



FEP

FACULDADE
DE ECONOMIA
UNIVERSIDADE
DO PORTO

A Recommendation Model for Client Retention: Enhancing Retention in the Telecommunications Industry

Francisco José Almeida Pires

Dissertation

Master in Modelling, Data Analysis and Decision Support Systems

Supervised by:

PhD Bruno Miguel Delindro Veloso

PhD Pedro Alexandre Cardoso

2024

Acknowledgments

This dissertation represents the culmination of hard work, dedication, and a deep commitment to my objectives. A project of this scale could not have been completed without the support and guidance of many individuals.

First, I would like to express my sincere gratitude to my supervisor, Bruno, for his guidance and support throughout this journey.

My heartfelt appreciation also goes to NOS for welcoming me so warmly and providing assistance at every moment. A special mention to Pedro Cardoso, my supervisor at NOS, whose positivity, encouragement, and mentorship were pivotal during these months. I am equally grateful to the entire Analytics Team for their camaraderie, passion, and the supportive environment we cultivated together. A special thanks to Susana Almeida, our team leader, whose exceptional leadership was marked not only by her professionalism but by her genuine care and concern for the well-being of the entire team. I would also like to acknowledge Luis Santos for his constant support and the valuable insights he shared.

To my friends, thank you for your unwavering support, energy, and encouragement during these challenging months.

Finally, to my parents, I extend my deepest gratitude for their lifelong faith, love, and support, which have been my foundation through every step of this journey.

Abstract

In the highly competitive Portuguese telecommunications market, innovative strategies are essential for customer retention, especially as expanding market share becomes increasingly challenging. This dissertation develops a recommender system to identify customers at risk of churn and deliver personalized offers to enhance retention rates. Through a data-driven approach, the two-tower model used in the study processes both customers and offers data separately to generate recommendations. The system is shown to be effective in personalizing services and improving customer retention by matching offers closely to individual customer preferences and behaviors. The study is based on real-world data from a leading telecommunications company and focuses on the Business-to-Business (B2B) segment. One of the main challenges addressed in the project is the "long tail" problem, where many possible offer combinations appear infrequently in the data. This is handled through strategic simplifications, allowing the model to effectively deal with a wide variety of customer profiles and service changes. The dataset used includes customer demographics, service usage patterns, and historical changes in service portfolios. The model was evaluated using A/B testing, comparing its performance to an existing recommendation system, the NBO model, which focuses on loyal customers, and with the as-is which does not have any analytical solution. Unlike the NBO, which primarily recommends service upgrades, the two-tower model also proposes downgraded offers when necessary to retain customers. The initial results demonstrate that the new model tends to be more effective in reducing churn, as it is specifically designed to meet the needs of clients who are not happy with the service or considering leaving due to competitive offers. The research highlights the potential of advanced deep learning models in improving customer retention in the telecommunications industry. Based on the initial results obtained, the two-tower model seems to be a robust solution for personalizing offers and addressing the specific challenges of churn in the B2B sector. Future research could expand on this work by incorporating larger datasets and exploring more complex customer behavior patterns to further enhance the system's accuracy and impact.

Keywords: Recommender Systems; Deep Learning; Two-tower model; Telecommunications; Churn

Resumo

No mercado altamente competitivo de telecomunicações português, são essenciais estratégias inovadoras para a retenção de clientes, especialmente à medida que se torna cada vez mais difícil expandir a quota de mercado. Esta dissertação desenvolve um sistema de recomendação para identificar clientes em risco de *churn* e oferecer-lhes propostas personalizadas para melhorar as taxas de retenção da empresa. Através de uma abordagem orientada por dados, o modelo de duas torres utilizado no estudo processa separadamente os dados dos clientes e das ofertas para gerar recomendações. O sistema demonstrou ser eficaz na personalização de serviços e na melhoria da retenção de clientes, ao alinhar as ofertas de forma mais precisa com as preferências e comportamentos individuais dos clientes. O estudo baseia-se em dados reais de uma empresa de telecomunicações líder de mercado e foca-se no segmento *Business-to-Business* (B2B). Um dos principais desafios abordados no projeto é o problema da distribuição *long-tail*, onde muitas combinações possíveis de ofertas aparecem muito raramente nos dados de treino. Este problema foi tratado através de simplificações estratégicas, permitindo que o modelo lidasse de forma mais eficaz com uma ampla variedade de perfis de clientes e alterações de serviços. O conjunto de dados utilizado inclui demografia dos clientes, padrões de utilização de serviços e alterações históricas nos portfólios de serviços. O modelo foi avaliado utilizando testes A/B, comparando o seu desempenho com um sistema de recomendação já existente, o modelo NBO, que se foca em clientes leais, e com a situação atual, que não possui qualquer solução analítica. Ao contrário do NBO, que recomenda principalmente *upgrades* de serviços, o modelo de duas torres também propõe *downgrade* de ofertas, sempre que necessário, para reter clientes. Os resultados iniciais demonstram que o novo modelo tende a ser mais eficaz na redução de *churn*, uma vez que foi especificamente desenhado para atender às necessidades de clientes insatisfeitos ou que consideram abandonar o serviço devido à existência de ofertas mais competitivas. A análise dos resultados destaca o potencial dos modelos de *deep learning* para melhorar a retenção de clientes na indústria de telecomunicações. Assim, o modelo de duas torres parece ser uma solução robusta para personalizar ofertas e enfrentar os desafios específicos do *churn* no setor B2B. Pesquisas futuras poderiam expandir este trabalho, incorporando conjuntos de dados maiores e explorando padrões de comportamento dos clientes mais complexos, de forma a melhorar ainda mais a precisão e o impacto do sistema.

Palavras-chave: Sistemas de recomendação; *Deep Learning*; Telecomunicações; Modelo Duas Torres; *churn*

Contents

Acknowledgements	iii
Abstract	iv
Resumo	v
1 Introduction	1
1.1 Motivation	1
1.2 Problem Description	2
1.3 Objectives	2
1.4 Dissertation Structure	3
2 Literature Review	5
2.1 Recommendation Systems	5
2.1.1 Collaborative Filtering	6
2.1.2 Content-Based Filtering	7
2.1.3 Hybrid filtering	7
2.1.4 Challenges in Recommendation Systemns	8
2.2 Deep Learning	9
2.2.1 Deep Learning in Recommender Systems	10
2.2.2 Two-Tower Model	12
2.3 Recommender Systems in Telco Churn Context	14
2.4 Summary	14
3 Methodology	15
3.1 CRISP-DM Overview	15
3.2 Business Understanding	15
3.3 Data Understanding	16
3.3.1 Training Panel	16
3.3.2 ITEM_ID	18
3.3.3 Exploratory Data Analysis (EDA)	20
3.3.4 Inference Panel	27
3.4 Data Preparation	28
3.4.1 Cleaning	28
3.4.2 Transformation	29
3.4.3 Feature Selection	29
3.5 Modelling	30
3.5.1 Two Tower	30
3.5.2 Training Stage	31

3.5.3	Inference Stage	34
3.6	Evaluation	35
3.6.1	A/B Test Design	35
3.6.2	Experimental Design	35
4	Results And Discussion	39
4.1	Results	39
4.2	Discussion	40
4.2.1	Answering Research Question 1	41
4.2.2	Answering Research Question 2	41
4.2.3	Answering Research Question 3	41
5	Conclusion	43
5.1	Final Remarks	43
5.2	Future Work	43
	Bibliography	45

Glossary

- AI** Artificial Intelligence. 1
- B2B** Business-to-Business. 3, 15, 16, 30, 31
- CBF** Content-Based Filtering. 7
- CF** Collaborative Filtering. 6
- CNN** Convolutional Neural Networks. 11
- CRISP-DM** Cross-Industry Standard Process for Data Mining. 3
- KPI** Key Performance Indicators. 3
- NBO** Next Best Offer. 2
- PME** Pequena e Média Empresa. 23
- RGU** Revenue Generating Unit. 17, 22
- RNN** Recurrent Neural Networks. 11
- RS** Recommender System. 5, 6

List of Figures

2.1	Neural Network representation	10
2.2	Multi-stage recommendation system (Wortz & Totten, 2023)	12
2.3	Two tower model architecture (Wortz & Totten, 2023)	13
3.1	Training Panel Illustration	17
3.2	Example of ITEM_ID	19
3.3	Distribution of ITEM_IDs highlighting the long tail problem	19
3.4	Monthly Portfolio changes: Number of changes per month from March 2023 to January 2024.	21
3.5	Distribution of the Number of Ps	21
3.6	Distribution of TV Rank, IF Speed, VM Cards, and IM Cards	22
3.7	Distribution of Data Plafonds for DM and IM Components	23
3.8	Distribution of Offer Price Before Change and Days Left in Loyalty Period .	23
3.9	Distribution of Bundle and Mobile Changes	24
3.10	Distribution of P Delta	24
3.11	Distribution of TV Rank Delta and IF Speed Delta	25
3.12	Distribution of VM Cards Delta and Mobile Data Delta	25
3.13	Distribution of Price Delta and Loyalty Period Delta	26
3.14	Distribution of Effort Delta for the New Offer	26
3.15	Distribution of Top 100 ITEM_IDs	27

List of Tables

3.1	Top-k Accuracy for Different Metrics with Mean and Standard Deviation for Dot Product and Euclidean Distance	32
3.2	Top-k Accuracy for Different Frequency Appearance Groups with Dot Product (Test Set Results Averaged Over 3-Fold Cross-Validation)	33
3.3	Top-k Accuracy for Different Frequency Appearance Groups with Euclidean Distance (Test Set Results Averaged Over 3-Fold Cross-Validation)	33
3.4	Test Set Results from the Model Using Dot Product (Best Run)	34
3.5	Comparison between evaluation metrics of the Model Output and Optimization Function test sets	35
4.1	A/B Experiment Results by Strategy	39
4.2	Accepted Offer Breakdown	40

Chapter 1

Introduction

This chapter presents the motivation and problem description behind the study, followed by an outline of the dissertation’s structure.

1.1 Motivation

In Portugal, the telecommunications industry is highly competitive and controlled mainly by three major players: Altice, NOS, and Vodafone. These companies have established a strong market presence, making it very challenging to growth in market share. However, the 2021 auction for the 5G spectrum conducted by the regulatory authority (ANACOM) is expected to bring significant changes to the market. A new competitor, DIGI, is set to enter the scene in 2024 and is anticipated to change the market landscape with their highly competitive pricing.

In a competitive industry, where clients can easily switch to competitors, retaining existing customers is essential due to the high costs of acquiring new ones (Dahiya & Bhatia, 2015). Consequently, fostering customer loyalty and preventing migration to competitors requires offering valuable services grounded in a deep understanding of customer behaviour, preferences, and feedback. This strategy not only acts as a defence against competitive threats but also actively builds a loyal customer base, ensuring sustained long-term growth and stability.

Leveraging Artificial Intelligence (AI) and machine learning is key in this process, as it allows for the enhancement of customer satisfaction and the boosting of company revenues. The telecommunications industry, with its wealth of data—including consumer profiles, usage patterns, and geographical information—offers a unique opportunity to tailor products more precisely to meet customer needs and improve the overall experience.

This dissertation’s main goal is the development of a recommender system for clients marked as prone to churn. A recommender system is an algorithm that suggests a personalized item based on consumer’s past behavior and preferences (Melville & Sindhvani, 2010). To achieve this goal, we employed a two-tower model architecture, a deep learning technique that processes data through two separate neural networks or ”tower”: one corresponds to the users data (consumers) and the other to the item data (service or products) (Wortz & Totten, 2023). The output of the model is a score that results from the dot product of the two embeddings, where the bigger the score, the more tailored to the user is the item, i.e. the better the recommendation.

1.2 Problem Description

This project took place as an internship at NOS, where the main goal was to develop a model to recommend offers to customers who were identified as likely to churn. I was part of the Churn Squad of the Advanced Analytics Department for the Business-to-Business (B2B) segment. The Churn Squad actively identifies clients at risk of churning and signal them to operators that reach out to these clients with new proposals. With the recommendation model, we aimed to assist operators in determining which offers are more likely to be accepted by clients, ultimately extending their loyalty period and increasing their lifetime value. The model will use a diverse set of data from both the users and the offers as input, and recommend the proposals that better suit the client's characteristics. With a more suitable offer, clients will be more likely to renew their loyalty to the company. This model benefits not only the company but also the client who gets a service that is more appropriate to his characteristics.

The Advanced Analytics Department also includes the Next Best Offer (NBO) squad. The NBO Squad has already applied a recommendation model that had a positive impact since its implementation. However, the NBO and the Churn Squad serve different segments of the customer base. The NBO Squad primarily targets stable/loyal clients. Their strategy involves enhancing customer experience by recommending offers that include an upgrade of services and, consequently, an increase in price.

In contrast, the Churn Squad focuses specifically on customers likely to churn. The approach here diverges significantly from the one of the NBO Squad. Instead of only upgrading offers, the Churn Squad also finds it necessary to propose downgraded offers, primarily focusing on the client's retention. This change is crucial because the targeted clients are more prone to leave due to unhappiness or competitive offers from other competitors. Therefore, the churn model operates on a retention-first logic, requiring a deep understanding of churn behavior and a strategic adjustment in the mapping of offers. We aim to demonstrate that simply transpose the NBO model to the churn context is not a good practice since the two model address fundamentally different customer needs and behaviors. Our goal is to show that this new model can be more effective in this context.

1.3 Objectives

The main objectives of this dissertation are designed to guide the development, evaluation, and impact assessment of the recommender system. These objectives can be summarized as follows:

1. Develop and deploy a recommender system using a two-tower model architecture that aims to present the most appealing offers to customers at risk of churn.
2. Evaluate the model's performance specifically in the churn context.
3. Assess the business impact of the new model on customer retention.
4. Compare the performance of the new recommender system with the existing NBO model in the churn context.

Based on these objectives, we have framed the following research questions, which will guide the investigation and analysis throughout this dissertation:

1. How can a recommender system using a two-tower model architecture effectively present the most appealing offers to customers at risk of churn?
2. Is the new recommender system more effective than the existing NBO model in addressing the needs of customers likely to churn?
3. What is the impact of the new model on business outcomes, particularly regarding customer retention?

