_two_threshold'** with **0.832**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**0.922**	0.993	0.517	0.470
After Hyperparameter Tuning (Random Search)	0.901	**0.995**	**0.636**	**0.734**

Table 4.4: Effect of Random Search Hyperparameter Tuning on Evaluation Metrics - Heuristic Miner

We can conclude that after hyperparameter tuning using Random Search, significant improvements were observed. **Fitness** remained strong at **0.901**, demonstrating continued alignment with the data, although it decreased slightly. This behavior is to be expected in optimization scenarios, since when optimizing other metrics fitness may lead to trade-offs with fitness. **Precision** improved slightly to **0.995**, indicating even more accurate predictions.

Notably, **simplicity** increased to **0.636**, reflecting a more straightforward process model. **Generalization** saw a substantial boost to **0.734**, suggesting enhanced performance on unseen data. This major enhancement in simplicity and generalization where also captured in the visualization of the Petri Net, Figure 4.1, that now presents a more interpretable model.

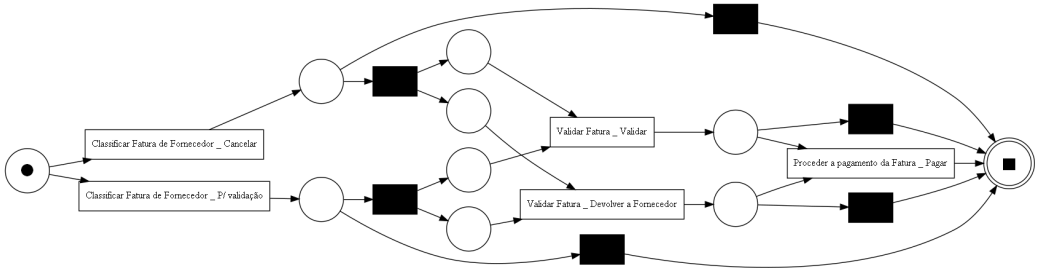


Figure 4.1: Heuristic Miner model from 'Faturas Fornecedor' after Random Search Hyperparameters Results.

Overall, hyperparameter tuning using Random Search led to a more balanced and effective process discovery model. The improvements in simplicity and generalization are particularly promising, as they indicate that the tuned model not only fits the training data well but also generalizes better to new scenarios.

Inductive Miner

This technique was also done using the same approach to the **Inductive Miner algorithm**.

Running the Random Search approach, the **'noise_threshold**' got a result of **0.160**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**0.999**	0.226	0.617	0.700
After Hyperparameter Tuning (Random Search)	0.964	**0.511**	**0.643**	**0.707**

Table 4.5: Effect of Random Search Hyperparameter Tuning on Evaluation Metrics - Inductive Miner

After hyperparameter tuning, there were noticeable improvements in several metrics. The **fitness** score remained high, indicating a good fit with the data. The **precision** score increased significantly, indicating a reduction in false positives and improved accuracy. The **simplicity** score also increased slightly, indicating a simpler and more interpretable Petri net structure. The **generalization** score remained at a high level, indicating a continued ability to generalize from the data.

In the Figure A.3 is a representation of the Petri Net after the new hyperparameter, where it is also apparent the improvement of the simplicity of the model compared to the default Petri Net.

4.2.1.4 Bayesian Optimization

In the **Bayesian Optimization** approach, being the goal to find the optimal set of hyperparameters for the **Heuristic** and **Inductive Miner** algorithms, the **objective function** to be minimized is defined, which takes as input a set of hyperparameters.

The **objective function** returns a value that quantifies the quality of the configuration of hyperparameters. In optimization problems, whether for hyperparameter tuning or other tasks, the goal is often to minimize a cost or **objective function**. By convention, optimization algorithms are designed to find the **minimum** of the function. If our scenario, the evaluation metric represents a "**higher is better**" context, in this case being the average of the four quality dimensions, this can achieved by **negating** the metric within the objective function.

The **Bayesian Optimization** algorithm, implemented using the **gp_minimize** function from the **scikit-optimize** library, iteratively explores the hyperparameter space defined by the ranges for each hyperparameter, within the **[0.0, 1.0]** range. The **number of optimization iterations** was **20**.

Heuristic Miner

The **Table 4.6** presents the result of running **Bayesian Optimization** for the **Heuristic Miner**, and the before and after scores when applying the hyperparameters that were chosen by the algorithm. The hyperparameter values that were provided resulted in '**dependency_thresh**'

with **0.948**, '**and_threshold**' with **0.216**, and '**loop_two_threshold**' with **1.0**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**0.922**	0.993	0.517	0.470
After Hyperparameter Tuning (Bayesian Optimization)	0.900	**0.995**	**0.636**	**0.734**

Table 4.6: Effect of Bayesian Optimization Hyperparameter Tuning on Evaluation Metrics - Heuristic Miner

The results reveal **substantial improvements** following the **Bayesian Optimization** approach. After fine-tuning the hyperparameters, we observe a slight decrease in the **fitness** and an increase in **precision** metrics. Notably, **simplicity** sees a **substantial boost**, increasing from **0.517** to **0.636**, indicating a more streamlined process model. Additionally, **generalization** improves notably, with the metric climbing from **0.470** to **0.734**, suggesting a better balance between overfitting and underfitting. These enhancements are also visible in the resulting Petri Net in **Figure 4.2**.

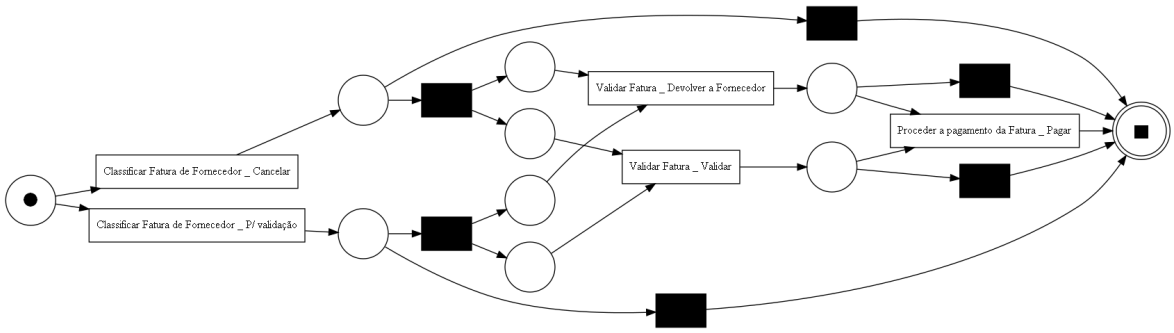


Figure 4.2: Heuristic Miner model from 'Faturas Fornecedor' after Bayesian Optimization Hyperparameters Results.