1.4 Dissertation Structure

In this section, the themes of each chapter of the dissertation are presented to provide a clear roadmap of the research journey undertaken. Each chapter addresses a specific aspect of the study, contributing to a comprehensive understanding of the development of this project and their respective results.

Chapter 1: Begins by presenting the motivation for the study, emphasizing the high competitiveness of the Portuguese telecommunications market and the challenges in customer retention. It outlines the problem description, highlighting the need for innovative strategies to prevent customer churn. The chapter defines the main objectives of the study and also introduces the research questions that guide the dissertation.

Chapter 2: Provides a comprehensive review of existing literature on recommender systems. It covers the fundamental concepts of collaborative filtering, content-based filtering, and hybrid approaches. The chapter explores the application of deep learning in recommender systems, emphasizing its advantages and challenges. A detailed research of the two-tower model architecture is included, explaining its relevance and effectiveness in handling large and complex datasets for personalized recommendations.

Chapter 3: Describes the methodological framework used in this dissertation, based on the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology. It addresses the phases of CRISP-DM, which include: Business Understanding, Data Understanding, Data Preparation, Modeling, and Evaluation.

Chapter 3.2: Outlines the business context in which the study is conducted. It describes the objectives of the project within the Advanced Analytics Department at NOS, focusing on the Business-to-Business (B2B) segment. It also identifies the Key Performance Indicators (KPI) used to evaluate the model, and the constraints and challenges encountered during the project.

Chapter 3.3: Details the processes involved in data collection and preparation. It explains the construction of the training panel and presents an exploratory data analysis (EDA) that provides insights into the data characteristics, distributions, patterns and correlations highlighting key variables and their relationships. It concludes with the constructions of the inference panel.

Chapter 3.4: Describes the steps taken to clean and transform the dataset in preparation for model input. It outlines the data cleaning procedures, including imputation of missing values and the removal of duplicates and outliers. It then covers data transformation techniques and the feature selection process. Finally, the chapter explains the data splitting strategy.

Chapter 3.5: Describes the development of the two-tower model architecture used for the recommender system. It details the training process, including data preprocessing, model

selection, and hyperparameter tuning. The chapter also presents the techniques used to optimize and validate the model, ensuring its robustness and accuracy in providing good recommendations.

Chapter 3.6: Presents the evaluation metrics used to assess the model's performance. It explains the A/B testing design, including the experimental setup, control and treatment groups, duration, and the metrics used to measure success. The chapter discusses the results of the performance analysis, highlighting the strengths and weaknesses of the new model in the churn context and comparing the new recommender system against the existing NBO model.

Chapter 5: Summarizes the results of the two-tower model recommender system, discusses the limitations of the research and suggests potential areas for future work.

The Bibliography section lists all the references and sources cited throughout the dissertation, providing a comprehensive resource for further reading and research.

Chapter 2

Literature Review

Chapter 2 provides a review of the existing literature on recommender systems, focusing on the core principles, methodologies, and recent advancements. This chapter begins by exploring the fundamental types of recommendation systems: collaborative filtering, content-based filtering, and hybrid approaches. It addresses their techniques, strengths, and limitations, along with the primary challenges these systems face.

Then we explore the application of deep learning in recommender systems, as well as the advantages and challenges of integrating deep learning into recommendation systems. Finally, the chapter introduces the two-tower model architecture, explaining its structure and effectiveness in generating personalized recommendations.

2.1 Recommendation Systems

With the development of social media, the rise of the Internet of Things (IoT), and the popularization of smartphones, data generation is everywhere, creating an information-rich environment. However, some challenges, such as information overload, arise in these conditions, making it difficult for individuals and companies to filter relevant content through the vast amount of data available. Recommendation systems are valuable tools in this overwhelming context. They help users navigate through these large amounts of data by personalizing content and making it more accessible and relevant to individuals' needs and preferences.

Recommender System (RS) are algorithms designed to suggest personalized items to users, typically based on their preferences and past behavior. An effective RS must provide recommendations that are relevant, diverse, and personalized. In addition, a robust RS should introduce novelty, not only recommend less popular items, but also introduce a sense of serendipity (the feeling of amazement at a new discovery) (Melville & Sindhvani, 2010). In RS, there are two entities: users and items. Users (individuals) give a rating to an item (product). This rating shows how much the user liked the item, but in some cases, we can have only a 1 or 0 if the user has interacted with the item or not. This information is either collected by implicit or explicit feedback. Implicit feedback is obtained indirectly through the user's interaction with the items, for example, through the tracking of users' browsing and purchasing behavior. Explicit feedback is given directly by users using a value in a range, for example, from 1 to 5 (Roy & Dutta, 2022).

RS are generally categorized into three main types: Collaborative Filtering, Content-Based Filtering, and Hybrid approaches (Roy & Dutta, 2022).

2.1.1 Collaborative Filtering

The Collaborative Filtering (CF) method is the most common technique implemented in RS. This method identifies users with similar tastes and uses their opinions and behavior to suggest items, assuming that similar users like similar items (Isinkaye, Folajimi, & Ojokoh, 2015). The user-item matrix or utility matrix combines the feedback provided by users. This is a two-dimensional matrix of n users and m items, where each entry represents the rating assigned by users to each item. Due to the usual large number of items, the users only interact with a small subset of them, resulting in a very sparse matrix since it contains a lot of missing values. The challenge of this approach is primarily focused on discovering the missing values in the utility matrix (Roy & Dutta, 2022). CF can also be divided into two groups: the Memory-Based and the Model-Based.

The Memory-Based method generates recommendations for user-item pair ratings based on their neighborhood. This neighborhood represents users with similar tastes. CF can be divided into user-based and item-based. User-based focuses on users' similarities. If two users have rated similarly the same items they are considered to have the same taste, so the system recommends items that one user likes to the other. Item-based focus on the relationship between items. The system analyzes how users rated various items and identifies the similarities between them. If two items are highly rated by many users, the system assumes that these two items are similar (Roy & Dutta, 2022). The primary strengths of this technique include its simplicity and efficiency. It is easily justifiable and also tends to be stable, as noted by Shah, Salunke, Dongare, and Antala (2017).

In model-based recommender systems, machine learning and data mining algorithms are employed to create a predictive model for user ratings of unrated items. (Roy & Dutta, 2022). This technique analyzes the user-item matrix to identify relationships between items. These relationships are then utilized to select the top- N recommendations. A key advantage of model-based methods is their ability to effectively address the issue of data sparsity, thus enhancing the overall recommendation quality (Isinkaye et al., 2015). A wide range of algorithms can be applied in this model. For instance, association rules can predict the occurrence of an item based on the presence of another. Clustering techniques segment groups within the data for more efficient recommendations, with methods like K-means and Self-Organizing Maps widely used for this kind of task. With clustering, the recommendations are based on group preferences instead of individual history. Decision trees, known for classifying data based on feature attributes, in spite of being less common when compared to other methods can be used as part of a broader suite of machine learning techniques used in certain types of RS. Neural networks are becoming more and more popular in recommender systems. Their ability to understand and model complex relationships has proven to be very effective in RS, especially when handling large and complex data. (Isinkaye et al., 2015).

As the main advantages Isinkaye et al. (2015) notes that CF techniques have the unique capability to offer serendipitous recommendations, suggesting relevant items that may not be explicitly included in a user's profile. Shah et al. (2017) highlights its ease of implementation and the straightforward nature of incorporating new incremental data. Furthermore, it indicates an increase in prediction performance when utilizing a Model-Based Collaborative Filtering approach. Roy and Dutta (2022) also emphasizes the efficiency of model-based techniques, particularly their ability to make recommendations without the prerequisite of including a new user's profile in the utility matrix. The success of model-based techniques often depends on the effectiveness of the algorithms used. CF has the potential to address issues

like sparsity and scalability through dimensionality reduction and advanced model learning techniques.

2.1.2 Content-Based Filtering

Content-Based Filtering (CBF) relies on the content of items and the user's past behavior to make recommendations. It focuses on item features to generate similar recommendations to the users. These features are represented as a set of descriptors that can be, for example, the genre and the author of a book. CBF combines the user's interactions with the items and models the user's preferences based on the features of the interacted items. The user profiles consist of the items that integrate the profiles (similar items) of the liked items. Similarity can be calculated with Vector Space Model, such as Term Frequency Inverse Document Frequency (TF/IDF), or Probabilistic models, such as Naive Bayes Classifier, Decision Trees, or Neural Networks (Roy & Dutta, 2022)(Isinkaye et al., 2015). Isinkaye et al. (2015) characterizes this technique as domain-dependent since it demands deep domain knowledge to extract meaningful features and generate good recommendations. CBF is most successful when recommending documents like web pages, publications, or news due to their rich textual content.

CBF is able to provide recommendations independent of user ratings, maintaining accuracy even when users have no previous ratings (Isinkaye et al., 2015). It also offers the flexibility to quickly adapt to changes in user tastes (Roy & Dutta, 2022). CBF's capacity to maintain user privacy and provide transparent recommendations make it a valuable approach in some scenarios (Roy & Dutta, 2022). However, this approach requires comprehensive item metadata and can suffer from over-specialization, limiting the diversity of recommendations (Isinkaye et al., 2015)(Shah et al., 2017).

2.1.3 Hybrid filtering

Hybrid filtering is an approach that combines multiple recommendation techniques to improve performance and address the limitations present in single-method systems (Roy & Dutta, 2022). Some of the combinations strategies are:

- **Weighted:** uses a linear formula to aggregate different recommendation scores by assigning a weight to each technique, balancing the strengths of each method.
- **Switching:** switches the recommendation techniques according to a heuristic that reflects the ability of the recommender to produce a good rating.
- **Cascade:** applies an iterative process to the recommendation. The first recommendation outputs a list of recommendations that is then refined by the next recommendation technique.
- **Mixed:** in this approach, the results of different recommendations are presented at the same time. For each user there are more than one recommendation.
- **Feature-Combination:** this method directly uses the output features of one recommendation model (for example, user ratings from collaborative filtering) as input features in another model (for example, in a content-based system). The integration is at the level of individual features being combined across models.

- **Feature-Augmentation:** the output of one model (such as ratings) is augmented with additional data or features to enhance the input set of another recommender system. It's not just about using one model's output as another's input; it's about expanding the feature space of the receiving model with additional, relevant information to improve its recommendations.
- **Meta-Level:** this approach goes beyond just using the output of one model as input for another. Instead, it uses the entire internal model from one technique (for example, the user profile or item embeddings generated by a content-based system) as the input for another technique (for example, to inform a collaborative filtering process). This involves a deeper integration of the underlying structures or representations learned by different models.

This approach has a disadvantage in terms of increased complexity. The process of integrating multiple techniques often requires a more complex setup and fine-tuning, leading to the introduction of more parameters that need to be managed. Determining the right combination of techniques or when to switch between systems can add to the complexity of the process. As a result, the trade-off between accuracy and complexity must be taken into account when designing a hybrid recommender system.

2.1.4 Challenges in Recommendation Systems

As sophisticated as they are, recommender systems face several complex challenges that can significantly affect their efficiency and accuracy. The main challenges encountered by the literature are:

- **Cold Start Problem:** This problem typically occurs when new users or products enter the system due to insufficient data to generate a recommendation. The strategies applied to address this issue include the collection of user preferences and demographic data or employing Bayesian models (Roy & Dutta, 2022) (Fayyaz, Ebrahimian, Nawara, Ibrahim, & Kashef, 2020).
- **Synonymy Problem:** This issue occurs when the system is unable to recognize similar items with different names, like "babywear" and "baby cloth". Techniques like automatic term expansion and Singular Value Decomposition are employed to resolve this (Roy & Dutta, 2022).
- **Sparsity Problem:** This refers to the lack of sufficient user-item interaction data, leading to inaccurate recommendations. Methods to mitigate this include demographic filtering and model-based collaborative techniques (Roy & Dutta, 2022) (Fayyaz et al., 2020).
- **Scalability Problem:** With the rapid addition of items and users in big data environments, recommender systems face challenges in handling large amounts of data. Dimensionality reduction and clustering-based techniques are common solutions (Roy & Dutta, 2022) (Fayyaz et al., 2020).
- **Diversity Issue:** Recommender systems may struggle to balance between suggesting similar items and maintaining diversity in recommendations. Measures like surprisal and personalization are used to evaluate this aspect (Fayyaz et al., 2020).

- **Long-Tail Problem:** A large portion of real-world datasets is characterized by a small number of popular items, while the majority of other items appear much less frequently. Therefore, recommender systems tend to bias their recommendations toward more popular items, while the less popular ones are often ignored (Steck, 2011). These less popular items are not recommended to users due to their lack of representation, which undermines the serendipity needed in recommendation models. Thus, it is critical to implement strategies that enable the recommendation of relevant long-tail items. However, this remains a gap in the literature with limited exploration (Sreepada & Patra, 2021). In this type of problem, improving accuracy for long-tail items often leads to an increase in variance and noise, which degrades overall recommendation accuracy (Steck, 2011).

2.2 Deep Learning

Deep Learning has been successfully employed in many domains in recent years, establishing itself as the state-of-the-art approach due to its capacity to handle complex tasks. This proliferation of deep Learning can be attributed to two main factors (Batmaz, Yurekli, Bilge, & Kaleli, 2019):

- *Big Data* : Deep Learning works better when provided with large amounts of data.
- *Computational power* : Deep Learning can be very computationally expensive, and due to the increase in computation capability, the application of this type of algorithm has become more frequent.

This approach uses artificial neural networks to emulate the workings of the human brain in processing data. It is capable of capturing nonlinear and complex relationships within data. It uses backpropagation and gradient descent in the process of learning deep representations for the data. Deep Learning is a compelling method capable of overcoming the limitations of conventional algorithms.

The neural networks are constituted by input, hidden, and output layers. The input layers are the ones that receive the data and feed it into the model. Each neuron represents one feature in the dataset. The hidden layers are the core of neural networks. This is where the majority of computation is done and where the network learns complex patterns and representations of data. Each neuron in the hidden layers computes a weighted sum of its inputs. This can be mathematically represented as

$$z = \sum_{i=1}^n w_i x_i + b \quad (2.1)$$

where w_i are the weights, x_i are the inputs to the neuron, b is the bias, and n is the number of inputs to the neuron.

The deeper the networks, the more hidden layers the network has. More layers and neurons are able to capture more complex representations of data but can also increase the risk of overfitting and the computational cost; for simpler problems, simpler networks are applied. Each neuron of the hidden layer has an activation function responsible for the non-linearity in the model, enabling the model to learn and represent more complex functions. Some examples of activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh.

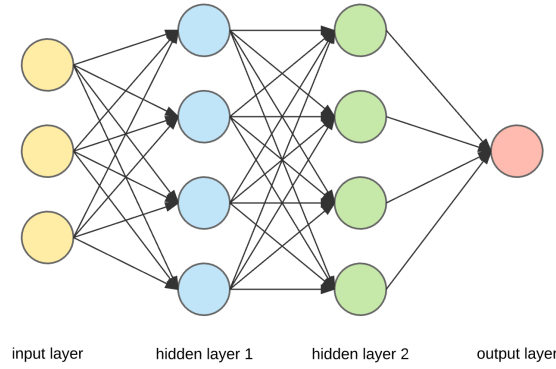


Figure 2.1: Neural Network representation

Lastly, there are the output layers, which are the final layers of the network. It scales the output of the last hidden layer to the format suitable for the task at hand; for regression tasks, a single value; for classification tasks, a probability distribution; and for multi-label classification, multiple values in an array, for example. The configuration of the output, like the number of neurons and the choice of activation function, is determined according to the task requirements for which the network is developed. Figure 2.1 is a representation of a neural network and its respective layers.

The process of passing information on through neurons is called forward propagation. Data is passed from the input layer to the output layer and, at each layer, suffers transformations by a weighted sum of inputs and biases, followed by the application of an activation function. This process enables the network to make complex mappings from inputs to outputs. It is an essential process to capture the complex relationships in the data that are enhanced by the depth and breadth of the network.

Neural networks use backpropagation and gradient descent to learn from the data and improve predictions. Backpropagation calculates the gradients of the loss function for each weight by propagating the gradients backward through the network, from the output layer to the input layer. These gradients indicate how to adjust the weights to minimize the loss.

Gradient descent optimizes the network by taking advantage of the gradients provided by backpropagation. It is an iterative process that adjusts the weights in the direction of the negative gradient to minimize the loss. This is a key step in the training of the neural network, where the learning rate parameter controls the size of the steps toward the minimum and after numerous iterations, the network improves the accuracy of the predictions (Aggarwal et al., 2018).

2.2.1 Deep Learning in Recommender Systems

In recent years, deep learning has greatly impacted recommender systems, changing the architecture of the models used and rapidly becoming the state of the art for this type of task. This is evidenced by the significant increase in the number of publications on recommendation systems using deep learning. As the main strengths of Deep Learning for recommender systems, we have (Zhang, Yao, Sun, & Tay, 2019):

- **Nonlinear Transformation:** As mentioned above, deep neural networks can introduce nonlinearity in the model through activation functions. Contrary to, for example,

matrix factorization, which is a linear combination of user and item latent factors, deep neural networks are able to capture the complex and nonlinear relationship of the data, capturing the user's preferences while increasing the performance of the recommender system.

- **Representation Learning:** Deep Learning is able to learn the best representations for the raw data automatically. It enables us to use a large amount of descriptive information both about items and users, enhancing the performance of the recommender. It is also helpful in the feature engineering process as it can learn by itself from the raw data. Lastly, it makes it possible to use different data types, such as text, images, or audio.
- **Sequence Modeling:** The ability to extract temporal dynamics from user/item interaction evolution, utilizing the unique capabilities of Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) to analyze and predict sequences, makes them indispensable for applications like machine translation and session-based recommendations.
- **Flexibility:** A great advantage of deep learning is its flexibility provided by frameworks like Tensorflow, Keras, and PyTorch. It makes the development much easier and faster.

However, there are also disadvantages to the application of deep neural networks in recommendation systems.

- **Explainability:** A significant drawback of neural networks is the "black box" behavior, where most of the time, it is not possible to know which features influence the final result, making it difficult to interpret the model.
- **Data Needs:** Neural networks are known for needing large amounts of data, primarily to support their rich parameterization. It can make it difficult to acquire sufficient data in terms of quantity and quality while simultaneously avoiding overfitting.
- **Hyperparameter Tunning:** The hyperparameter tuning can become an arduous task as complex networks introduce more hyperparameters, such as the number of layers, activation, and loss functions. Searching the hyperparameter space can take a long time and may not even be feasible.

One example of a successful application of deep learning in recommender systems is given by Steck et al. (2021), which implemented deep learning for Netflix recommendations. Initially, deep learning did not provide a significant improvement over well-tuned traditional methods. However, after incorporating diverse types of features, the deep learning approach significantly outperformed traditional methods. It was also stated that there is no "solution to rule them all," so different techniques are combined for different types of recommendations (i.e., there is an algorithm for the displayed video on the front page and a different one for the "continue watching" section).

Regarding the approaches implemented, the authors divided the models into two main types: bag-of-items and sequential models.

The bag-of-items approach is implemented for modeling long-term interests. There was a transition from Restricted Boltzmann Machines and matrix factorization techniques to autoencoders and asymmetric matrix factorization.

Meanwhile, the sequential models approach takes into consideration the temporal dimension in the user/item interaction. For this task, Recurrent Neural Networks and transformers were used to provide recommendations based on a sequence of interactions.

Deep Learning greatly improved Netflix recommendation, mainly because it enables the use of heterogeneous data, which traditional models struggle with. However, some careful considerations arise, such as sensitivity to objective functions, fairness, and explainability.

When it comes to the implementation of recommender systems for telecommunications recommendations, there is not a great deal of literature on the subject. However, within the company where I am developing my project, the NBO squad has adopted a two-tower model for enhancing recommendations, as already stated above.

2.2.2 Two-Tower Model

Usually, the recommendation task can be divided into two stages: the retrieval stage and the ranking stage. In the retrieval stage, the model selects a subset of relevant items from a large corpus of candidates, which can contain millions of items depending on the domain. Then, in the ranking stage, the model will evaluate and provide the top-k best recommendations to the user (Wortz & Totten, 2023). In Figure 2.2, we have an illustration of this process.

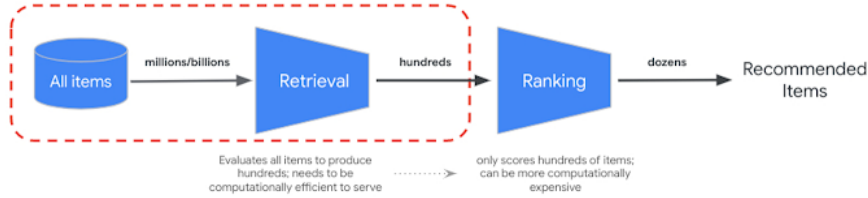


Figure 2.2: Multi-stage recommendation system (Wortz & Totten, 2023)

A new architecture has emerged in the last years that has been successfully implemented in recommendation systems. The Two tower model is a derivation of the dual-encoder framework where two separate encoders process different types of input data independently. As a retrieval task, the two-tower model is a multi-class classification neural network where each tower encodes the users and items, respectively. After the encoding process in each tower, the resulting embeddings are combined through a dot product.

$$s(x, y) = \langle u(x, \theta), v(y, \theta) \rangle \quad (2.2)$$

where u corresponds to the user embeddings, v to the items embeddings, θ to the tower's parameters, and s to the inner product of the two embeddings. The model uses deep neural networks to learn from high-dimensional data, such as user interaction histories and item attributes and effectively represents this information in a lower-dimensional embedding space. This allows the model to capture complex user behaviors and item characteristics in a more computationally efficient manner. The following Figure 2.3 presents a representation of the architecture of the two-tower model.

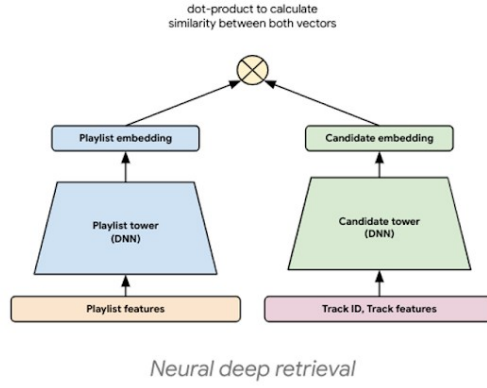


Figure 2.3: Two tower model architecture (Wortz & Totten, 2023)

In training models like the two-tower model, negative sampling is a common technique where the training data includes a mix of items that users have interacted with (positive examples) and items they have not (negative examples). This method helps the model to distinguish between users' likes and dislikes. However, as the number of items grows, traditional negative sampling becomes inefficient due to the increased complexity and volume of data.

To address this challenge, models may employ batch softmax optimization, which estimates each item's probability across a batch. This method avoids the need to compute probabilities against the entire item corpus. However, this method is subject to introducing bias because popular items are overly penalized as negatives due to the high probability of being included in a batch. In order to not get penalized by this limitation, a correction is made, inspired in the logQ correction where the logit: (Yi et al., 2019)

$$s^c(x_i, y_j) = s(x_i, y_j) - \log(p_j) \quad (2.3)$$

Yang et al. (2020) introduced a new technique regarding the sampling aspect of the model. The use of batch negative sampling has some drawbacks, such as the selection bias, as already mentioned, but also a lack of flexibility in adjusting sampling distribution. Instead of using a negative sampling approach, the authors presented a mixed negative sampling, which incorporates "index data," which is a selected subset of negative items from the corpus included in every batch. This method improved the results of the batch negative sampling by mitigating the selection bias. It ensures that every item in the corpus has the potential to be selected as a negative example. Additionally, mixed negative sampling is also able to improve the flexibility in controlling the sampling distribution. By combining a mixture of item frequency-based unigram distribution from training data and uniform distribution from index data, which can be adjusted through the B' parameter.

Despite the challenges associated with training on vast item corpus, such as the potential bias introduced by batch softmax optimization, the Two-Tower model demonstrates robustness through approaches like the logQ correction. This model, stands out as a reliable solution in the literature of recommendation systems, offering a scalable and efficient approach to capturing and predicting user behavior with high precision.

2.3 Recommender Systems in Telco Churn Context

Customer churn refers to the process where subscribers switch from one service provider to another (Berson & Thearling, 1999). In a highly competitive market like telecommunications, managing churn is fundamental and requires strategies focused on retaining valuable customers. However, an effective churn prediction model alone is not sufficient, it is also necessary to develop and implement appropriate retention actions (Hung, Yen, & Wang, 2006).

In the telecommunications industry, key factors driving churn include service quality, network coverage, voice quality, customer service, and value-added services. Pricing issues, outdated technology, inadequate service features, and slow responses to customer complaints are identified as the main reasons for churn. When high-quality service is provided, customers tend to be more loyal and less likely to churn Mahajan, Misra, and Mahajan (2017).

As noted by Gorgoglione and Panniello (2011), most research has focused on predicting and identifying customers prone to churn, while less attention has been given to the actions required to retain these customers specifically (like personalized recommendations for prone to churn clients for example). Their case study concluded that integrating churn prediction models with personalized recommender systems can improve customer retention strategies. However, challenges remain, such as the complexity of customer decision-making.

Renjith (2015) discusses the application of machine learning techniques for churn prediction and highlights the role of recommender systems in designing personalized retention strategies. It focuses on a similarity-based approach to propose retention actions the identified customers. By leveraging on collaborative filtering, the system recommends personalized offers to at-risk customers based on the preferences and behavior of loyal customers with similar profiles.

2.4 Summary

Although there have been significant advancements in recommender systems and the application of deep learning, there still exists a gap in the literature concerning the application and validation of these advanced models, such as the two-tower architecture, specifically for churn reduction context. While the existing research covers general recommendation methodologies and the benefits of deep learning in various contexts, there is a limited focus on how these models can be tailored effectively to address the challenges of telecom churn reduction.

This dissertation aims to bridge this gap by developing and evaluating a two-tower recommender system designed specifically for the telecommunications industry, offering both theoretical insights and practical solutions for improving customer retention. This research intends to contribute to the field by providing a focused analysis of the application of advanced recommendation techniques in telco industry, addressing both technical and business challenges.

Chapter 3

Methodology

This chapter outlines the methodological framework adopted in this dissertation, based on the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology. The CRISP-DM methodology is a widely used framework that provides a structured approach to planning and implementing data mining projects. This chapter details each phase of the CRISP-DM process and explains the specific steps taken in the context of developing and evaluating our model.

3.1 CRISP-DM Overview

The CRISP-DM methodology consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Each phase is interconnected and iterative, allowing for continuous refinement and improvement of the model. The following sections describe each phase in detail as applied to this dissertation.

3.2 Business Understanding

In this chapter, we provide a clear and comprehensive understanding of the business context, outlining the main objectives, the key performance indicators (KPIs) for evaluating our model's performance, and the primary constraints faced in this project.

The main objective of this dissertation is to develop the first recommender system applied to the churn context in B2B at NOS. This model arises in a moment of high competitiveness where a major player is set to enter the market very soon and is expected to cause a considerable impact to the competitive landscape, especially due to the very competitive prices offered by the new competitor. This context underlines the urgency and importance of implementing effective churn reduction strategies to maintain market share. The model will leverage a data-driven approach in recommending the best new offer to clients with high propensity to churn, aiming to retain clients without destroying value through personalized offers and thereby increase retention rates without significantly reducing the price.

As part of its broader strategy, NOS continues to emphasize a data-driven approach to problem-solving. The company utilizes the vast amounts of data at its disposal to guide decision-making, optimize operations, and enhance customer experiences. By capitalizing into extensive datasets, at NOS we are able to identify patterns, predict customer behaviors, and implement solutions that are both efficient and effective. This recommender system

reflects NOS's ongoing commitment to implement data-driven insights to tackle complex business challenges and drive sustained growth.

As the success criteria to evaluate the performance of our model we choose the KPI:

- **Hit-Rate:** This KPI measures the effectiveness of the recommendations in terms of conversion. A higher hit-rate indicates that a significant portion of the recommended offers are accepted by the customers, demonstrating the relevance of the suggestions made by the model.