Inductive Miner

This technique was also performed using the **Inductive Miner** algorithm. Running the **Bayesian Optimization** approach, the '**noise_threshold**' got a result of **0.079**, and the **before** and **after scores** when applying the hyperparameters selected are shown in **Table 4.7**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**0.999**	0.226	0.617	0.700
After Hyperparameter Tuning (Bayesian Optimization)	0.992	**0.540**	**0.644**	**0.737**

Table 4.7: Effect of Bayesian Optimization Hyperparameter Tuning on Evaluation Metrics - Inductive Miner

After applying **Bayesian Optimization** to tune the hyperparameters, the **fitness** score maintained its high accuracy. The **precision** score notably increases to **0.540**, indicating a

reduction in false positives and enhanced precision. **Simplicity** sees a slight improvement, with the score increasing to **0.644**, suggesting a somewhat simpler model. **Generalization** also shows improvement, with a score of **0.737**, indicating enhanced generalization capabilities.

The improvements in **precision** and **generalization** are particularly notable, indicating the potential for enhanced process models and more reliable results. After the fine-tuning process it resulted in Petri Net in the Figure A.4.

These observations highlight the effectiveness of **Bayesian Optimization** hyperparameter tuning in enhancing the quality dimensions of the Inductive Miner algorithm. The resulting process model is not only more precise but also more interpretable.

4.2.1.5 Genetic Algorithm

In this section, we describe the implementation of a Genetic Algorithm for multi-objective optimization. This algorithm aims to find the optimal value of the hyperparameters to minimize an objective function.

The Genetic Algorithm shares similar objective function that was previously described Bayesian Optimization method, and we will highlight the commonalities.

1. **Fitness Function Setup:**

   A minimization fitness object (`FitnessMin`) is created for multi-objective optimization. The negative weights are applied to indicate that the algorithm aims to minimize objectives.

2. **Individual Representation:**

   An individual class (`Individual`) is created with a fitness attribute, allowing for multi-objective optimization.

3. **Toolbox Configuration:**

   A toolbox is set up to hold various functions and settings. The hyperparameter (dependency threshold) to be optimized is registered as a uniform random value between 0 and 1.

4. **Initialization:**

   The individual is defined as a list containing the hyperparameter value, and the population is defined as a list of individuals. The evaluation function (objective function) is registered within the toolbox.

5. **Genetic Operators Registration:**

   Genetic operators, including crossover and mutation, are registered using specific methods such as `cxBlend` and `mutGaussian`. These operators are essential for creating genetic diversity within the population [10].

6. **Population Initialization:**

   A population of a specified size, in our case we chose 15, is generated, with each individual having a random hyperparameter value within the defined range.

7. **Evolution Loop:**

   The Genetic Algorithm iterates over a specified number of generations, which in our case was 20. In each generation:

   - The algorithm identifies and stores the best-performing individual (hyperparameter value) in that generation, considering the average evaluation of the population.
   - Fitness values for each individual in the population are calculated based on the objective function.
   - The best hyperparameter and its corresponding evaluation are updated if a better individual is found.
   - NSGA-II selection is employed to choose the next generation, considering the multi-objective nature of the optimization problem.
   - Crossover and mutation are applied to create offspring, introducing genetic variation.
   - The old population is replaced with the offspring.

8. **Results Tracking:**

   Throughout the algorithm's execution, the best hyperparameter values and their corresponding evaluations are stored, allowing for the tracking of the algorithm's progress.

Heuristic Miner

The Heuristic Miner algorithm was subsequently tested running the **Genetic Algorithm**. The hyperparameter values that were provided resulted in '**dependency_thresh**' with **0.950**, '**and_threshold**' with **0.259**, and '**loop_two_threshold**' with **0.780**. Table 4.8 presents the results of the evaluation metrics before and after applying the **Genetic Algorithm**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**0.922**	0.993	0.517	0.470
After Hyperparameter Tuning (Genetic Algorithm)	0.900	**0.994**	**0.636**	**0.733**

Table 4.8: Effect of Genetic Algorithm Hyperparameter Tuning on Evaluation Metrics - Heuristic Miner

The **fitness** metric exhibited a slight decrease from **0.922** to **0.900** post-hyperparameter tuning. While this might seem counterintuitive, it's important to note that optimization algorithms often aim to strike a balance between various quality dimensions, in this case, the

average of the four metrics. The **precision**, remained practically the same maintaining its high value.

A significant boost in the **simplicity** metric, from **0.517** to **0.636**, indicates that the Genetic Algorithm guided the creation of streamlined and less complex process models. Improved simplicity is desirable for enhanced process understanding and management. **Generalization**, a crucial aspect of model quality, climbed notably from **0.470** to **0.733**. This demonstrates that the tuned hyperparameters achieved a more balanced model, striking a good equilibrium between overfitting and underfitting.

In the Figure 4.3 it is presented the resulted Petri Net and the apparent enhancement in comprehensibility.

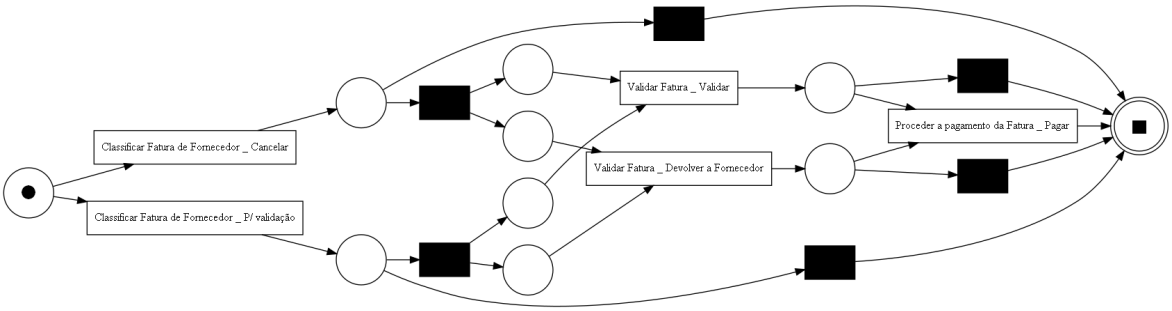


Figure 4.3: Heuristic Miner model from 'Faturas Fornecedor' after Genetic Algorithm Hyperparameters Results.

Inductive Miner

The Inductive Miner algorithm was subsequently tested running the **Genetic Algorithm**. The best hyperparameter value that was provided resulted in '**noise_threshold**' with **0.082**. Table 4.9 presents the results of the evaluation metrics before and after applying the **Genetic Algorithm**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**0.999**	0.226	0.617	0.700
After Hyperparameter Tuning (Genetic Algorithm)	0.993	**0.540**	**0.644**	**0.737**

Table 4.9: Effect of Genetic Algorithm Hyperparameter Tuning on Evaluation Metrics - Inductive Miner

Notably, **fitness** shows a slight decrease in value, this behavior has already been seen in other methods since it is common for some metrics to decrease when trying to optimize others. However, **precision** demonstrated a noteworthy increase from **0.226** to **0.540**, indicating that the tuned hyperparameters helped reduce false positives, leading to more precise process models.