During the development of this project, we faced two primary constraints:

- **Time Constraints:** The project demanded significant effort across various domains to be completed within the defined timeline. Developing this end-to-end model was a complex and demanding task, requiring meticulous planning and execution to meet the deadlines.
- **Data Availability:** While the two-tower model has been an effective recommender system at NOS, applying it to our specific context of churn prediction in B2B posed challenges due to the limited size of our training dataset. In the B2B sector, we work with significantly less data volume compared to B2C, which presents challenges in training our model and limits our ability to fully develop and optimize it.

3.3 Data Understanding

In this chapter, we explore and understand the data used to build our model. This is a fundamental step in every data science project since a deep understanding of the data is essential for building a robust and insightful model. We will detail the construction of our training panel, including the challenges faced during this data collection stage. Furthermore, this chapter will present an in-depth exploration of the various components of the training panel, including the "client's DNA," the initial portfolio, the portfolio changes, and the importance of the `ITEM_ID` attribute, which serves as the label for the model. We proceed to an exploratory data analysis (EDA) that provides valuable insights into the data, highlighting key patterns and distributions. Finally, we present the steps that led to the construction of the inference panel, which, due to the nature of the model, follows a different logic from the training panel.

3.3.1 Training Panel

Every data science project starts with data collection, a critical step for creating a robust and valuable model. During this stage, several challenges arose. In the operations department, which is responsible for approaching clients with new offers, there is no tracking of the proposed offers. This creates a significant challenge because, in this context, there is no way to know if a client rejected an offer, so we opted to create the training panel only with effective portfolio changes—that is, we will only learn from positive cases, the proposed offers that were accepted by our customers. While this approach ensures the reliability of our positive samples, it inevitably results in the loss of valuable feedback from customers who rejected offers, potentially introducing a bias in the model's learning process.

To construct the training panel, we used a SQL query that gathers information from several profiles, captures the portfolio changes and enriches them with additional contextual information about the user and the portfolio prior to the changes. The scope of changes considered in building this panel includes:

- Changes that extended the loyalty period, since our primary objective is to retain the client with an extension of the loyalty period.
- Both upgrades and downgrades of offers, unlike the NBO model, which only accounts for offer upgrades.
- Changes that either increase or decrease the customer's billing, also different from NBO, which only considers increases in billing.
- Both bundles and standalone changes. Bundles are the usual telecom offers that include the fixed component (TV and Internet) and standalones refer to mobile voice or mobile internet Revenue Generating Unit (RGU)s independent from the bundles.
- Temporal period from 03/2023 to 01/2024

Clients' DNA				Initial Portfolio				Portfolio Alterations				ITEM_ID
Company Info	Data Consumption	...	Churn's Score	Tv	IF	...	VM cards	Tv	IF	...	VM cards	100101



Client Tower



Offer Tower

Figure 3.1: Training Panel Illustration

The training panel is divided into three key parts: the "client's DNA", the initial portfolio, and the portfolio change. Among these, the ITEM_ID attribute holds significant importance, which we will delve into in a dedicated section.

The "client's DNA" is a key part of the training panel, containing all the contextual data about the client. This includes information about the company, such as the number of employees and the sector of activity, the strategic segmentation that categorizes clients based on various strategic factors, competitive landscape information, records of interactions between the client and NOS, and mobile data consumption. This information is fundamental for creating a comprehensive profile and will be used by the model to learn patterns and the client's behavior.

Further, the "initial portfolio" segment captures the state of the client's portfolio before any changes were made. This includes an extensive and detailed description of the services and products included in the initial offer, along with a comprehensive cost segmentation by product type.

Portfolio changes are based on the same attributes as the initial portfolio, with the addition of some new features. However, these attributes are represented using a 'delta logic approach'. This approach focuses on the variations of each component, providing a dynamic view of the portfolio changes.

The clients' tower processes the clients' DNA and the initial portfolio to uncover patterns and understand behaviors based on historical data and contextual details while the offer's tower examines the delta changes in the portfolio as seen in Figure 3.1.

Before proceeding, it is necessary to distinguish between bundle and standalone clients since our model incorporates both bundle and standalone clients. Bundle clients are defined as customers who subscribe to at least one fixed component, such as television (TV), fixed internet (IF), or fixed voice (VF). Standalone clients, in contrast, subscribe only to mobile services, which may include mobile voice (VM) or mobile internet (IM).

It is also essential to provide a clear understanding of the terms "bundle change" and "standalone change", as these can often lead to confusion. A bundle change refers to changes within any component that is part of a bundle, whether the change is in a fixed or mobile component. On the other hand, a standalone change applies to changes within a mobile component that is not part of any bundle.

The integration of data regarding standalone changes requires particular attention. Although these changes concern standalone components, this does not imply that the clients lack a bundled service. Indeed, some clients may subscribe to more than one bundle. To manage this complexity, we decided that for each standalone change, if the client has a bundle, the features related to the bundle components (TV, IF, VF) should correspond to the bundle with the highest billing. This approach ensures that the most economically significant bundle influences the recommendations and insights derived from the model, as well as the identification and characterization of the client. This decision was made with the goal of integrating standalone components into a package, as this is a common recommendation for retention. By bundling the same components, the billing can often be reduced.

3.3.2 ITEM_ID

The ITEM_ID is a critical element of the dataset, serving as the label for the model. This identifier is a code that represents a specific portfolio change, guiding the model in its recommendations. Rather than treating the items as distinct packages, the model interprets them as sequences of component changes. The ITEM_ID is structured as a string composed of a series of digits, each corresponding to a specific offer component.

For the TV component, the digit represents the change in the box's rank. For example, if a client upgrades from an Iris box (rank 2) to a UMA box (rank 1), the digit will be 1, indicating a rank upgrade of one level. For VM and IM cards, each digit reflects the increase or decrease in the number of mobile cards.

For the data components, such as fixed internet (IF), mobile voice data (VM data), and mobile internet data (IM data), the values are categorized into bins. Each digit in the ITEM_ID corresponding to these data components represents a specific bin: for example, for IF an interval of [0-100] mbps is represented by a 1, [101-200] mbps by a 2, and so forth. For VM and IM data components, the values correspond to different intervals but the logic still the same.

The ITEM_ID also has information about the effort of the offer which is binarized through specific bins. Below, an example illustrates the ITEM_ID.

1 0 0 -1 2 0 (0.1,0.25]
 TV IF VM VM IM IM
 card data card data Effort Bin

Figure 3.2: Example of ITEM_ID

In the example shown in Figure 3.2, the ID indicates an increase of one rank in the TV component, a decrease of one VM card, and an increase of two ranks in the data plafond for IM, while all other components remain unchanged, the effort of this offer is between 10% and 25%. This example is just one of many possible IDs that represent changes in the portfolio. The model recommends the ITEM_IDs most likely to be accepted for each user, positioning this as a multi-class classification problem with a large number of classes.

When applied across all possible changes, the logic demonstrated in Figure 3.2 results in a substantial number of ITEM_IDs. However, many of these IDs rarely appear within the dataset, which poses a significant challenge to the model's ability to learn those patterns effectively. This issue can be categorized as an extreme long tail problem. For instance, applying this ITEM_ID logic results in 7413 unique ITEM_IDs, with 63% appearing only once and 86% appearing up to only five times in the dataset. Only a tiny fraction of these IDs are frequent enough to be considered popular, as depicted in the subsequent figure highlighting the distribution of the most common ITEM_IDs.

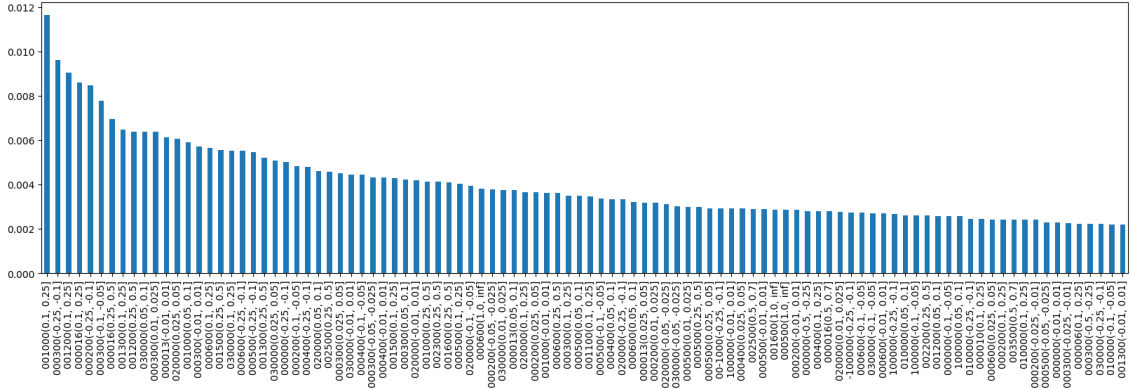


Figure 3.3: Distribution of ITEM_IDs highlighting the long tail problem

To address the challenge posed by the high prevalence of rarely occurring ITEM_IDs, several constraints were implemented to reduce the frequency of these extreme and infrequent changes. Typically, these rare IDs correspond to substantial modifications across multiple components of a portfolio or significant changes within one or more components, such as the addition of 10 VM cards for example.

Additionally, many of these rare IDs exhibit considerable similarities, such as differing only in the number of mobile voice (VM) cards increased for example. To aggregate these extreme changes and similar variations, limitations were applied to the logic governing ITEM_IDs.

Caps were introduced for changes in the number of VM and IM cards. Any increase or decrease beyond ± 3 cards was capped at 3 or -3, respectively. The chosen threshold of 3 was selected because the majority of changes in the dataset fall within this interval, making it a reasonable boundary for capturing most of the significant variations. For data components,

a simpler system was implemented: changes in the mobile data plafond were represented by a flag with digits 1, -1, and 0, indicating increases, decreases, or no change, respectively.

These modifications reduced the number of unique ITEM_IDs from 7413 to 3909, and the proportion of IDs that appeared only once decreased from 63% to 50%. Although this still represents a significant challenge due to a high percentage of rare occurrences, the reduction in the total number of unique IDs offers a more manageable scenario for addressing the multi-label problem.

In this initial approach, we faced multiple challenges, including the integration of both bundle and standalone clients, the small dataset size, the extreme multi-label classification problem, and the long-tail distribution of ITEM_IDs. This strategic decision to simplify the ITEM_ID structure by limiting the granularity of certain changes allows us to focus on the model's performance and understanding core behavioral patterns before introducing additional complexities such as the specifics ranges of mobile data to recommend.

3.3.3 Exploratory Data Analysis (EDA)

This section presents the Exploratory Data Analysis (EDA) for the training panel, providing an overview of the data and extracting valuable insights for the modeling stage.

As previously mentioned, the training panel comprises portfolio changes and includes both contextual and historical features. The dataset contains 47,953 rows and 195 features. Notably, the dataset size of less than 50,000 instances poses a challenge for training a neural network, as this is a relatively small volume of data for such type of model.

With regard to missing values, some features had missing data, but the percentages were minimal. Therefore, these features were retained and imputed. The highest percentages of missing values were observed in the variables TELCO_SPENT_AMT and SALES_AMT, each with around 40% missing values.

We begin our analysis by examining the number of monthly changes. As depicted in Figure 3.4, the highest number of changes occurred in March and May, followed by a period of lower changes of portfolio.

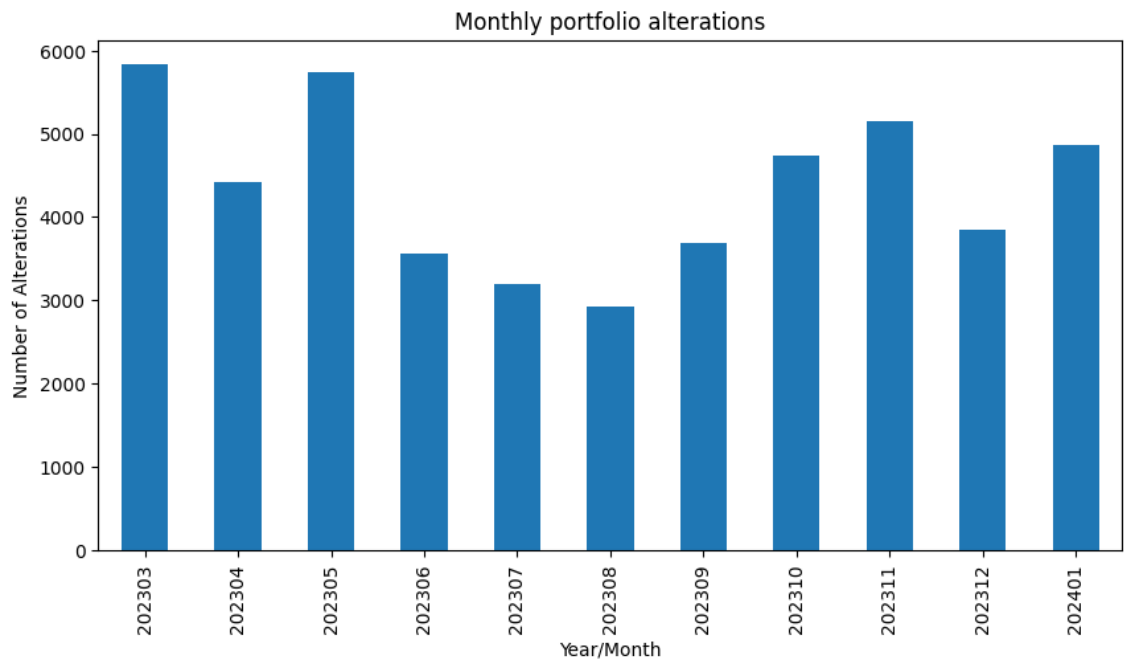


Figure 3.4: Monthly Portfolio changes: Number of changes per month from March 2023 to January 2024.

Looking at the client’s portfolio before changes, Figure 3.5 shows that the majority of clients have a bundle with 4 Ps, around 40%, followed by 30% of clients with bundle of 3 Ps. Approximately 13% of clients have 0 Ps, corresponding to clients with no bundle, just standalone components. The Ps correspond to product families. There can be up to 5 Ps: TV, fixed internet, fixed phone, mobile internet and voice cards.

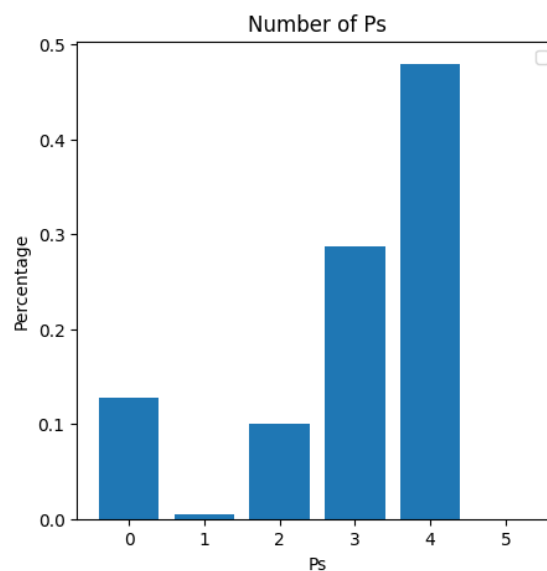


Figure 3.5: Distribution of the Number of Ps

Regarding TV components, almost half (40%) of the clients have the best TV rank. For

IF, most clients have between 200 and 1000 Mbps of internet speed, highlighting a demand for high-speed internet services. For the distribution of VM cards, there is an even distribution between 1 and 4 cards (around 15%/20% each), while more than 20% of clients do not have a single VM card. This plot is limited as there were outliers with unusually high numbers of cards, such as 143, likely corresponding to companies with many employees. Regarding IM cards, almost 70% of clients do not have any IM RGU, and those who do mostly have only 1. Similar to the distribution of VM cards, there are also some extreme outliers with hundreds of IM RGUs.

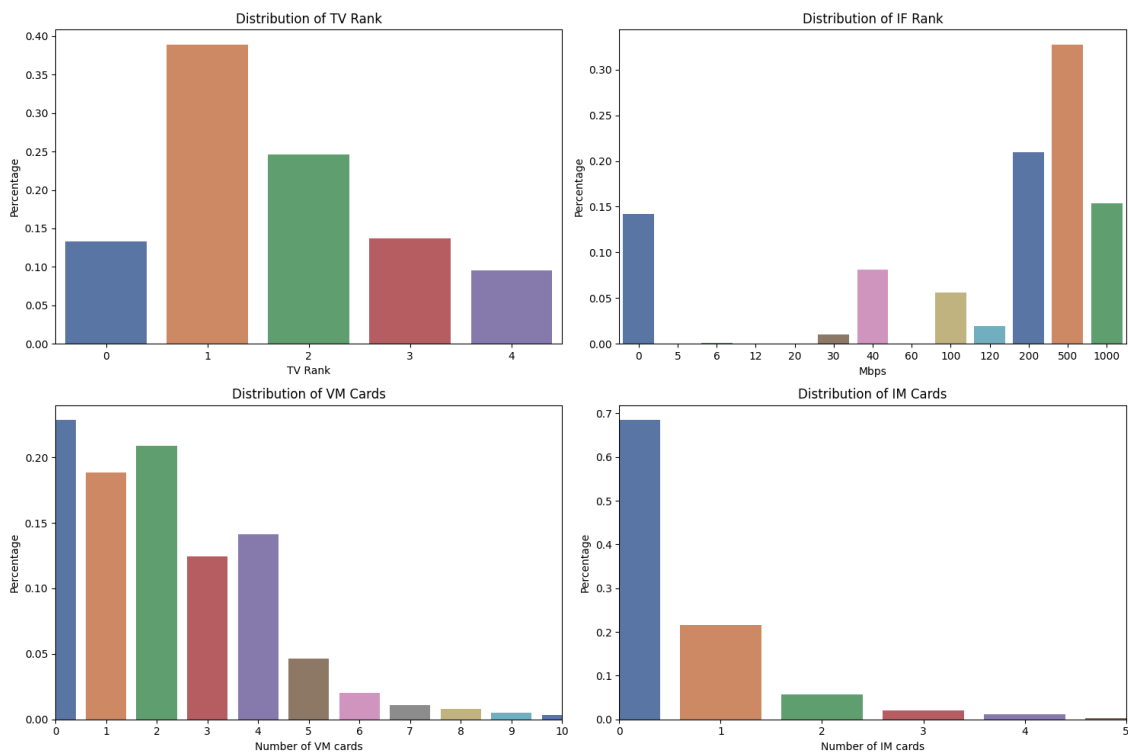


Figure 3.6: Distribution of TV Rank, IF Speed, VM Cards, and IM Cards

In Figure 3.7, we can see the distribution of data plafonds for the mobile components. For DM, most clients have 10, 20, or 30 GBs of mobile data, but 16% of users with VM cards do not have any data plafond. This presents a clear opportunity to upsell data plans to these clients, especially now that there is an increasing demand for this type of service. The rest of the users have a diverse range of plafonds; the figure only presents the most common ones. For the IM component, the scenario is different, with the majority of users having unlimited data plafonds, representing 40% of the cases, and the second most common plafond being 7 GBs with 30%. This indicates a growing preference for unlimited data, further highlighting the increasing need for constant connectivity among clients.

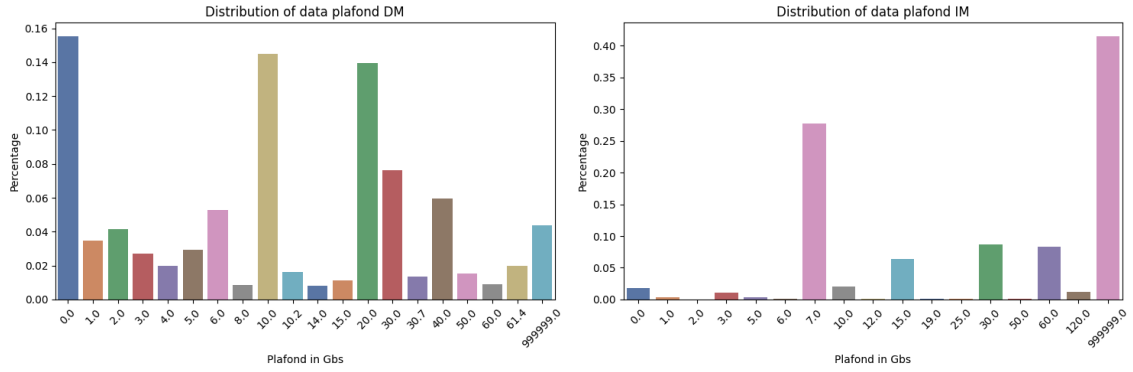


Figure 3.7: Distribution of Data Plafonds for DM and IM Components

In Figure 3.8, we present the distributions of the billing amount before change and the days left in the loyalty period on the day of change. Regarding the billing amount, most are concentrated between 0 and 100, with a significant drop-off as the price increases. Some prices go up to 2779, but these outliers were excluded from the plot for visualization purposes. Regarding the loyalty period, most changes occur for clients whose loyalty period has already expired. The distribution is fairly even for those with remaining days in their loyalty period. There are also some high-value outliers in this case. The presence of high-value outliers in both distributions may suggest a subset of clients with distinct characteristics, potentially representing larger companies within the Pequena e Média Empresa (PME) universe.

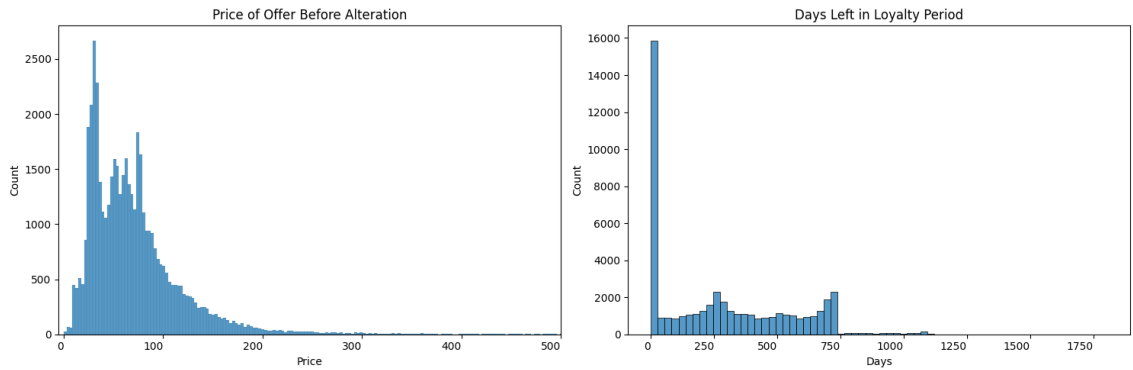


Figure 3.8: Distribution of Offer Price Before Change and Days Left in Loyalty Period

After analyzing our "client's DNA," we now examine the changes in the offers accepted by these clients. In Figure 3.9, we see that 64% of the portfolio changes are "bundle changes," while the remaining 36% correspond to "standalone changes." Additionally, around 60% of all changes include mobile component modifications (VM or IM).

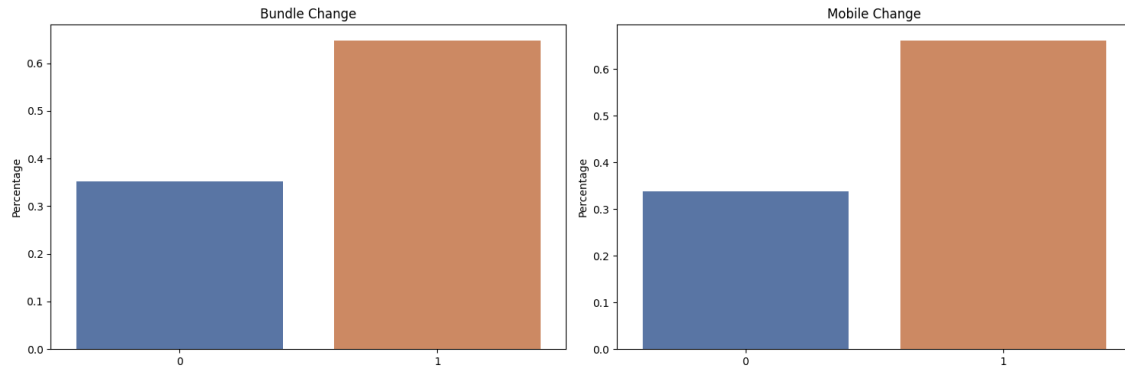


Figure 3.9: Distribution of Bundle and Mobile Changes

For changes in P , we see that most changes maintain the number of P s in the bundle, as the majority of packages already contain 4 P s. When there is an alteration, it is typically an increase or decrease of 1 P .

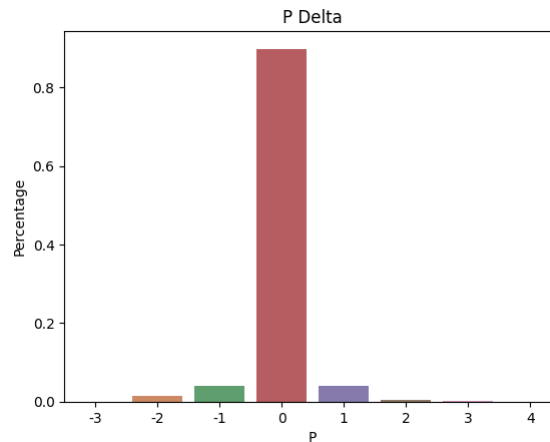


Figure 3.10: Distribution of P Delta

Figure 3.11 shows the distribution of deltas for TV and IF components. We see that most changes do not alter these components. When there is a change in TV rank, it is mainly to increase a rank, while for IF speed, the increase is typically between 300 to 500 Mbps. The peaks at 0 will be present across all components since changes usually occur in one or two components at most, leaving the others unaltered. This creates a significant number of 0 changes across all components delta.

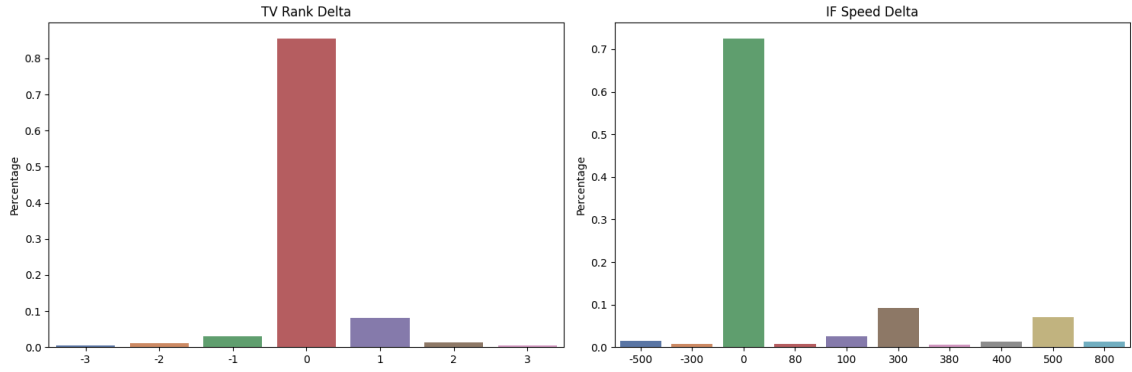


Figure 3.11: Distribution of TV Rank Delta and IF Speed Delta

The distribution of the deltas in mobile components presented in Figure 3.12 also shows a peak in 0 changes, as previously mentioned. When there is a change in VM cards, we see a tendency to increase the number of cards, primarily by one or two. Regarding mobile data, the most common changes are 10 and 30 GBs increases, followed by the unlimited plafond. Only a tiny fraction decreased their mobile data plafond. For IM cards, the increase of 1 card is also the most common, and the most popular mobile data changes are the unlimited plafond data changes and 7 GBs. On the other hand, when considering negative deltas, a significant number of clients are decreasing their mobile data plafond by 7 GBs.