Simplicity experienced a slight enhancement, rising from **0.617** to **0.644**. Furthermore,

generalization showed progress, increasing from **0.700** to **0.737** and maintaining its high generalization. In Figure A.5 we present the output of the visualization of the Petri Net when applying the chosen hyperparameteter.

4.2.2 Second Experiment - Case 'Correspondência'

In the second and final demonstration, we delve into another important process from iPortaldoc, known as **'Correspondência'**. This process encompasses a range of tasks, including **classification**, **forwarding** correspondences, taking **notices**, and **archiving** or **forwarding** changes. With its multifaceted nature, the 'Correspondência' process serves as a good candidate for evaluating the efficacy of hyperparameter tuning methods. In the Table 4.10 it is the first 10 records after converting the JSON file into a DataFrame.

	idworkrev	descricao	datarealizacao
0	16350	Classificar e encaminhar correspondencia receb...	2016-08-25 18:44:42+01
1	16350	Classificar e encaminhar correspondencia receb...	2016-08-25 18:44:42+01
2	16350	Tomar conhecimento de correspondencia recebida...	2016-08-26 11:39:38+01
3	16350	Tomar conhecimento de correspondencia recebida...	2016-08-26 11:39:38+01
4	16474	Encaminhar correspondencia p/ destinatario _ A...	2016-09-27 15:40:22+01
5	16474	Encaminhar correspondencia p/ destinatario _ E...	2016-09-27 15:40:22+01
6	16474	Tomar conhecimento de correspondencia recebida...	2016-09-27 15:46:53+01
7	16474	Tomar conhecimento de correspondencia recebida...	2016-09-27 15:46:53+01
8	16512	Encaminhar correspondencia p/ destinatario _ A...	2016-10-10 11:40:39+01
9	16512	Encaminhar correspondencia p/ destinatario _ E...	2016-10-10 11:40:39+01

Table 4.10: Correpondencia JSON after being converted into DataFrame

In the following sections, we will present the outcomes of applying the techniques explained in the previous section. In the next sections we will explore the results achieved for each hyperparameter tuning method, using the **Heuristic Miner** and **Inductive Miner** algorithms.

4.2.2.1 Heuristic Miner

Before jumping into the hyperparameter tuning algorithms results, first, we will show the metric values and visualization of the Petri Nets with the default values. First starting with the **Heuristic Miner**, the four dimension values are in the table 4.11.

The **fitness** metric, with a value of **0.803**, suggests a high level of conformity between the mined model and the observed data. A **precision** score of **0.927** indicates a strong ability to correctly identify relevant patterns. On the other hand, **simplicity** and **generalization** scores of **0.520** and **0.457**, respectively, suggest that there is room for further optimization in these areas. In the Figure B.2, represents the resulting Petri Net, where we can see the impact that low simplicity and generalization scores have.

Metric	Value
Fitness	0.803
Precision	0.927
Simplicity	0.520
Generalization	0.457

Table 4.11: Summary of Evaluation Metrics for Heuristic Miner.

4.2.2.2 Inductive Miner

Now for the **Inductive Miner** algorithm, the Table 4.12 was generated containing the summary of evaluation metrics for the **Inductive Miner**, showcasing its performance. Figure B.1, it is shown the resulting Petri Net.

Metric	Value
Fitness	1.0
Precision	0.279
Simplicity	0.64
Generalization	0.691

Table 4.12: Summary of Evaluation Metrics for Inductive Miner.

The **fitness** metric, with a perfect score of **1.0**, indicates an exceptionally high degree of alignment between the model generated by the Inductive Miner and the actual data. This suggests that the miner is exceptionally adept at capturing and representing the inherent patterns within the data. Moving on to **precision**, which stands at approximately **0.279**, we observe that the miner demonstrates low precision in identifying relevant patterns within the data.

Simplicity is another crucial dimension, and it registers at **0.64**. This suggests that the Inductive Miner tends to produce moderately straightforward models, although simplicity has room for improvement. Lastly, **generalization** exhibits a score of approximately **0.691**. This metric implies that the miner has a reasonably good capacity to generalize its findings to new, unseen data, indicating its potential utility in predictive modeling tasks.

4.2.2.3 Random Search

In this section, we will delve into the application of the **Random Search** hyperparameter tuning technique to both the **Heuristic Miner** and the **Inductive Miner** algorithms. This approach aims to optimize the performance of these data mining methods by systematically exploring a range of hyperparameter configurations. Similar to the initial demonstration, we will conduct **20** iterations of the Random Search process for each miner, assessing how these tunings impact key evaluation metrics.

Heuristic Miner

The Table 4.13 presents a comparative analysis of evaluation metrics for the Heuristic Miner **before** and **after** undergoing the **Random Search** hyperparameter tuning process.

After running the algorithm, the hyperparameters chosen were, **'dependency_thresh'** with **0.735**, **'and_threshold'** with **0.984** and, **'loop_two_threshold'** with **0.241**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	0.803	0.927	0.520	0.457
After Hyperparameter Tuning (Random Search)	**0.838**	**0.953**	**0.587**	**0.722**

Table 4.13: Effect of Random Search Hyperparameter Tuning on Evaluation Metrics - Heuristic Miner

After the Random Search process, notable enhancements can be observed across several key metrics. The **fitness** score increases to **0.838**, indicating a better alignment between the miner's model and the observed data. **Precision** slightly improves to **0.953**. **Simplicity** improves to **0.587**, suggesting that the tuned Heuristic Miner produces models that are relatively simpler. Lastly, **generalization** also sees a notable improvement, with a score of **0.722**, indicating the miner's enhanced ability to apply learned patterns to new, unseen data.

The Figure B.3 we can see a more visually interpretable Petri Net, mostly due to **simplicity** and **generalization** enhancements.

Inductive Miner

Table 4.14 provides a comprehensive insight into the impact of **Random Search** hyperparameter tuning on the performance of the **Inductive Miner** algorithm. The initial set of metrics, labeled as "Before Hyperparameter Tuning" showcases the algorithm's baseline performance with default hyperparameters, we use the same table structure demonstrated in previous sections.

After running the algorithm, the hyperparameters chosen were, **'noise_threshold'** with **0.466**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**1.0**	0.279	0.64	**0.691**
After Hyperparameter Tuning (Random Search)	0.944	**0.637**	**0.643**	0.663

Table 4.14: Effect of Random Search Hyperparameter Tuning on Evaluation Metrics - Inductive Miner

Following the application of the **Random Search** hyperparameter tuning process, the table highlights the changes in these critical evaluation metrics. Notably, while the **fitness** score

slightly decreases to **0.944**, it **remains** at a **high** level, signifying that the tuned Inductive Miner continues to effectively capture data patterns. What stands out is the **significant improvement in precision**, which jumps to **0.637**, indicating a substantial enhancement in the algorithm's ability to accurately identify relevant patterns.

Simplicity experiences a minor increase to **0.643**, suggesting that the algorithm's models maintain a relatively simple structure post-tuning. Furthermore, the **generalization** score sees a slight **decrease** to **0.663**, indicating that the algorithm's improved precision may come at a marginal cost to its generalization ability.