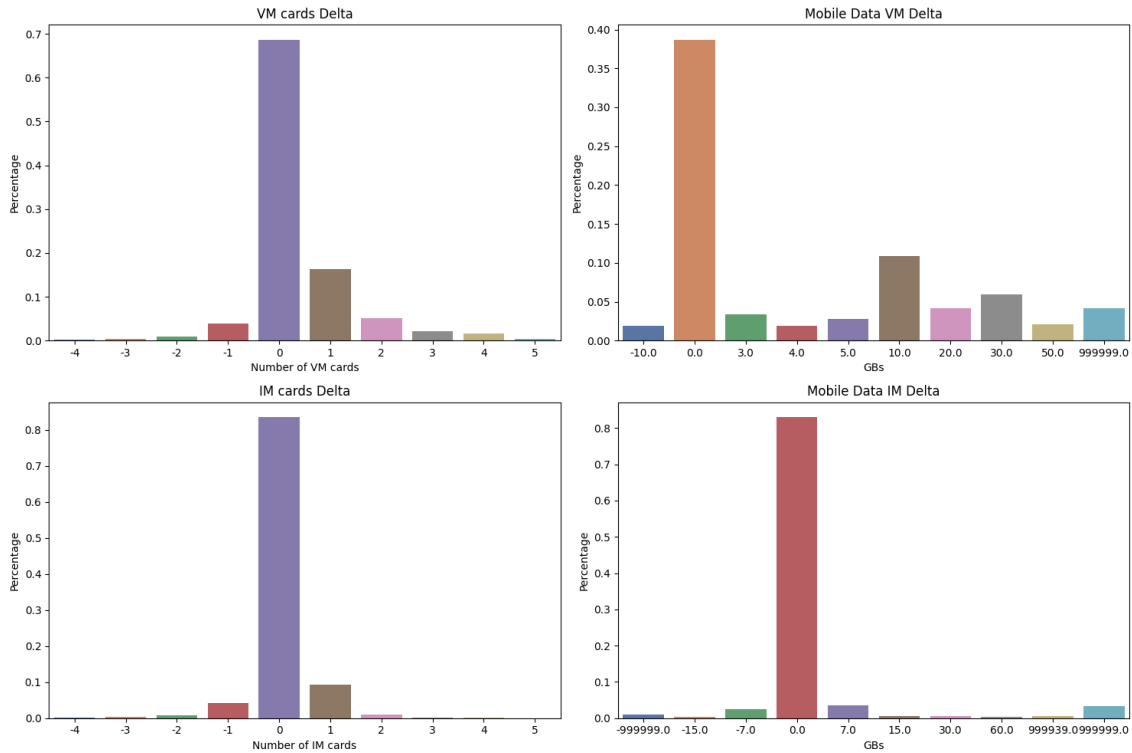


Figure 3.12: Distribution of VM Cards Delta and Mobile Data Delta

In Figure 3.13, we present the price and loyalty period changes. For the price delta, the plot was limited for visualization purposes. A clear concentration of values around 0 indicates that the changes are not very different from the initial prices. The mean price delta is 6.29, with

values ranging from -716.76 to 900. These values represent extreme outliers corresponding to very specific clients. Regarding the loyalty period, except for particular clients, when there is an extension, it is typically for 2 years (731 days), Since most clients have their loyalty period already expired. The distribution mirrors the distribution of days left in the loyalty period in Figure 3.8.

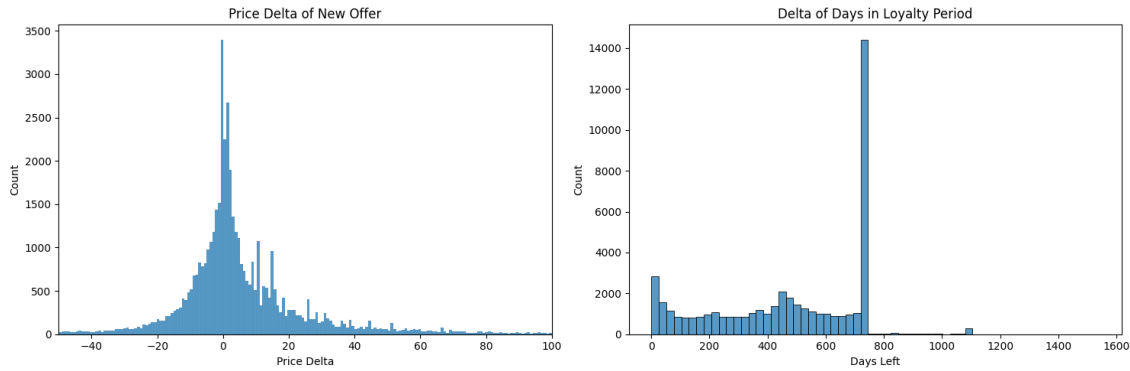


Figure 3.13: Distribution of Price Delta and Loyalty Period Delta

Another valuable variable to analyze in the portfolio changes is the effort required for the new offer. As shown in Figure 3.14, there is a significant concentration of values from -25% to 50%. The mean effort is 22%. This plot was also limited for visualization purposes, as there were some outliers with values as extreme as -65% and 752%.

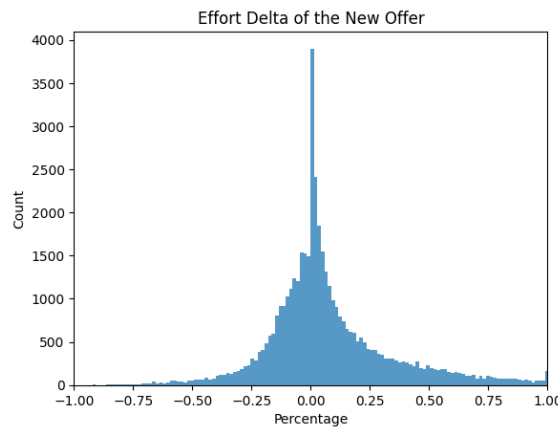


Figure 3.14: Distribution of Effort Delta for the New Offer

Figure 3.15 shows the top 100 most common ITEM_IDs, highlighting one that represents 3.5% of all changes, involving an increment of mobile data for the VM component with an effort between -25% and -10%. The following most popular change include the increment of one VM card combined with an increase in the data plafond, changing only the effort of each offer. The most popular changes are those involving an increase in mobile data for VMs and an increase in VM cards. As previously discussed, the long tail effect is also clearly evident.

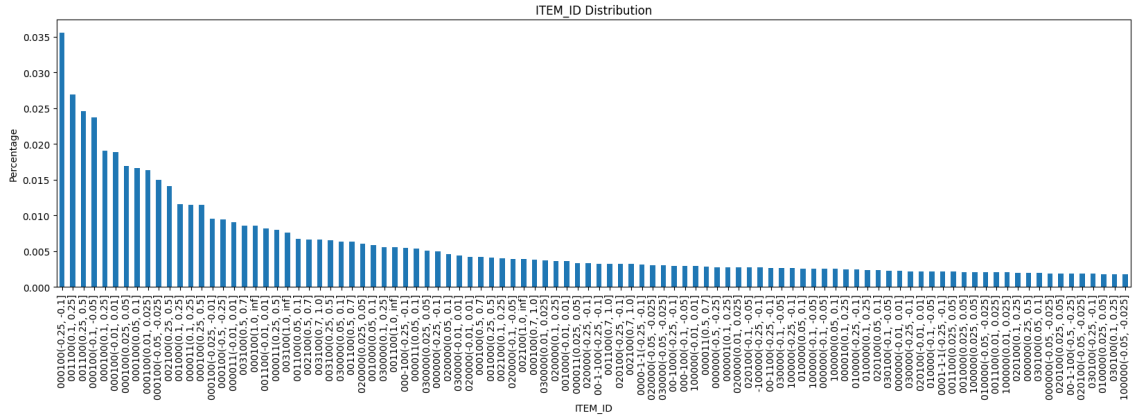


Figure 3.15: Distribution of Top 100 ITEM_IDs

When performing a correlation analysis, we observed that many variables in the training panel exhibited high correlation values. This can be attributed to the different sources of variables used to construct the training panel, which sometimes resulted in redundant information. For instance, price-related variables were segmented by fixed, standalone, mobile component, and final price. We opted to retain only the final price variable, as it is the most relevant for making recommendations. Additionally, the source and destination ranks for each component were initially included to aid in building the training panel and visualizing the differences between old and new offers. However, at this stage, these variables are less useful and contribute to redundancy. By filtering out these variables, we streamlined the training panel, reducing redundancy and enhancing its potential effectiveness for training.

3.3.4 Inference Panel

The inference panel plays a crucial role in the modeling process, as it serves as the input data for generating recommendations. Unlike the training panel, where each instance corresponds to a historical portfolio change, the inference panel is constructed to represent all potential client-offer combinations. As expected, this results in a vast number of possible combinations, leading to a substantial number of instances in the inference panel. However, not all client-offer combinations make sense from a business perspective, so specific filters were applied to refine these combinations.

- **Preserve Technology Level:** To ensure consistent service quality, we avoid downgrading a client's technology level (e.g., clients with FTTH will not be offered DTH).
- **Consistency with Premium Channels:** If a client currently subscribes to premium channels, any new offer should include them; conversely, if they do not have premium channels, the new offer should not introduce them.
- **Effort-Based Discount Limits:** Specific effort intervals are applied according to the remaining loyalty period, setting a maximum level of discount that can be offered to ensure profitability while considering client retention.
- **No Billing Increases with Downgrades:** Offers that increase the billing amount while downgrading components are filtered out, as these would likely be unattractive to clients and could increase churn risk.

An important aspect of this inference panel is its focus on bundle offers due to external challenges in tracking all possible standalone offers. As a result, standalone clients are not included in this panel, although standalone components can be integrated into bundle offers for clients who subscribe to both. This exclusion means that only pure standalone clients will not receive recommendations from this model.

This panel will be used as the input to predict the recommendations. A score will be attributed to each client-offer combination. This score is calculated as the dot product of the client and offer embeddings, indicating the similarity between the pair. These scores will highlight the offers most likely to be accepted by the clients.

This inference dataset includes specific information such as the package names and ID references and their respective prices. This information is crucial for the operation as it identifies the specific pack to recommend to the client. In addition, the inference panel must also include the variables used during the training stage. Similar to the training panel, we enrich our inference panel through an SQL query, incorporating information that characterizes and contextualizes the clients. This process was particularly demanding due to the need to track and match all available offers with the entire customer base in a short timeframe. The primary challenge arose from the sheer volume of potential client-offer combinations, which frequently led to memory errors and significantly slowed down the process, making development both time-consuming and technically challenging. The inference panel ended up with around 28 million instances.

3.4 Data Preparation

In this chapter, we will address all the steps involved in cleaning and transforming the dataset to prepare it for input into the model. This process is structured as a pipeline applied consistently to both the training and inference panels. The pipeline ensures that the data is preprocessed uniformly, enhancing the reliability and robustness of the model's predictions. Additionally, the chapter will dig into the specific techniques and tools used to implement these steps, providing a detailed overview of the data preparation methodology.

3.4.1 Cleaning

The first step in data preparation is cleaning, which ensures that only high-quality data is fed into the model. As discussed earlier in Chapter 3.3, some variables had a considerable amount of missing values, though none exceeded 40%. Therefore, we opted to retain these variables and apply imputation techniques. We used three different imputation methods: mode, mean, and a constant value.

The mode was applied primarily to binary variables, such as flags related to competition, service issues, and campaign outcomes. The median was used for continuous variables like data consumption, the number of days taken to resolve technical problems, and client information such as the number of employees, sales amount, and annual expenditure on telecommunications services. Constant values were employed in specific cases: for the variables `DIGITAL_INVOICE_FLG` and `DIRECT_PAYMENT_FLG`, missing values were imputed with -1. For categorical variables such as `CAE`, `SECTOR_NAME`, `strategic_segment_dsc`, and `offer_segment_dsc`, the string "missing" was used.

Along with imputing missing values, we also dropped all duplicate rows and removed

outliers. For outlier removal, we employed the Isolation Forest algorithm, which is particularly effective in identifying outliers in high-dimensional data. Our target variable was the percentage of effort, so we specifically removed outliers related with this variable to reduce noise in our dataset.

3.4.2 Transformation

Following the imputation of missing values, a series of procedures were applied in the pipeline. We began by converting all categorical variables to lowercase to reduce redundant values when calculating the feature vocabulary. To use the two-tower model with the TensorFlow library, we had to explicitly define the data type of each variable. Thus, categorical variables were assigned the type 'str,' and numerical variables were assigned the type 'float' (np.float64).

Given that we are using a neural network, it is crucial to normalize the numerical variables to ensure efficient training and convergence. We chose to apply the RobustScaler to our numerical features since our features do not adhere to a normal distribution, and several features contain outliers. The RobustScaler is particularly well-suited for this scenario as it scales the data according to the median and interquartile range, making it less sensitive to outliers and better to handle non-normally distributed data. Another essential step in our preprocessing pipeline is calculating the candidate sampling probabilities for each ITEM_ID. This step is vital for applying a correction that mitigates the bias introduced by negative sampling during model training and for the effective implementation of negative sampling. We compute these probabilities based on the frequency of each item in the dataset, ensuring that items are sampled proportionally to their occurrence. This approach helps maintain a fair and effective sampling process, leading to a more robust model.

To conclude the preprocessing steps, we converted our dataset into a TensorFlow dataset format, given that we are using the TensorFlow library. This ensures compatibility and efficiency when training the model.

3.4.3 Feature Selection

For feature selection, we explored various approaches, including algorithms like Boruta, however, due to the extended training time required by our model, Boruta proved inefficient, making it challenging to achieve optimal results. Similarly, other more sophisticated techniques were too time-consuming and inefficient for our needs. Therefore, we focused on selecting the most relevant features from the initial 190 variables through a more practical approach. This selection was driven by a comprehensive business analysis to identify the variables most relevant to the problem, along with the removal of correlated variables and testing different feature sets. We ultimately ended up with 33 numerical features and 3 categorical features for the client tower, and 17 numerical features for the item tower.

To conclude, it is important to address the approach taken for splitting the data into training and test sets. We opted for an 85% allocation for training and 15% for testing. This decision was driven by the relatively small size of our training panel, which required maximizing the amount of data available for training. By allocating a larger portion of the data to training, we aimed to expose the model to as many examples as possible, thereby enhancing its learning capabilities and overall performance.

3.5 Modelling

In this chapter, we will present all the decisions and modeling aspects of this project. We will begin by explain the entire project flow, which includes the training stage, inference stage and post inference processing. This is followed by an explanation of the reasons that led us to choose this architecture, detailing its functionality and the rationale behind our decisions. Additionally, we explore the more sophisticated elements of this architecture, the challenges encountered during the training stage, and the strategies employed to address them. The chapter also covers the hyperparameter tuning process and the methods used in this step. Finally, it concludes with a discussion of the training results and an evaluation of our model's performance.