The Figure B.4 we can see a more visually interpretable Petri Net.

4.2.2.4 Bayesian Optimization

In this section, we dive into the exploration of **Bayesian Optimization** for 'Correspondencia' JSON, this powerful hyperparameter tuning technique, will be applied to both the **Heuristic Miner** and the **Inductive Miner** algorithms. As with the previous demonstrations, we will conduct **20 iterations** of Bayesian Optimization for each miner, systematically optimizing hyperparameters to enhance their performance. We will assess how this method impacts critical evaluation metrics and analyze the results in enhancing the performance of the Heuristic and Inductive Miner algorithms.

Heuristic Miner

Table 4.15 offers a comparative analysis of evaluation metrics for the Heuristic Miner both before and after undergoing the Bayesian Optimization hyperparameter tuning process. It's important to emphasize that the "Before Hyperparameter Tuning" section represents the algorithm's initial performance state with default hyperparameters.

After running the algorithm, the hyperparameters chosen were, **'dependency_thresh'** with **0.976**, **'and_threshold'** with **0.801** and, **'loop_two_threshold'** with **0.0873**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	0.803	**0.927**	0.520	0.457
After Hyperparameter Tuning (Bayesian Optimization)	**0.837**	0.774	**0.778**	**0.894**

Table 4.15: Effect of Bayesian Optimization Hyperparameter Tuning on Evaluation Metrics - Heuristic Miner

Notably, the **fitness** score sees a slight increase to **0.837**, indicating a modest improvement in the alignment between the miner's model and the observed data. The **precision** metric demonstrates a notable decrease to **0.774** after the application of Bayesian Optimization for hyperparameter tuning. This decrease is attributed to the algorithm's transition to a

more simplified model structure, which might lead to a reduced capacity for precise pattern identification.

Simplicity shows major improvement, increasing to **0.778**, indicating that the tuned Heuristic Miner produced a simpler model. Lastly, the **generalization** score improves significantly to **0.894**, signifying a notable enhancement in the algorithm's capacity to apply learned patterns to previously unseen data.

In Figure 4.4 we can highly see this enhancements since the resulting Petri Net has a more user-friendly structure.

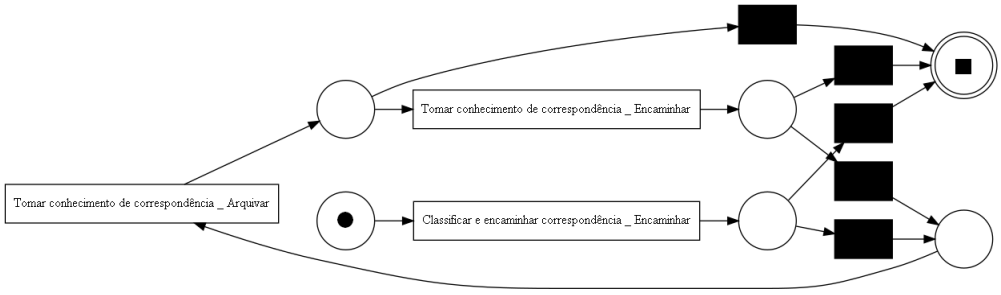


Figure 4.4: Heuristic Miner model from 'Correspondencia' after Bayesian Optimization Hyperparameters Results.

Inductive Miner

Table 4.16 offers a comparative analysis of evaluation metrics for the Inducitve Miner both **before** and **after** undergoing the **Bayesian Optimization** hyperparameter tuning process.

After running the algorithm, the hyperparameters chosen were, **'noise_threshold'** with **0.394**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**1.0**	0.279	0.64	**0.691**
After Hyperparameter Tuning (Bayesian Optimization)	0.944	**0.637**	**0.643**	0.663

Table 4.16: Effect of Bayesian Optimization Hyperparameter Tuning on Evaluation Metrics - Inductive Miner

After interpreting the resulting values, we can see that the four quality dimension results are the same returned by 4.14.

The convergence of results between **Random Search** and **Bayesian Optimization** can be attributed to several factors, Such as the simplicity of the optimization landscape, the characteristics of the hyperparameter space ([0.0,1.0]), and the number of iterations performed by both techniques (20).

4.2.2.5 Genetic Algorithm

In this section, we delve into the use of **Genetic Algorithm** as a hyperparameter tuning method for both the **Heuristic Miner** and the **Inductive Miner**. We conducted **20 generations** of Genetic Algorithm tuning for each miner, with a **population size of 15** in each generation. Genetic Algorithms are inspired by biological evolution and aim to find optimal settings for our algorithms.

In the next section, we'll present the results for **both** miners together, as their evaluation metrics showed **similar trends** to those observed with other tuning methods.

Heuristic Miner and Inductive Miner

Table 4.17 offers a comparative analysis of evaluation metrics for the Heuristic Miner both before and after undergoing the **Genetic** algorithm. After running the algorithm, the hyperparameters chosen were **'dependency_threshold'** with **0.963**, **'and_threshold'** with **0.425** and **'loop_two_threshold'** with **0.181**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	0.803	0.927	0.520	0.457
After Hyperparameter Tuning (Genetic Algorithm)	**0.837**	**0.774**	**0.778**	**0.894**

Table 4.17: Effect of Genetic Algorithm Hyperparameter Tuning on Evaluation Metrics - Heuristic Miner

Table 4.18 offers a comparative analysis of evaluation metrics for the Inductive Miner both before and after undergoing the Genetic algorithm. After running the algorithm, the hyperparameter chosen was **'noise_threshold'** with **0.341**.

Metric	Fitness	Precision	Simplicity	Generalization
Before Hyperparameter Tuning	**1.0**	0.279	0.64	**0.691**
After Hyperparameter Tuning (Genetic Algorithm)	0.944	**0.637**	**0.643**	0.663

Table 4.18: Effect of Genetic Algorithm Hyperparameter Tuning on Evaluation Metrics - Inductive Miner

After analyzing both tables, we can observe both of them have results already retrieved in previous methods. **Heuristic Miner** got the same metrics values for **Genetic Algorithm** (Table 4.17) and **Bayesian Optimization** (Table 4.15), whereas **Inductive Miner** got the same results for all the tested hyperparameter tuning methods (Tables 4.18, 4.16 and 4.14), this convergence can happen for several reasons.

The convergence of results due to factors like **simplicity** of the **optimization landscape**,

characteristics of the hyperparameter **space**, and the **number** of **iterations**, can potentially happen with various optimization methods. If the problem is relatively simple, and the optimization landscape doesn't contain complex interactions or intricate patterns, multiple optimization methods, including grid search, Genetic Algorithms, Random Search, and others, might converge to **similar** or **identical** solutions, especially if they explore a limited portion of the hyperparameter space in a similar way.

Another possible reason that is very important to consider is the possibility of reaching either **local or global optima**. The likelihood of reaching a local or global maximum **depends** on several factors, including the **optimization landscape's complexity**, the choice of **optimization method**, and the **number of iterations** or **samples** explored. **Genetic Algorithms** and **Bayesian Optimization**, are designed to mitigate the risk of getting stuck in local optima by exploring the space more broadly or using probabilistic models. We decided to test enhancing the number of **generations to 50** and **population size to 35**, and obtained the same values, giving even more emphasis that it reached a global maximum.

4.2.3 Results and Analysis

In this section, we provide an overview of the overall results obtained from hyperparameter tuning using various optimization methods for both the Heuristic Miner and the Inductive Miner. The evaluation metrics, including **fitness**, **precision**, **simplicity**, and **generalization**, were significantly influenced by these optimization techniques. The Tables 4.19 and 4.20, provided the compiled results for both demonstrations, **'Faturas Fornecedor'** and **'Correspondencia'**.

Process Mining Algorithm	Metric	Fitness	Precision	Simplicity	Generalization	Weighted Average
Heuristic Miner	Before Hyperparameter Tuning	0.922	0.993	0.517	0.470	**0.725**
	Random Search	0.901	0.995	0.636	0.734	**0.817**
	Bayesian Optimization	0.900	0.995	0.643	0.734	**0.818**
	Genetic Algorithm	0.900	0.994	0.636	0.733	**0.816**
Inductive Miner	Before Hyperparameter Tuning	0.999	0.226	0.617	0.700	**0.635**
	Random Search	0.964	0.511	0.643	0.707	**0.706**
	Bayesian Optimization	0.992	0.540	0.644	0.737	**0.728**
	Genetic Algorithm	0.993	0.540	0.644	0.737	**0.728**

Table 4.19: Effect of Hyperparameter Tuning on Evaluation Metrics - 'Faturas Fornecedor' - Heuristic and Inductive Miners

Process Mining Algorithm	Metric	Fitness	Precision	Simplicity	Generalization	Weighted Average
Heuristic Miner	Before Hyperparameter Tuning	0.803	0.927	0.520	0.457	**0.677**
	Random Search	0.838	0.953	0.587	0.722	**0.775**
	Bayesian Optimization	0.837	0.774	0.778	0.894	**0.821**
	Genetic Algorithm	0.837	0.774	0.778	0.894	**0.821**
Inductive Miner	Before Hyperparameter Tuning	1.0	0.279	0.64	0.691	**0.652**
	Random Search	0.944	0.637	0.643	0.663	**0.722**
	Bayesian Optimization	0.944	0.637	0.643	0.663	**0.722**
	Genetic Algorithm	0.944	0.637	0.643	0.663	**0.722**

Table 4.20: Effect of Hyperparameter Tuning on Evaluation Metrics - 'Correspondencia' - Heuristic and Inductive Miners

Across the board, we observed notable improvements in the performance of both mining algorithms after hyperparameter tuning. All the methods consistently yielded **competitive results** and very good performance. However, in terms of the final **average scores**, **Bayesian Optimization** and **Genetic Algorithm** stood out as the **top-performing methods**. Also to note that, in both demonstrations, the highest reached average was from **Heuristic Miner** algorithm with Bayesian Optimization and Genetic Algorithm.

Random Search, while providing very competitive results, also offered the advantage of being the quickest in terms of runtime, taking only approximately **5 seconds** to complete. This makes it a strong choice for scenarios with time constraints or where quick model iteration is essential. On the other hand, **Genetic Algorithm**, while providing excellent final average scores, exhibited computational overhead. In particular, **Genetic Algorithm** took significantly longer to converge, with compilation times ranging from **30 seconds to as much as 5 minutes** when using a larger number of generations and population sizes.

In summary, the results highlight the effectiveness of hyperparameter tuning in enhancing the performance of process mining algorithms. While **Bayesian Optimization** and **Genetic Algorithm** emerged as the top choices in terms of final average scores, practitioners can also consider **Random Search**, which delivers very competitive results and excels in terms of runtime efficiency. The choice of optimization method should take into account both performance gains and computational time, depending on the specific needs of the application.

Chapter 5

Conclusion

This thesis explored the application of hyperparameter tuning algorithms within the realm of process mining, with a specific focus on the **Heuristic Miner** and **Inductive Miner** algorithms. It aimed to optimize these algorithms by exploring three distinct hyperparameter tuning approaches: **Random Search**, **Bayesian Optimization**, and **Genetic Algorithm**.

The starting point for this investigation was the recognition that manual hyperparameter configuration in process mining can be a **laborious** and **time-consuming** task, limiting **efficiency** and **accuracy**. To address this, we rigorously tested the efficacy of the aforementioned hyperparameter tuning algorithms, each offering its own strategy, whether through **random sampling**, **probabilistic modeling**, or **evolutionary techniques**.

The results of our exploration yielded promising insights into the synergy between hyperparameter tuning and process mining. Our tests demonstrated that these tuning algorithms can indeed enhance the process mining models in all four quality dimensions, fitness, precision, simplicity, and generalization. Notably, automation of hyperparameter selection showcased the potential for improving process models.

After the evaluation of the results, we can conclude that **Bayesian Optimization** and **Genetic Algorithm** retrieved overall the best scores, however, **Random Search**, which delivered very competitive results, was the best in terms of runtime efficiency. In terms of Process Mining algorithms, **Heuristic Miner** got overall better results then **Inductive Miner**.

However, it's crucial to acknowledge that the field of hyperparameter tuning in process mining is still an under-explored field, with significant room for **future growth** and **refinement**. As data volumes expand and workflow systems become more intricate, the demand for advanced hyperparameter tuning methods becomes increasingly apparent.

In conclusion, this research marks a significant stride towards optimizing process mining algorithms. By harnessing the power of **Random Search**, **Bayesian Optimization**, and **Genetic Algorithm**, we open doors to greater **efficiency** and **accuracy** within workflow management systems. While this **journey** represents a substantial **achievement**, the path

ahead promises **further discoveries** and **advancements** within the ever-evolving landscape of process mining.

5.1 Future Work

Future research in this domain may delve deeper into the intricacies of process mining by exploring more **intricate models**. This entails testing algorithms with complex workflows and unconventional patterns to better understand their capabilities in handling real-world complexities. Moreover, manipulating the weights assigned to evaluation metrics is an interesting avenue to pursue. Specifically the simplicity metric. **Adjusting the weights** to further optimize this metric will focus the optimization on **interpretability**, which could lead to the development of models that are not only accurate but also more comprehensible, aligning closely with user expectations.

Additionally, it's crucial to **diversify** hyperparameter tuning algorithms. Future research can broaden the scope by considering **alternative tuning techniques**. Exploring newer algorithms or adaptations tailored specifically to the nuances of process mining could provide fresh insights and potentially superior results. Conducting comparative studies across a wider range of hyperparameter tuning methods will contribute to a more comprehensive understanding of their effectiveness in the context of process mining.

Furthermore, developing a **real-time application** presents an exciting direction for future research. Investigating the integration of automation and adaptive learning techniques could lead to more agile and responsive process mining systems, capable of meeting the evolving needs of modern organizations.