This project was developed from end to end, beginning with the initial steps of every data science project, including research on the topic and data collection for our model, and culminating in model implementation and evaluation of results in a real-world scenario. Regarding the modeling steps, we can divide them into three main stages:

- **Training Stage:** This is the major part of the modeling process. It involves achieving the best combination of model architecture, methods, and hyperparameters to ensure optimal generalization of our historical data. This stage aims to produce a model that can make accurate predictions on new, unseen data by effectively learning from past data.
- **Inference Stage:** In this stage, we input the data we want to infer into our model. For a recommendation system, this involves providing every possible client-offer combination. The model then generates predictions based on the historical data it encountered during the training stage.
- **Optimization Function:** In this stage, we re-rank the recommendations based on churn score, recommendation score, and price delta to align with business goals. This process ensures that the final output optimizes client retention while preserving revenue, avoiding excessive discounts or value-destroying offers

3.5.1 Two Tower

To address this recommendation system challenge, and based on the literature review conducted in Chapter 2, we decided to implement the dual network architecture, also known as the Two-Tower Model. The dual network architecture represents the state of the art in recommendation systems and is utilized by major companies such as Amazon, Google, Spotify, and YouTube. A variant of this model is also currently implemented at NOS in both the residential and B2B segments within the NBO context, which further reinforced our decision to proceed with this approach.

The Two-Tower Model is a deep learning architecture characterized by the two independent neural networks. One network (tower) processes information related to the client, including historical and contextual data, while the other processes information related to the offers, such as the features of different available offers.

Each network learns the patterns within its respective dataset and transforms this information into vector representations known as embeddings. These embeddings are mapped into a shared vector space, which allows for the comparison of client and offer data. The

similarity between a client and an offer is computed using the dot product of their respective embeddings, which quantifies how closely aligned these vectors are within the shared embedding space.

To train the Two-Tower Model, loss functions that measure the similarity between embeddings, such as contrastive loss or triplet loss, are typically used to ensure that similar embeddings are closer than dissimilar ones. Categorical Crossentropy, on the other hand, is more commonly used for classification tasks, as it measures the difference between predicted and actual probability distributions. This model benefits of a historical client-offer dataset and also techniques like mini-batch gradient descent, regularization (dropout), and optimization algorithms (Adam or Adagrad for example) that enhance performance and prevent overfitting. Research (2024)

To measure the model’s performance metrics such as recall and top k accuracy are commonly employed due to the multi-label nature of the problem. These metrics help evaluate how well the model ranks relevant offers higher than irrelevant ones for each client.

Usually, the Two-Tower Model involves two stages: Retrieval and Ranking. In the retrieval step, the model filters from millions of possibilities to a more manageable subset. The ranking step then uses the output of the retrieval model to re-rank the client-offer pairs, fine-tuning the results to identify the best matches.

One important decision we made is to use only the first stage of the Two-Tower Model, the retrieval stage. The decision not to apply the ranking stage was primarily driven by time constraints, but also because a similar strategy had already been employed within the company. To address this, we will implement a post-inference stage where an optimization function will be applied to re-rank the recommendations, better aligning them with our needs.

Despite the challenge of having a relatively small dataset as seen in Chapter 3.3, especially for a neural network, we have received positive feedback from within the company indicating that it is possible to implement the Two-Tower Model with the available data in the B2B sector.

3.5.2 Training Stage

In the training stage, we explored the hyperparameter space for the model and selected the most appropriate configurations to our project. This process involved an extensive exploration of different approaches and combinations of techniques, ultimately leading to the model we present. To assess the effectiveness of our model, we conducted cross-validation with 3 k-folds. While we typically aim for more folds to ensure robust validation, we limited the number to 3 due to the significant time required for each model run. We primarily used top-k accuracy as our evaluation metric, which measures how often the true label is within the top-K predictions. Additionally, we employed personalized metrics such as top k accuracy for components/effort of the ITEM_ID, so we were able to identify if the model was making good predictions for each individual component. We also evaluate accuracy for less frequent items, to address the long tail problem.

We experimented with various combinations of user and item tower layers, dropout rate, the use of batch normalization, candidate sampling probabilities, embedding size, temperature, the number of hard negatives, and the number of epochs. These principal hyperparameters were thoroughly tested during our tuning process. During this process we noted the importance of weight initialization on the model’s final performance, experimenting with

both GlorotUniform and HeeNormal distributions. The activation function also required tuning as different activation functions yielded significantly different performances, as did the choice of the model optimizer.

As discussed in Chapter 3.3, our dataset exhibits a long-tail distribution, which is common in large recommendation systems. This issue can significantly impact the model’s ability to generalize from less frequent items, so we also explored different solution to mitigate this problem like the following techniques:

- **Euclidean Distance as an Affinity Measure:** We replaced the dot product with the Euclidean distance to improve the differentiation between less frequent items. As a dissimilarity measure, the larger the distance, the more different the two embeddings are. To maintain the same model’s optimization logic as the dot product (the larger the better), we use the inverted Euclidean Distance.
- **Exclusion of Less Frequent ITEM_IDS:** Initially, we trained the model excluding ITEM_IDS that appeared fewer than five times in our dataset. This was done to reduce noise and enhance the model’s focus on more representative data points. However, after thorough evaluation, it was observed that this exclusion did not significantly improve the model’s overall performance. As a result, the final model was trained including all ITEM_IDS, retaining even those with fewer appearances to maintain the model’s generalization capability.

The choice between the dot product and the euclidean distance was a critical decision for our problem. Due to its importance, we present the results and discuss the implications of each choice.

The results in the following tables were obtained for the same hyperparameters, varying only the affinity measure.

Metrics	Dot Product		Euclidean Distance	
	Train Set	Test Set	Train Set	Test Set
Top 1 Accuracy	0.023143	0.020534	0.036039	0.031364
Top 3 Accuracy	0.065189	0.060338	0.065660	0.056821
Top 10 Accuracy	0.147042	0.134459	0.109649	0.102672
Top 50 Accuracy	0.383859	0.351195	0.182316	0.179887
Top 100 Accuracy	0.522009	0.487623	0.215411	0.211533

Table 3.1: Top-k Accuracy for Different Metrics with Mean and Standard Deviation for Dot Product and Euclidean Distance

As we can see in table 3.1 the Dot Product produces better results in every top-k accuracy except for the top 1, indicating that the Dot Product tends to perform better overall. Additionally, the performance increase for higher top-k values is more significant with the dot product compared to the Euclidean distance, which tends to stagnate.

However, these results do not reflect the performance of each affinity measure on long-tail items, which was our primary concern. To address it, we divided our dataset into seven groups based on ITEM_ID frequency. The groups were divided as follows: items with 1

	1	2_5	6_20	21_100	101_200	201_300	+300
Top_1_accuracy	0.000000	0.000000	0.000000	0.001910	0.005133	0.014085	0.045380
Top_3_accuracy	0.000000	0.002079	0.000000	0.008275	0.027721	0.058685	0.127590
Top_10_accuracy	0.003279	0.004158	0.002710	0.033100	0.100616	0.190141	0.263400
Top_50_accuracy	0.009836	0.010395	0.020325	0.173775	0.356623	0.485915	0.609668
Top_100_accuracy	0.022951	0.012474	0.032520	0.334182	0.518480	0.678404	0.789214

Table 3.2: Top-k Accuracy for Different Frequency Appearance Groups with Dot Product (Test Set Results Averaged Over 3-Fold Cross-Validation)

	1	2_5	6_20	21_100	101_200	201_300	+300
Top_1_accuracy	0.032787	0.018711	0.012195	0.010185	0.025667	0.037559	0.050641
Top_3_accuracy	0.072131	0.056133	0.032520	0.032463	0.049281	0.065728	0.076291
Top_10_accuracy	0.232787	0.126819	0.075881	0.074475	0.079055	0.093897	0.114436
Top_50_accuracy	0.485246	0.234927	0.161247	0.145767	0.129363	0.138498	0.178889
Top_100_accuracy	0.557377	0.272349	0.201897	0.180777	0.158111	0.161972	0.202655

Table 3.3: Top-k Accuracy for Different Frequency Appearance Groups with Euclidean Distance (Test Set Results Averaged Over 3-Fold Cross-Validation)

appearance, items with 2-5 appearances, items with 6-20 appearances, items with 21-100 appearances, items with 101-200 appearances, items with 201-300 appearances, and items with more than 300 appearances. The obtained results for each affinity measure are shown below:

By analysing the two tables, we can clearly see the benefits of the euclidean distance compared to the dot product in terms of performance for the long tail ITEM_IDs. The performance is significantly better with euclidean distance for the less frequent items, indicating that recommendations using euclidean distance tend to include more items from the long tail. However, the performance for more frequent items is much worse compared to the dot product, and the rate of performance increase keeps the tendency to stagnate with higher top-k values.

With this analysis, we can state that for less frequent item, euclidean distance does a better job of identifying the correct ITEM_IDs. However, the much lower performance for most frequent item, along with the slower performance increase as top-k values rise, led us to choose the dot product as our affinity measure. The lack of consistent performance increase with higher top-k values is a crucial indicator since we plan to use an optimization function after the model output. Therefore, we expect better results if our higher top-k values perform well, making this criterion decisive in the choice of affinity measure.

After choosing the affinity measure we proceeded to the hyperparameters optimization. To achieve this, we used the Optuna library, which employs Bayesian optimization, to explore hyperparameters such as hidden layers for both towers, batch size, epochs, and dropout rate. We defined reasonable value ranges for each parameter and conducted several trials. The optimal hyperparameters obtained were as follows.

For the user tower, the input layers included one dense layer with the size of the number of numeric features and three embedding layers, each with size 7. The hidden layers comprised two layers, both with size 32. For the item tower, the input layers included one dense layer with the appropriate size and three hidden layers with sizes 64, 32, and 32, respectively. We used batches of 128 samples and applied batch normalization to stabilize and accelerate the training process. Candidate sampling probability was also used to improve training efficiency. The LeakyReLU activation function was employed in both towers, and the Adam optimizer was used for the model. The model was trained with 500 epochs. The model outputs a

matrix (clients X offers), where each value represents the dot product of the embeddings from each tower, indicating the affinity between a specific client and offer. This matrix serves as the foundation for making recommendations, as higher values indicate a stronger predicted match between clients and offers.

Metrics	Test Set
Top 3 Accuracy	0.06033
Component Accuracy	0.27679
Effort Accuracy	0.27468

Table 3.4: Test Set Results from the Model Using Dot Product (Best Run)

In Table 3.4, we present the results from the best training run, highlighting the three metrics used to evaluate our model’s performance: Top 3 Accuracy, Component Accuracy, and Effort Accuracy.

The Top 3 Accuracy metric represents the accuracy of predicting the entire item ID, which includes both components and effort. A value of 6.03% may suggest that the model struggles to accurately predict both components and effort together within the top 3 predictions. However, this accuracy is higher than the frequency of the most popular ITEM_ID in our dataset. Given the clear presence of a long tail distribution and the extreme multi-class nature of the classification problem, a lower Top 3 Accuracy is expected. The most frequent ITEM_ID only appears in a small fraction of the dataset (around 3.5%), making the prediction task particularly challenging. Additionally, the complexity of predicting both the components and the effort simultaneously adds to the difficulty, which is reflected in the relatively low Top 3 Accuracy.

On the other hand, when we examine the components and effort accuracy separately, we observe significantly higher accuracy rates. With a Component Accuracy of 27.68%, the model shows the ability to accurately predict the components of an ITEM_ID independently of the effort. Similarly, with the Effort Accuracy at 27.47%, it indicates the model’s effectiveness in predicting the effort associated with the ITEM_ID. This accuracy is comparable to the Component Accuracy, showing that the model performs consistently well when predicting either the components or the effort independently.

The higher Component Accuracy is particularly encouraging, as it indicates that the model has successfully captured key patterns and preferences related to the components of the offers. This aspect is fundamental for the overall success of the recommender system, as accurately predicting the right components can significantly enhance customer retention.

3.5.3 Inference Stage

After completing the training stage, we proceed to the inference stage, where we generate the model’s recommendations for each client. This involves building the inference panel, which includes all possible client-offer combinations that meet the conditions outlined in Chapter 3.3. This stage is time-consuming, as the model must score approximately 28 million combinations.

Once inference is complete, we apply an optimization function to the model’s output. Given that the two-tower model is used primarily for retrieval, this step helps fine-tune recommendations according to business needs. Additionally, to diversify our offers, we decided

to differentiate the top three recommendations into distinct segments: one focused on mobile voice (RGUs or plafond), one on mobile internet (RGUs or plafond), and one on other components. This decision aligns with business objectives to provide diverse offers.

The optimization function estimates the potential revenue from each offer if accepted or not. It combines the affinity score (model output), price delta, and churn score (from the Predictive Churn model) using the following formula:

$$\text{New_Score} = \left(\text{New_Price} \times \left(\frac{\text{Affinity_Score}}{1 + \%_Effort} \right) \right) - (\text{Old_Price} \times (1 - \text{Churn_Score})) \quad (3.1)$$

To evaluate the optimization function, we compared it against the model's original output using the same metrics as in training, and we also included an expected revenue estimate. The results are shown in the following table:

Metrics	Model Output	Optimization Function
Top 3 Accuracy	0.06033	0.06455
Component Accuracy	0.27679	0.28016
Effort Accuracy	0.27468	0.28452
Income	32267.74	35007.00

Table 3.5: Comparison between evaluation metrics of the Model Output and Optimization Function test sets

The table highlights that the Optimization Function marginally outperforms the Model Output in Top 3 Accuracy (0.06455 vs. 0.06033), Component Accuracy (0.28016 vs. 0.27679), and Effort Accuracy (0.28452 vs. 0.27468). Additionally, the Optimization Function leads to a higher income (35,007 vs. 32,267.74), indicating its potential positive impact on financial outcomes.

Although the Optimization Function appears to offer improved accuracy and financial performance, real-world testing through an A/B Test is necessary to confirm its effectiveness.

3.6 Evaluation

3.6.1 A/B Test Design

To evaluate the effectiveness of our recommendation model we designed an A/B test to measure its impact in client retention. The primary objective of this A/B test is to compare the performance of the recommendation model under different configurations and against a control group to understand if the model has impact in the clients retention.