Finally, an interesting avenue for future research would be to conduct a **comparative study** between the evaluation metrics obtained through hyperparameter tuning in process mining and those derived from **manually crafted models** by domain **experts**, offering insights into the efficacy of automated techniques in optimizing process discovery. In this case, instead of trying to maximize the evaluation metrics, we would try to minimize the mode of the difference between the evaluation metrics from hyperparameter tuning with those obtained manually by experts.

Appendix A

'Faturas Fornecedor' Petri Nets

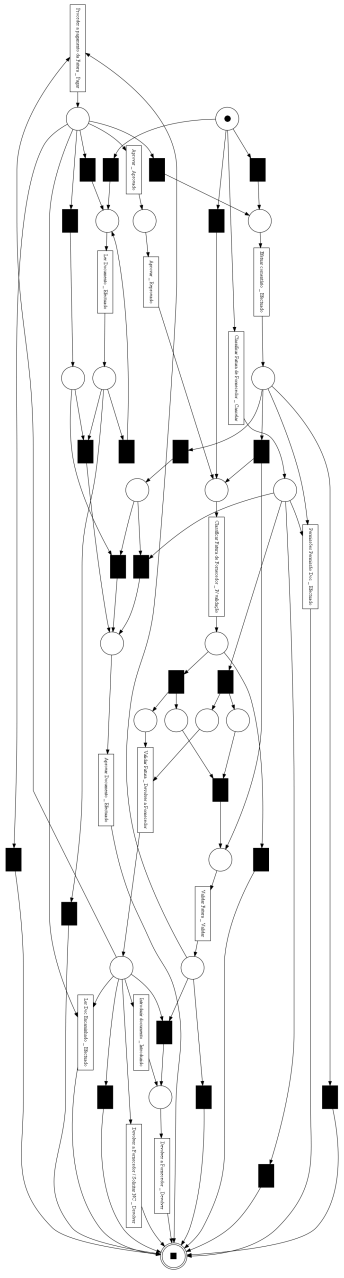


Figure A.1: Heuristic Miner model from 'Faturas Fornecedor' with Default Parameters.

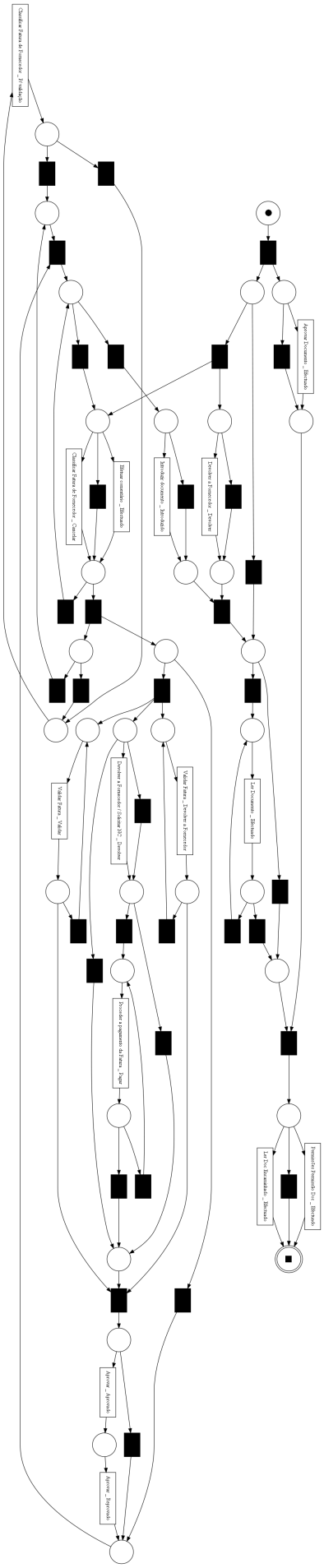


Figure A.2: Inductive Miner model from 'Faturas Fornecedor' with Default Parameters.

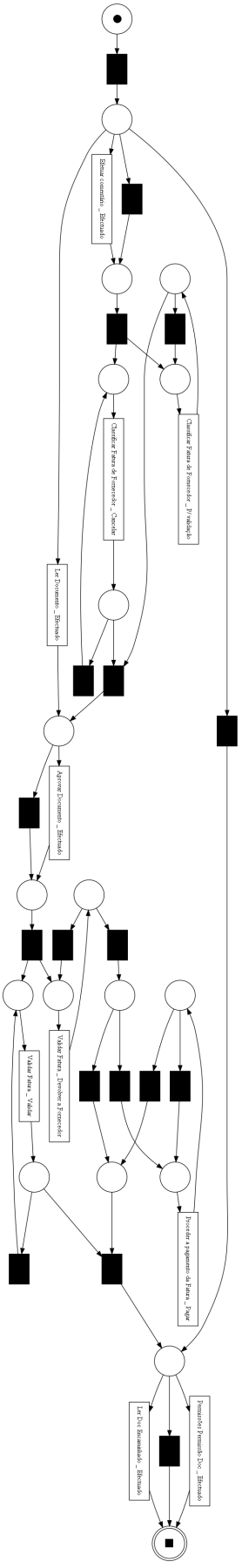


Figure A.3: Inductive Miner model from 'Faturas Fornecedor' after Random Search Hyperparameters Results.

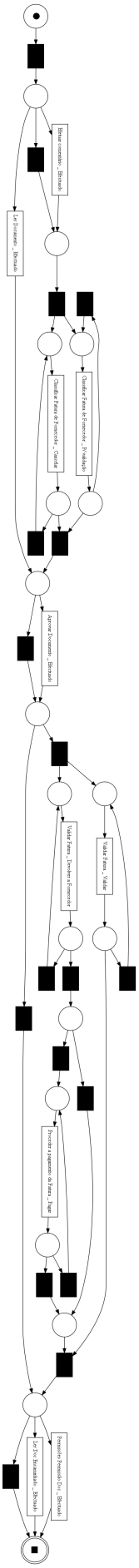


Figure A.4: Inductive Miner model from 'Faturas Fornecedor' after Bayesian Optimization Hyperparameters Results.

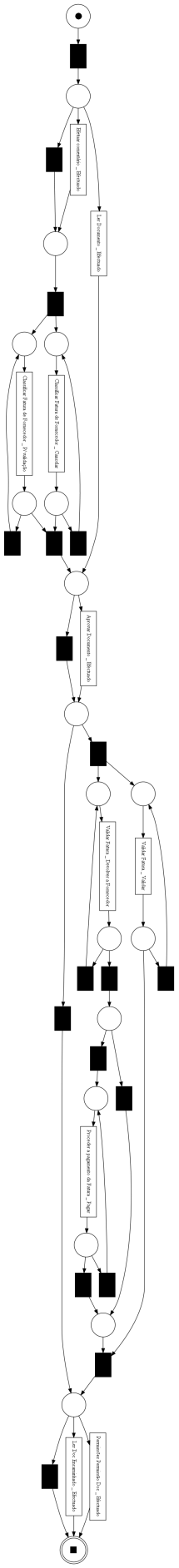


Figure A.5: Inductive Miner model from 'Faturas Fornecedor' after Genetic Algorithm Hyperparameters Results.

Appendix B

'Correspondência' Petri Nets

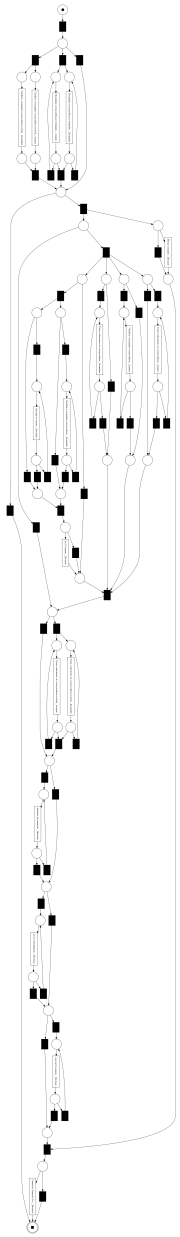


Figure B.1: Inductive Miner model from 'Correspondência' after Genetic Algorithm Hyperparameters Results.

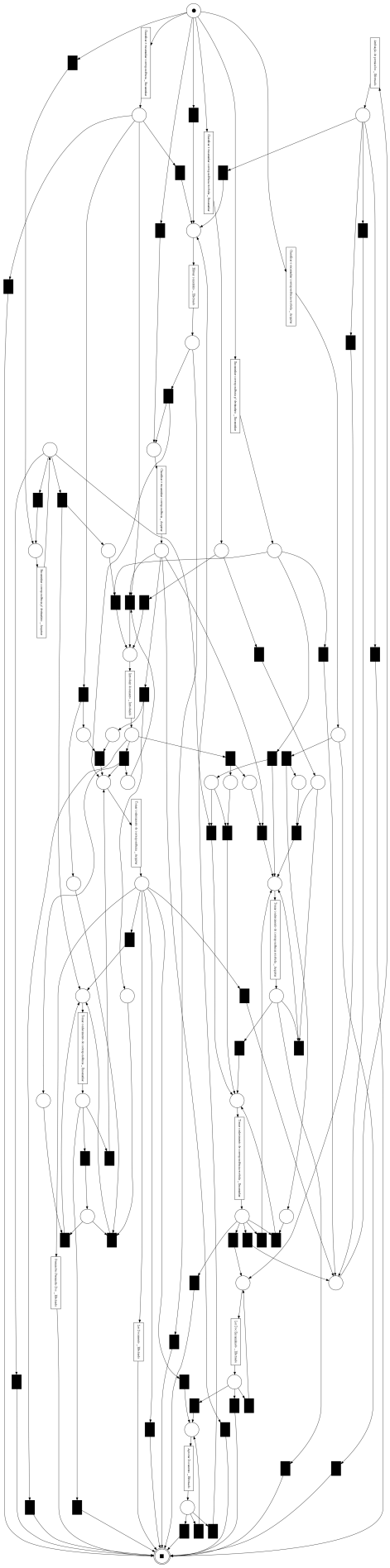


Figure B.2: Inductive Miner model from 'Correspondência' after Genetic Algorithm Hyperparameters Results.

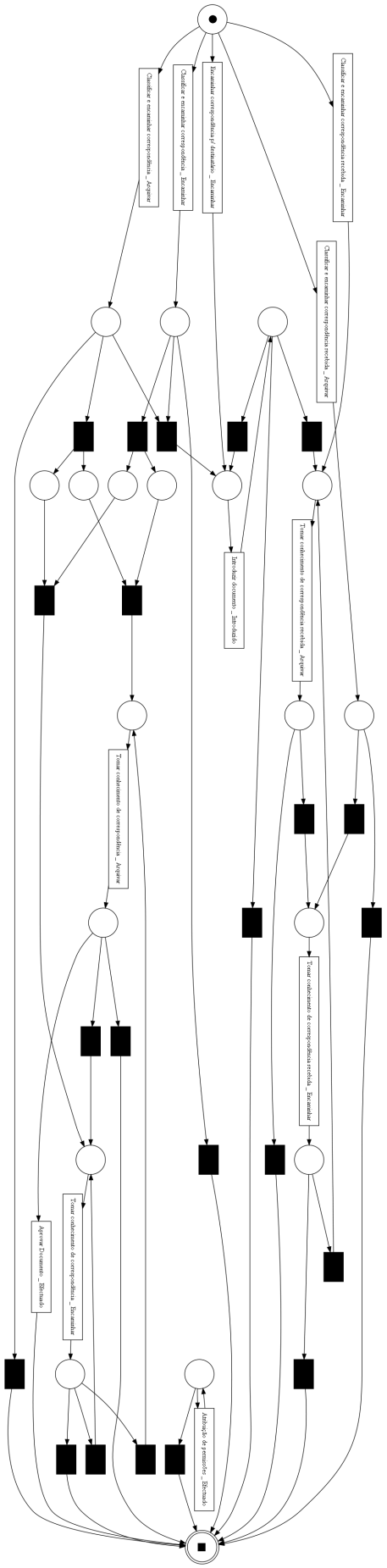


Figure B.3: Heuristic Miner model from 'Correspondencia' after Random Search Hyperparameters Results.

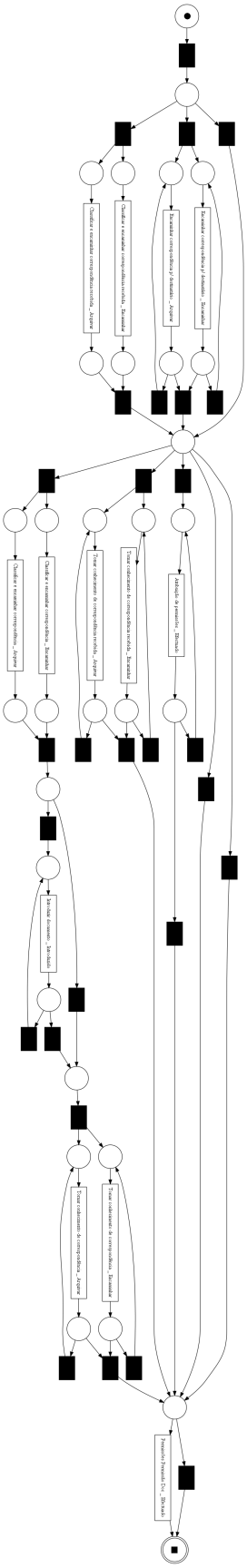


Figure B.4: Inductive Miner model from 'Correspondencia' after Random Search Hyperparameters Results.

Bibliography

[1] Wil Aalst. Process discovery: Capturing the invisible. *Computational Intelligence Magazine, IEEE*, 5:28 – 41, 03 2010. doi:10.1109/MCI.2009.935307.

[2] Wil Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, volume 136. 01 2011. ISBN: 978-3-642-19344-6. doi:10.1007/978-3-642-19345-3.