3.6.2 Experimental Design

To have an effective test it is necessary to design it accordingly with our needs and with our purpose. The experimental design aims to eliminate any potential biases, control for external variables, and provide a clear comparison between the different configurations of the model. This approach allows us to isolate the effects of each configuration and determine their direct

influence on the retention rates of clients. Additionally, by incorporating a control group that does not receive any offers, we establish a benchmark that helps us understand the true impact of the model, independent of any other external factors that could influence client behavior.

Groups Definition

The experiment was structured into four distinct groups, each representing a different configuration of the recommendation model. These groups were defined as follows:

- **Group 1: Model Output Only**

This group receives recommendations directly from the model without any additional optimizations or offer strategies. It serves as a baseline to evaluate the model's effectiveness.

- **Group 2: Model Output + Optimization Function + Typology**

In this group, the model output is refined using an optimization function. This function is designed to incorporate some additional business rules and also try to maximize revenue. Additionally, also following business guidelines our 3 selected offers are from 3 different typologies: VM offer (only changes related with mobile voice (rgus or plafond), IM offer (only changes related with mobile internet (rgus or plafond), all the others offers. With this group, we want to assess the added value provided by the optimization function and business guidelines.

- **Group 3: NBO Offer**

This group uses the offers generated by the NBO model, only upgrades and billing increases. With this group, we want to evaluate the performance of the NBO model in the churn context and compare it with our churn-specific model.

- **Group 4: No Offer (Control Group)**

The control group receives no offers, acting as a benchmark against which the performance of the other groups can be measured. This helps isolate the effects of the recommendation model and compare the as-is with the analytical offer.

Randomization Process

To ensure the validity of the results, all participants were randomly assigned to one of the four groups. Randomization is crucial in eliminating selection bias and ensuring that each group is statistically comparable regarding user behavior patterns. This process guarantees that any observed differences in outcomes can be attributed to the effects of the different configurations rather than other external factors

Metrics Used for Evaluation

The effectiveness of each group will be measured by key performance indicators (KPIs), which in our case are mainly the Retention With Portfolio Alteration - this means that the client changed his portfolio which implies an extension on the loyalty period - and also if the client effectively accepts one of the recommended offers - this is because the client can be retained with portfolio alteration but the change was not one recommended by our model.

The design is structured to accommodate the practical constraints of the project, such as time limitations, while still ensuring that the test yields meaningful insights. Given that the test period was not sufficient to achieve full statistical significance, the obtained results will primarily serve to identify trends and provide preliminary insights. These trends, although not conclusive, will be essential in guiding future iterations of the model and subsequent phases of testing. The focus is on understanding not only whether the recommendation model influences client retention but also how different configurations of the model perform relative to each other and the control group.

Chapter 4

Results And Discussion

4.1 Results

These results were obtained between July 25th and August 29th. Due to the limited scope of this initial experiment and the time constraints imposed by the deadlines for this dissertation, we were unable to gather statistically significant results. Therefore, these findings should be interpreted as indicative trends rather than definitive conclusions. Further experimentation is recommended to achieve more conclusive results.

The outcomes of the A/B testing are presented in Tables 4.1 and 4.2.

	Clients	RCA	RSA	NR	Hit Rate
Model	154	11	20	1	34%
Model + Opt.	165	14	3	33	28%
NBO	120	9	6	18	27%
No offer	154	10	5	25	25%

Table 4.1: A/B Experiment Results by Strategy

Table 4.1 illustrates the number of clients in each group and their respective outcomes. The columns represent the total number of clients (Clients), those who accepted a new offer (RCA - Retained with Changes), those who retained their existing portfolio (RSA - Retained without Changes), and those who rejected any offer (NR - Not Retained). The Hit Rate is calculated as the ratio of clients who accepted a new offer (RCA) to the total number of clients who were contacted (RCA + RSA + NR).

$$HitRate = \frac{RCA}{RCA + RSA + NR}$$

This hit rate reflects the percentage of successful recommendations that resulted in a client accepting a new offer.

It is important to note that not all customers in each group responded to the operators' call; therefore, only customers classified as RCA, RSA, or NR were taken into account when calculating the percentage of RCA, as these were the customers who received an offer.

The Model group shows that out of 154 clients, 11 were classified as RCA, 20 as RSA, and 1 as NR, resulting in a hit rate percentage of 34%. For the Model + Opt. scenario,

the number of clients increases to 165, with 14 classified as RCA, 3 as RSA, and 33 as NR, yielding a hit rate percentage of 28%. In the NBO scenario, 120 clients were contacted, with 9 classified as RCA, 6 as RSA, and 18 as NR, resulting in a hit rate percentage of 27%. The No Offer scenario shows that out of 154 clients, 10 were classified as RCA, 5 as RSA, and 25 as NR, leading to a hit rate percentage of 25%.

These results suggest some differences between the groups, with a trend indicating that analytical models tend to perform better in retaining clients. The best performance is observed in the Model group, followed by the Model + Opt. and NBO groups. The No Offer group, where no offers were made, performs the worst in terms of client retention.

	OFFER 1	OFFER 2	OFFER 3	OTHER OFFER	NO OFFER ID
Model	9%	0%	0%	64%	27%
Model + Opt.	14%	14%	0%	43%	29%
NBO	0%	0%	11%	44%	44%
No Offer	-	-	-	70%	30%

Table 4.2: Accepted Offer Breakdown

Table 4.2 provides a detailed breakdown of the types of offers that were accepted by clients classified as RCA. The table includes columns: Offer 1, Offer 2, Offer 3, Other Offer, and No Offer ID. Each column represents the percentage of accepted offers within each category.

For the Model group, 9% of clients accepted Offer 1, while 64% accepted Other Offers. Additionally, 27% of the offers could not be identified. In the Model + Opt. scenario, there is a higher acceptance of Offer 1 and Offer 2, each at 14%, while Other Offers account for 43%, and 29% of the offers remain unidentified. In the NBO scenario, there are no instances of acceptance for Offer 1 or Offer 2. However, 11% of the clients accepted Offer 3, with Other Offers and No Offer ID each accounting for 44%. The No Offer scenario shows that 70% of the clients accepted offers classified as Other Offers, with the remaining 30% of offers being unidentified.

One notable finding from the analysis of this table is the higher acceptance rate for the recommended offers in the Model + Opt. category. This is promising for the analytical and optimization approach, as it suggests that this method results in offers that are better aligned with customer needs.

While it might seem concerning that Other Offer is the most prevalent category, indicating that recommended offers were not frequently accepted, this can be explained by cases where the accepted offer differed slightly from the recommended ones, either in terms of components or pricing. To better understand this phenomenon, it would be beneficial to implement a similarity measure between the recommended offers and the accepted offers in future analyses. This would allow for a more accurate assessment of the effectiveness of the analytical model. Our observations show that in many cases, this holds true, reinforcing the conclusion that the analytical model outperforms the current approach (No Offer group).

4.2 Discussion

The results presented in this chapter give us some indications about the effectiveness of the two-tower recommender system in addressing customer churn. In this section, we discuss

how these results provide answers to the research questions outlined in Chapter 1.

4.2.1 Answering Research Question 1

RQ1: How can a recommender system using a two-tower model architecture effectively present the most appealing offers to customers at risk of churn?

The two-tower model demonstrated a higher hit rate compared to other strategies, with the Model group achieving a 34% hit rate and the Model + Optimization group achieving 28%. This suggests that the model is effective in presenting offers that appeal to customers at risk of churn. By jointly analyzing customer data and offer characteristics, the two-tower model effectively personalized recommendations, thereby addressing RQ1.

4.2.2 Answering Research Question 2

RQ2: Is the new recommender system more effective than the existing NBO model in addressing the needs of customers likely to churn?

When comparing the two-tower model to the existing NBO model, the results indicate that the two-tower system is more effective in the context of churn prevention. The Model group outperformed the NBO group in terms of hit rate (34% vs. 27%) showing a better alignment with customer needs. The NBO model, which focuses on upselling to stable customers, appears less suited for retaining at-risk customers, although still shows a better performance than no analytical offer at all (group No Offer). Therefore, our model seems to be more effective in addressing the needs of customers likely to churn, providing a positive answer to RQ2.

4.2.3 Answering Research Question 3

RQ3: What is the impact of the new model on business outcomes, particularly regarding customer retention?

The initial findings suggest that the two-tower model positively impacts business outcomes by improving customer retention. The higher hit rates in the Model and Model + Optimization groups imply that more customers accepted new offers, reducing the churn rate, enhancing customer satisfaction, and thus increasing long-term revenue. This addresses RQ3 by highlighting the model's potential to improve key business metrics related to customer retention.

Chapter 5

Conclusion

5.1 Final Remarks

This dissertation successfully developed and evaluated a two-tower model recommender system designed for the telecommunications sector, specifically targeting customer retention among clients at high risk of churn. The research showed that the model was able to generate personalized and relevant offers. This was achieved by aligning offers more closely with customer needs, resulting in improved retention rates and overall customer satisfaction.

However, the project faced several challenges. The dataset used for training the two-tower model was relatively small, characteristic of the B2B sector, which inherently has less available data compared to B2C contexts. This limited dataset restricted the model's ability to generalize, particularly when handling long-tail distributions, where some offer combinations appeared only a few times in the training data. This limitation impacts the model's ability to make accurate recommendations, specially for less common items. During training, the system focused exclusively on positive outcomes (accepted offers). The absence of data on rejected offers introduces a potential bias, as the model is not learning from the full spectrum of customer behavior. Although complex, including these negative samples in future iterations could enhance the model's ability to recommend more suitable offers by learning not only what customers prefer, but also what they tend to reject.

Despite these limitations, preliminary A/B testing results are promising. The two-tower model showed the potential to enhance customer retention by delivering personalized offers, although more testing is required to achieve statistical significance. The observed hit rates and trends indicate that the model is well-suited for the task, and further improvements in data collection and model refinement can enhance its overall impact.

5.2 Future Work

Several areas of this research could be expanded and refined in future work. First, it is essential to continue the experiment to ensure statistical significance and confirm the model's impact on client retention.

Additionally, the project's scope should be broadened to include recommendations for standalone clients, which were not covered in this initial study. This will require a comprehensive mapping of all standalone offers, a challenging task, but one that will significantly increase the range of clients benefiting from the recommendations.

One of the key challenges identified was the relatively small dataset used to train our model. Given that neural networks typically perform better with larger datasets, it is recommended to expand the training dataset size or explore additional data sources. This could improve the model's ability to capture more diverse behavior, patterns and increase its overall effectiveness.

On the technical side, enhancing the model's training process could improve its performance. One potential improvement is to give greater weight to portfolio changes within the retention segment, as this is the key focus of the recommendations. Currently, the model considers portfolio changes across all segments equally. Implementing a ranking stage following the retrieval phase could also improve performance. This ranking stage would replace the current optimization function, potentially leading to more accurate recommendations and better overall results.

Bibliography

- Aggarwal, C. C., et al. (2018). Neural networks and deep learning. *Springer*, 10(978), 3.
- Batmaz, Z., Yurekli, A., Bilge, A., & Kaleli, C. (2019). A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, 52, 1–37.
- Berson, A., & Thearling, K. (1999). *Building data mining applications for crm*. USA: McGraw-Hill, Inc.
- Dahiya, K., & Bhatia, S. (2015). Customer churn analysis in telecom industry. In *2015 4th international conference on reliability, infocom technologies and optimization (icrito)(trends and future directions)* (pp. 1–6).
- Fayyaz, Z., Ebrahimian, M., Nawara, D., Ibrahim, A., & Kashef, R. (2020). Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *applied sciences*, 10(21), 7748.
- Gorgoglione, M., & Panniello, U. (2011). Beyond customer churn: generating personalized actions to retain customers in a retail bank by a recommender system approach. *Journal of Intelligent Learning Systems and Applications*, 3(02), 90–102.
- Hung, S.-Y., Yen, D. C., & Wang, H.-Y. (2006). Applying data mining to telecom churn management. *Expert Systems with Applications*, 31(3), 515–524.
- Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, 16(3), 261–273.
- Mahajan, V., Misra, R., & Mahajan, R. (2017). Review on factors affecting customer churn in telecom sector. *International Journal of Data Analysis Techniques and Strategies*, 9(2), 122–144.
- Melville, P., & Sindhvani, V. (2010). Recommender systems. *Encyclopedia of machine learning*, 1, 829–838.
- Renjith, S. (2015). An integrated framework to recommend personalized retention actions to control b2c e-commerce customer churn. *arXiv preprint arXiv:1511.06975*.
- Research, G. (2024). *Tensorflow recommenders*. Retrieved from <https://www.tensorflow.org/recommenders/> (Accessed: 2024-08-16)
- Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1), 59.
- Shah, K., Salunke, A., Dongare, S., & Antala, K. (2017). Recommender systems: An overview of different approaches to recommendations. In *2017 international conference on innovations in information, embedded and communication systems (iciiecs)* (pp. 1–4).
- Sreepada, R. S., & Patra, B. K. (2021). Enhancing long tail item recommendation in collaborative filtering: An econophysics-inspired approach. *Electronic Commerce Research and Applications*, 49, 101089.
- Steck, H. (2011). Item popularity and recommendation accuracy. In *Proceedings of the fifth acm conference on recommender systems* (pp. 125–132).

- Steck, H., Baltrunas, L., Elahi, E., Liang, D., Raimond, Y., & Basilico, J. (2021). Deep learning for recommender systems: A netflix case study. *AI Magazine*, 42(3), 7–18.
- Wortz, J., & Totten, J. (2023, Apr). *Tensorflow deep retrieval using two towers architecture | google cloud blog*. Google. Retrieved from <https://cloud.google.com/blog/products/ai-machine-learning/scaling-deep-retrieval-tensorflow-two-towers-architecture>
- Yang, J., Yi, X., Zhiyuan Cheng, D., Hong, L., Li, Y., Xiaoming Wang, S., ... Chi, E. H. (2020). Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion proceedings of the web conference 2020* (p. 441–447). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/3366424.3386195> doi: 10.1145/3366424.3386195
- Yi, X., Yang, J., Hong, L., Cheng, D. Z., Heldt, L., Kumthekar, A., ... Chi, E. (2019). Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th acm conference on recommender systems* (pp. 269–277).
- Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1), 1–38.