[3] Wil Aalst. Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems*, 3:7.1–7.17, 07 2012. doi:10.1145/2229156.2229157.

[4] Wil Aalst. *Process Mining: Data Science in Action*. 01 2016. ISBN: 9783662498507. doi:10.1007/978-3-662-49851-4.

[5] Arya Adriansyah, B.F. Dongen, and Wil Aalst. Conformance checking using cost-based fitness analysis. *The Journal of Physical Chemistry*, pages 55 – 64, 10 2011. doi:10.1109/EDOC.2011.12.

[6] Hussain Alibrahim and Simone A. Ludwig. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1551–1559, 2021. doi:10.1109/CEC45853.2021.9504761.

[7] James Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 03 2012.

[8] J. Buijs, B. Dongen, and Wil Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems*, 23, 03 2014. doi:10.1142/S0218843014400012.

[9] Dusanka Dakic, Darko Stefanović, Ilija Cosic, Teodora Vučković, and Milovan Medojevic. *Business Process Mining Application: A Literature Review*, pages 0866–0875. 01 2018. ISBN: 9783902734204. doi:10.2507/29th.daaam.proceedings.125.

[10] DEAP Project. DEAP documentation: Tools module. Accessed: 2023-08-12.

[11] M. Dramski. Extensible event stream format for navigational data. *Zeszyty Naukowe Akademii Morskiej w Szczecinie*, 2016.

[12] Enas Elgeldawi, Awny Sayed, Ahmed Galal, and Alaa Zaki. Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis. *Informatics*, 8, 11 2021. doi:10.3390/informatics8040079.

[13] Dirk Fahland and Wil M. P. van der Aalst. Repairing process models to reflect reality. pages 229–245, 2012.

[14] Matthias Feurer and Frank Hutter. *Hyperparameter Optimization*, pages 3–33. 05 2019. ISBN: 978-3-030-05317-8. doi:10.1007/978-3-030-05318-5_1.

[15] Fraunhofer FIT. Heuristic miner in pm4py, 2023. Accessed: 2023-09-16.

[16] Fraunhofer FIT. Inductive miner in pm4py, 2023. Accessed: 2023-09-16.

[17] Fraunhofer Institute for Applied Information Technology. Understanding process mining, Jan 2021. Accessed: 2023-08-12.

[18] Cleiton Garcia, Alex Meincheim, Elio Junior, Marcelo Dallagassa, Denise Vecino Sato, Deborah Carvalho, Eduardo Santos, and Edson Scalabrin. Process mining techniques and applications - a systematic mapping study. *Expert Systems with Applications*, 133, 05 2019. doi:10.1016/j.eswa.2019.05.003.

[19] Stijn Goedertier, Jochen De Weerdt, David Martens, Jan Vanthienen, and Bart Baesens. Process discovery in event logs: An application in the telecom industry. *Applied Soft Computing*, 11(2):1697–1710, 2011. ISSN: 1568-4946. The Impact of Soft Computing for the Progress of Artificial Intelligence. doi:https://doi.org/10.1016/j.asoc.2010.04.025.

[20] Zhaoyang He, Yuyue Du, Lu Wang, Liang Qi, and Haichun Sun. An alpha-fl algorithm for discovering free loop structures from incomplete event logs. *IEEE Access*, PP:1–1, 05 2018. doi:10.1109/ACCESS.2018.2840818.

[21] IPBRICK. Document management training - understand its importance – iportaldoc, 2022. Accessed: 2023-08-12.

[22] Sander Leemans, Dirk Fahland, and Wil Aalst. Discovering block-structured process models from event logs - a constructive approach. pages 311–329, 01 2013. doi:10.1007/978-3-642-38697-8_17.

[23] Sander Leemans, Erik Poppe, and Moe Wynn. Directly follows-based process mining: Exploration a case study. pages 25–32, 06 2019. doi:10.1109/ICPM.2019.00015.

[24] Wen-Yang Lin, Wen-Yung Lee, and Tzung-Pei Hong. Adapting crossover and mutation rates in genetic algorithms. *J. Inf. Sci. Eng.*, 19:889–903, 09 2003.

[25] Heidy M. Marin-Castro and Edgar Tello-Leal. Event log preprocessing for process mining: A review. *Applied Sciences*, 11(22), 2021. ISSN: 2076-3417. doi:10.3390/app112210556.

[26] Vahideh Naderifar, Shahnorbanun Sahran, and Zarina Shukur. A review on conformance checking technique for the evaluation of process mining algorithms. *Faculty of Information Science and Technology*, 2023.

[27] Phyllalintang Nafasa, Indra Waspada, Nurdin Bahtiar, and Adi Wibowo. Implementation of alpha miner algorithm in process mining application development for online learning activities based on moodle event log data. pages 1–6, 2019. doi:10.1109/ICICoS48119.2019.8982384.

[28] Vu Nguyen. Bayesian optimization for accelerating hyper-parameter tuning. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 302–305, 2019. doi:10.1109/AIKE.2019.00060.

[29] Wenyu Peng, Zhenyu Zhang, Ryan Hildebrant, and Shangping Ren. Empirical studies of three commonly used process mining algorithms. pages 2492–2499, 2021. doi:10.1109/SMC52423.2021.9658861.

[30] Process and Data Science Group RWTH Aachen University. Process mining: Event data. Accessed: 2023-08-12.

[31] Eric Rojas, Jorge Munoz-Gama, Marcos Sepúlveda, and Daniel Capurro. Process mining in healthcare: A literature review. *Journal of Biomedical Informatics*, 61:224–236, 2016. ISSN: 1532-0464. doi:https://doi.org/10.1016/j.jbi.2016.04.007.

[32] A. Rozinat and W.M.P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008. ISSN: 0306-4379. doi:https://doi.org/10.1016/j.is.2007.07.001.

[33] Anne Rozinat, I Jong, C Günther, and Wil Aalst. Conformance analysis of asml's test process. *Mathematics of Computation - Math. Comput.*, 459, 01 2009.

[34] Ashok Saini, Ruchi Kamra, and Utpal Shrivastava. Conformance checking techniques of process mining: A survey. 12 2021. doi:10.3233/APC210213.

[35] Maloth Srinivas and Lalit Patnaik. Patnaik, l.m.: Adaptive probabilities of crossover and mutation in genetic algorithms. ieee trans. syst. man cybern. 24, 656-667. *Systems, Man and Cybernetics, IEEE Transactions on*, 24:656 – 667, 05 1994. doi:10.1109/21.286385.

[36] W. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9): 1128–1142, 2004. doi:10.1109/TKDE.2004.47.

[37] A. Weijters, Wil Aalst, and Alves Medeiros. Process mining with the heuristics miner-algorithm. *Cirp Annals-manufacturing Technology - CIRP ANN-MANUF TECHNOL*, 166, 01 2006.

[38] A. Weijters, Wil Aalst, and Alves Medeiros. *Process Mining with the Heuristics Miner-algorithm*, volume 166. 01 2006.

[39] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. 07 2020.


U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO