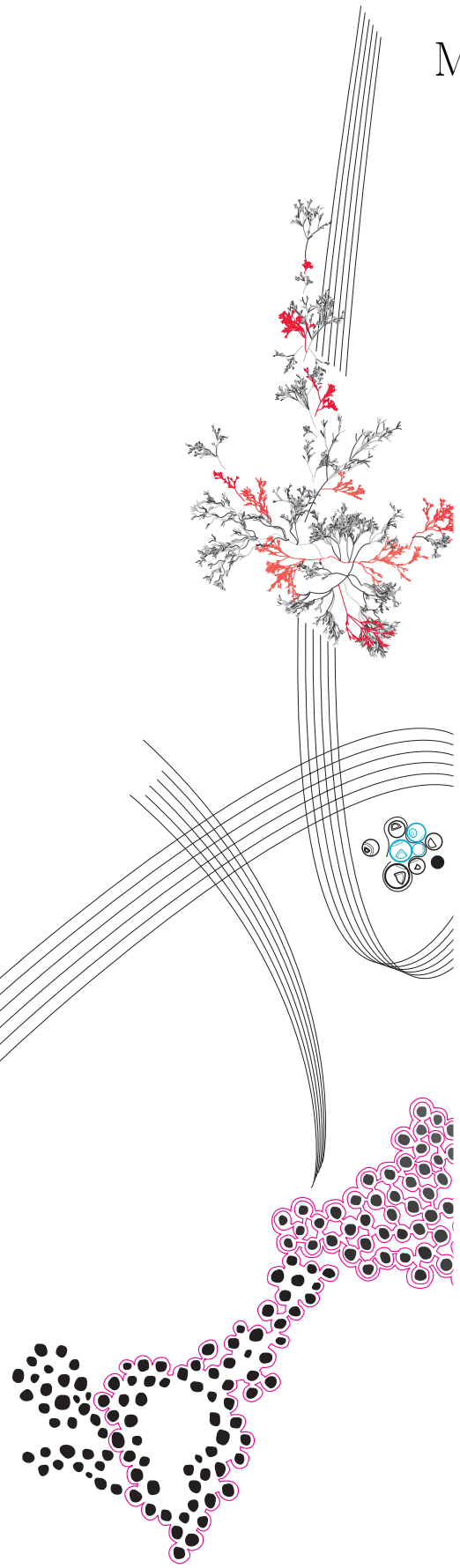


Master Business Information Technology

Final Project

Exploring and Evaluating Alternative Models for Cross-Selling Recommendations

Marzieh Adineh

Supervisor:

- Dr. Fabian Jansen (ING)
- Simon Kaptijn (ING)
- Dr. Maya Daneva (UT)
- Dr. Abhishta Abhishta (UT)
- Dr. Faiza Bukhsh (UT)

July, 2023

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente

UNIVERSITY OF TWENTE.

Over this journey, I learned to be kinder, even to those who excluded me from their selective attention. Hope their kindness expands one day.

Contents

List of Figures

List of Tables

Abbreviation

AUC	Area Under the ROC Curve
BPR	Bayesian Personalized Ranking
CB	Content-Based
CF	Collaborative Filtering
coo-matrix	A sparse matrix in a COOrdinate format
CSC	Compressed Sparse Column
DA	Data Analytics
DAP	Data Analytics Platform
DSRM	Design Science Research Methodology
EDA	Exploratory Data Analysis
FPR	False Positive Rate
indptr	Index Pointer
INGA	ING Analytics
IS	Information Systems
PCM	Payments & Cash Management
RS	Recommender System
SGD	Stochastic Gradient Descent
SLR	Systematic Literature Review
TPR	True Positive Rate
TS	Transaction Services
WARP	Weighted Approximate-Rank Pairwise

WB Wholesale Banking

Glossary

Analytics Maturity	Analytics maturity is a model commonly used to describe how companies, groups, or individuals advanced through stages of data analysis over time.
AUC	AUC is a common abbreviation for Area Under the Receiver Operating Characteristic Curve (ROC AUC). It's a metric used to assess the performance of classification machine learning models.
Decision Trees	Decision Trees are a non-parametric supervised learning method used for classification and regression.
LightFM	LightFM is a Python implementation of a number of popular recommendation algorithms for both implicit and explicit feedback.
Pandas	Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.
ROC	An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate. False Positive Rate.
XGBoost	Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree machine learning library.

Abstract

Today, many companies in the financial services sector face clients who experience a variety of difficulties in selecting the right products that align with their unique needs and preferences. Conventional approaches to product recommendations have demonstrated their limitations in providing suggestions, often leaving customers feeling overwhelmed by the vast array of options or unaware of alternative products that may suit their needs better. The present thesis addresses this challenge. Zooming in on the context of one global financial services company, namely ING which is a global company in the financial services sector, this master project sets out to propose a solution for providing personalized recommendations to customers for cross-selling financial products and maximizing untapped revenue potential. To achieve this objective, we adopted a Design Science grounded research process that included problem analysis, solution design, and solution evaluation.

This thesis makes a valuable contribution to strengthening the Transaction Services (TS) sales cash advisory team at ING by empowering them to offer customized recommendations to customers. The developed recommender system utilizes customer interactions and metadata, bridging the gap in cross-selling supplementary products and stimulating engagement and sales. By tailoring recommendations to individual preferences, the system benefits both new customers in search of suitable products and existing customers aiming to diversify their product portfolio.

The primary objective of this master thesis is to analyze and evaluate the performance of recommender systems in the banking sector, specifically focusing on cross-selling recommendations.

In this thesis, the literature review examines studies conducted in the banking sector to gain insights into the technologies and techniques used for generating personalized recommendations for clients. The literature review conducted in this thesis follows a systematic literature review approach, employing established methodologies and procedures. Various databases were utilized to gather relevant literature. The literature review outlines the methodology used for the review, including the search strategy and data extraction strategy. Moreover, the findings from the literature review indicate that developing recommender systems tailored to the banking domain requires considerations. These requirements provide valuable insights for designing effective recommender systems in the banking industry.

Employing the design science-based research process, the three main parts of this research are as follows. The problem analysis first explores the difficulties that banking organizations encounter when trying to promote appropriate products to clients. It focuses on the significance of customized cross-selling suggestions to raise client satisfaction and boost sales.

Second, a comprehensive strategy leveraging recommender system techniques, like collaborative filtering and content-based models, is proposed by the solution design. The design intends to produce personalized product recommendations by leveraging consumer data and preferences. The LightFM library is utilized to build a hybrid recommendation system that utilizes user-item interactions and metadata to create embeddings that capture individual user profiles and product characteristics. The suitability of the LightFM model, as well as other models like XGBoost and Decision Tree, is explored.

Thirdly, the performance of the developed recommender is evaluated through a series of experiments and comprehensive data analysis. Two experiments are conducted to test the model's performance: Experiment 1 involves training the model on randomly selected samples of user-item interactions and evaluating its performance on the remaining interactions. Experiment 2 explores the model's ability to perform well after being trained on a more diverse set of user-item interactions. Evaluation metrics, particularly the area under the curve (AUC), are utilized to assess the recommender system's performance. By assessing the AUC scores, the performance of the developed model in accurately recommending relevant products is determined. The results of the experiments demonstrate the model's ability to provide accurate recommendations, thus addressing the challenge of product selection and cross-selling in the banking industry.

Keywords: *ING, cross-selling recommender system, collaborative filtering, content-based models, problem analysis, solution design, evaluation, LightFM, hybrid matrix factorization model, feature engineering, XGBoost, Decision Tree, user-item interaction, experiment, AUC score.*

Exploring and Evaluating Alternative Models for Cross-Selling Recommendations

1 Introduction

In today's digital era, recommender systems have become indispensable tools for enhancing sales and improving customer satisfaction in various domains, including e-commerce. These systems leverage advanced algorithms and user data to predict and recommend the most relevant items or services to users, increasing the likelihood of successful sales conversions. While recommender systems have been widely implemented in these industries, their adoption in the banking sector presents a unique opportunity to drive product sales and revenue growth.

Banking institutions offer a wide range of products and services to cater to their client's evolving needs. The difficulty, however, lies in successfully selling and marketing the correct products to the proper clients. Traditional methods of product recommendations often fall short in providing personalized and targeted suggestions to individual customers, resulting in missed opportunities and reduced customer satisfaction. To address this challenge, the integration of recommender systems in the banking industry holds immense potential to revolutionize cross-selling practices and boost product sales.

The primary objective of this master thesis is to analyze and evaluate the performance of recommender systems in the banking sector, specifically focusing on cross-selling recommendations. The research project aims to address the following key components:

As a problem analysis, it will delve into the existing challenges faced by banking institutions in effectively promoting and selling the right products to customers. It will explore the limitations of traditional recommendation methods and highlight the need for personalized cross-selling recommendations to enhance customer satisfaction and drive sales.

Regarding solution design, the master thesis proposes a comprehensive solution design by utilizing state-of-the-art recommender system algorithms, including collaborative filtering and content-based models. The design will incorporate advanced techniques for leveraging customer data, such as transaction history, and user preferences to generate personalized product recommendations. Specifically, we will analyze an ING dataset to identify common attributes from a subset of clients with certain financial products and find common in the product criteria to indicate which clients could benefit from additional ING solutions. The objective of the master thesis is to build a recommender system using the LightFM library, which combines

collaborative filtering and content-based models to accurately recommend products to customers.

For solution evaluation, the performance of the proposed recommender system will be evaluated through a series of experiments and data analysis. The performance of the system will be measured using evaluation measures such as the area under the curve (AUC).

By tracing back to these three components, the research project aims to provide valuable insights into the potential of the recommender in the banking sector, enabling banks to make data-driven decisions and enhance their cross-selling strategies. Ultimately, the integration of personalized recommendations will not only increase customer satisfaction but also can drive potential revenue growth, positioning banking institutions at the forefront of customer-centric financial services.

In addition to the proposed LightFM model, this research project explores alternative recommender systems to compare their performance in generating cross-selling recommendations within ING. The alternative models that will be considered include XGBoost and Decision Tree. These models offer distinct approaches to recommendation generation and will be evaluated alongside the LightFM model to determine their performance in improving cross-selling strategies. By considering multiple models, this master thesis aims to provide a comprehensive analysis of different recommender system approaches and their applicability in ING.

1.1 Research Context and Company Information

The aim of this section is to provide valuable insights into ING's business, data infrastructure, and product offerings at ING. Leveraging this knowledge enables the development of a cross-selling recommender system that is tailored to the company's specific needs and maximizes the chances of successful cross-selling.

ING is a leading global financial institution headquartered in Amsterdam, Netherlands. It operates in over 40 countries and serves more than 38 million customers worldwide. As a modern bank, ING is committed to providing its customers with innovative products and services that are tailored to their needs [2].

The bank has made significant investments to create a strong data infrastructure to support its operations because data is an essential part of ING's business strategy. In order to acquire insights into consumer behavior, market trends, and risk management, ING has built an ambitious data-driven approach to decision-making.

By leveraging its strong data infrastructure and commitment to data-driven decision-making, ING has been at the forefront of driving innovation and improving its operations across various divisions. With its focus on data-driven decision-making, ING's Wholesale Banking (WB) division, including the Transaction Services (TS) department, has harnessed the power of data to drive growth and profitability within the corporate banking landscape. As one of the key pillars of ING's business strategy, the Wholesale Banking division, particularly the TS Sales department, plays an important role in leveraging data insights.

ING's Wholesale Banking division has been serving corporate clients, catering primarily to the largest companies in the market. TS department is an integral part of ING's Wholesale Banking division and operates globally in trade, payments, and cash management, as well as in cash pooling solutions. TS Sales department entails growing the daily banking business and increasing profitability by generating new TS mandates within the Wholesale Banking client portfolio.

TS Sales offers a comprehensive suite of TS solutions across ING network for PCM (Payments & Cash Management) and trade finance activities. This includes all products in the TS domain, such as payments and cash management, liquidity management, supply chain finance, receivables finance, guarantees, and L/C (Letter of Credit), among others. With TS Sales, clients can access a broad range of solutions that cater to their unique requirements, helping them achieve their business

objectives and enhance their overall financial performance.

The TS Sales department of ING's WB offers a comprehensive range of products and solutions. This wide array of products and services enables clients to access tailored solutions that align with their specific needs and contribute to their financial success. Providing these details about the TS department is for understanding the breadth of products and services offered by TS Sales, emphasizing the importance of personalized recommendations to cater to the diverse needs of clients.

To facilitate the integration of data-driven strategies into its operations, ING has set up a dedicated data organization called the Data Analytics Platform (DAP) unit, which is responsible for managing the bank's data assets and developing advanced analytics capabilities. The DAP unit works closely with other business units to identify opportunities for data-driven solutions and develop data projects that support the bank's strategic goals [1].

1.2 Problem Statement

In the rapidly evolving banking industry, data optimization has become a crucial aspect for organizations seeking to achieve strategic objectives, improve business performance, and maximize revenue. However, ING, one of the leading global banks, faces the challenge of harnessing the full potential of its data. With the increasing importance of personalized recommendations in enhancing customer experiences and driving cross-selling opportunities, there is a notable gap in the existing recommender systems at ING.

Today, ING customers often find it difficult to choose the right financial products that match their unique needs and preferences, despite the vast array of options available. This leads to frustration, reduced customer satisfaction, and lower engagement with ING's products, which ultimately might impact the company's revenue. Traditional methods of product recommendations are not effective in providing personalized recommendations to customers. Although banks have implemented recommendation systems, most of them do not offer personalized recommendations, leaving customers struggling with finding the right product.

The TS sales cash advisory team is responsible for providing financial advice to customers regarding products and services. However, despite the wealth of products available, there is currently no system in place for cross-selling additional products to clients within this team. This represents a significant gap in the team's ability.

To address this challenge, this thesis is focused on developing a cross-selling recommender system that helps the TS sales cash advisory team provide recommendations to customers, driving engagement and increasing sales. By bridging this gap, this work is expected to contribute to the team's effectiveness and the overall success of ING.

As an advantage for the TS sales cash advisory team, the implementation of the cross-selling recommender system is expected to significantly enhance the effectiveness of the TS sales Cash Advisory team at ING. With the recommender system in place, the team will no longer need to rely solely on manual analysis and extensive data gathering to identify suitable cross-selling opportunities for customers. Instead, they can leverage the system's capabilities to automatically generate personalized recommendations based on customers' financial profiles and preferences. This automation will free up valuable time and resources previously spent on tedious manual processes. Furthermore, the recommender system's data-driven insights will enable the TS sales Cash Advisory team to gain a deeper understanding of customer behavior and preferences. This knowledge can inform the team's decision-making processes, allowing them to proactively anticipate customer needs and recommend appropriate financial products or services. As a result, the team will be able to deliver more timely and relevant recommendations, leading to enhanced customer engagement and ultimately, increased revenue for ING.

This system will take into account not only the user's interactions with products but also incorporate metadata (side information) that reflects each user's unique features. By adopting this approach, each customer will be provided with relevant recommendations that match their profile, thus increasing their satisfaction and engagement with ING's products. This personalized recommender system will not only benefit new customers but also existing ones who are searching for additional products that align with their preferences.

1.3 Research Scope

The scope of this thesis is to develop a cross-selling recommender system using the LightFM model and evaluate its performance through experiments.

Regarding the recommender system, a machine learning model is a mathematical representation or algorithm that is trained on historical data to make predictions or decisions. It is a key component of a recommender system, as it learns patterns and relationships from the data to generate personalized recommendations.

A machine learning model is needed to make accurate and relevant recommendations to users. A model serves as the foundation of the recommender system, enabling it to analyze data, identify patterns, and generate personalized recommendations.

This thesis includes empirical work which involves training the LightFM model on randomly sampled user-item interactions and on randomly selected subsets of rows from the dataset. The area under the curve (AUC) metric (explained further in section 5.1.1), an evaluation statistic for recommender systems, is used to evaluate the model's performance [17][28].

In order to establish a reference point for performance evaluation, a baseline model is created to set a simple and fair performance benchmark. The purpose of this baseline model is to provide a straightforward prediction of item ratings based on binary preferences. This simple approach allows for a comparison with more sophisticated and complex models, enabling a comprehensive assessment of their effectiveness.

The scope further extends to comparing the performance of the LightFM model with two alternative models namely XGBoost and Sklearn Decision Tree, to assess their performance in building a recommendation system for ING.

1.4 Research Design

In this section, various key aspects of the study are discussed, including the stakeholders and goals, research objectives, research questions, and the methodology employed to achieve the research objectives.

- Firstly, the stakeholders and goals of the study are identified. This involves identifying the relevant parties or individuals who have a vested interest in the research outcomes.

- Next, the research objectives are outlined. These objectives serve as the guiding principles for the study, delineating the specific outcomes that the research aims to achieve. Each objective is designed to contribute to the overall research goal of optimizing data utilization and improving business performance at ING Bank.

- The research questions play a pivotal role in shaping this master thesis. These questions are formulated to address specific areas of inquiry and provide a framework for data collection, analysis, and interpretation. By answering these research questions, the study aims to provide valuable insights and recommen-

dations to support ING.

- Finally, the methodology used to reach the research objectives is discussed. The research methodology provides a structured approach to conducting research that focuses on creating innovative solutions to practical problems.

1.4.1 Stakeholders and Goals

Identifying the appropriate stakeholders is crucial in developing a project's objectives and constraints, which serve as the source of requirements. In the domain of the recommender system, stakeholders can be individuals, teams, or institutions affected by the problem being addressed. For the purpose of our stakeholder identification, we will use the taxonomy of Ian Alexander [4].

In the developed model, the main stakeholders are the business project consultants and project owners who seek to enhance their processes. Functional beneficiaries are the organizations that apply the recommender system and benefit from its results, including normal operators who directly interact with it and whose goals need to be validated.

Other stakeholders involved in the model's development include the University of Twente, which supplies knowledge, and ING, which sponsors the research. The author is the researcher and developer of the model.

Table 1 summarizes the stakeholders involved in the development of the recommender system can be classified and their goals identified based on Alexander's Taxonomy.

Stakeholder	Alexander's Taxonomy	Goals
University of Twente	Knowledge Supplier	Contribute to scientific community and practice
Project Owners (TS Sales Cash Advisory)	Functional beneficiaries	Improve INGA operations through application of business impact capabilities
INGA	Sponsor	Develop tools and capabilities to assist clients in effectively managing and reaping the benefits of data assets
Author	Researcher, Developer	Research and develop a framework that fulfills the requirements and objective of the main stakeholders

TABLE 1: *Possible stakeholders of the framework, based on Alexander's Taxonomy*

- **University of Twente (Knowledge Supplier Stakeholder)**
  Classification: External Stakeholder
  Goals: As a knowledge supplier, the University of Twente contributes to the development of the recommender system by providing research expertise, guidance, and academic support. Their goals include advancing the field of recommender systems, contributing to scientific knowledge, and promoting collaboration between academia and industry.

- **Project Owners (TS Sales Cash Advisory) (Sponsor Stakeholders)**
  Classification: Primary Stakeholders
  Goals: Their main objective is to enhance their business processes and decision-making capabilities through the implementation of the recommender system. They aim to improve efficiency, optimize resource allocation, and increase customer satisfaction by providing recommendations.

- **INGA (Sponsor Stakeholder)**
  Classification: Primary Stakeholders
  Goals: INGA, as the sponsor of the research, has a vested interest in the successful development and implementation of the recommender system. Their goals may involve leveraging the system's capabilities to enhance their own business operations, gain competitive advantage, or improve the customer experience within their banking or financial services domain.

- **Author (Researcher and Developer)**
  Classification: Internal Stakeholder
  Goals: The author of the thesis and developer of the recommender system is the main researcher responsible for the design, implementation, and evaluation of the system. Their goals include conducting a comprehensive study, contributing to the research field, and developing a successful recommender system that meets the requirements and expectations of the stakeholders.

  In summary, the stakeholders involved in the development of the recommender system include business project consultants project owners, beneficiary organizations, normal operators, the University of Twente, INGA, and the author. Each stakeholder group has distinct goals, ranging from improving business processes and decision-making to leveraging accurate recommendations, advancing knowledge, and achieving research objectives. Understanding the stakeholders and their goals is for defining project objectives, aligning requirements, and ensuring the successful development and deployment of the

recommender system.

1.4.2 Research Objectives

The overall goal of this thesis is to develop and evaluate a cross-selling recommender system specifically tailored to ING Bank. To achieve this goal, the thesis sets out the following specific objectives:

- **Investigate various recommender system approaches (explained further in section 3.1.1) and determine the most suitable one for ING's customers.**

  This involves exploring different techniques that are commonly used in recommender systems. By analyzing the strengths and limitations of each approach, the thesis seeks to identify the most suitable recommender system approach for ING's customers. This investigation will provide insights into the best practices and strategies that can be applied to create an effective and personalized cross-selling recommender system.

- **To identify an appropriate learning model that can be applied to this research.**

  This entails selecting a machine learning model that can analyze and process the available data to generate recommendations.

- **To identify data that can be used to train and test the model in this research.**

  This involves gathering relevant data from ING. The data will be preprocessed and prepared to be fed into the learning model for training and evaluation.

- **Evaluate the performance of a developed recommender system and compare it with other models.**

  By conducting a comparative analysis, the thesis seeks to determine the strengths and weaknesses of the developed model and provide insights into its performance compared to other recommender systems.

1.4.3 Research Questions

The section presents the research questions and the sub-research questions for the thesis. The motivation for each sub-question is provided as well. Based on the aforementioned problem statement and research objectives, the research questions are formulated as follows:

RQ 1: **What are the specific requirements and considerations for developing recommender systems tailored to the banking domain, according to published literature?**

The motivation for this research question arises from the need to understand the requirements and considerations involved in designing and implementing recommender systems specifically for the banking industry. While recommender systems have been widely studied in various domains, the banking sector has its own distinct characteristics, regulations, and customer preferences that must be taken into account.

By exploring the published literature, this research question aims to understand these factors can provide valuable insights for researchers and practitioners in the field, enabling them to design more effective and tailored recommender systems that meet the needs and challenges of the banking industry.

RQ 2: **In what way can ING optimize its use of data to achieve their strategic objectives and improve business performance and possibly lock in untapped revenue potential?**

The motivation for proposing research question 2 stems from the increasing importance of data-driven decision-making in ING. As a global financial institution, ING recognizes the potential of leveraging data to drive strategic initiatives, enhance operational performance, and ultimately achieve its business objectives. By optimizing the use of data, ING can gain valuable insights into customer behavior, preferences, and needs, enabling them to tailor their products and services more effectively. Furthermore, leveraging data analytics and advanced technologies can provide ING with a competitive edge by enabling them to identify new revenue opportunities and deliver personalized customer experiences. Therefore, research question 2 serves as a guiding principle to explore the potential of data optimization in empowering ING to make informed decisions, improve business performance, and maximize revenue generation.

- ***SQ 1:*** **Which recommendation algorithms are suitable for making recommendations on our case study?**

  The motivation for proposing Sub-Question 1 arises from the need to identify the most appropriate recommendation algorithms to address the specific requirements and challenges of ING's case study. Recommender systems play a significant role in improving cross-selling practices and maximizing revenue growth in the banking sector. However, not all recommendation algorithms are equally suitable for every context. By exploring and evaluating different recommendation algorithms, we can determine which ones are most relevant in the context of ING's case study. This knowledge will enable us to make informed decisions regarding algorithm selection, implementation, and customization, ensuring that the recommender system aligns with ING's strategic objectives and delivers accurate and personalized recommendations to customers.

- ***SQ 2:*** **How is the performance of our chosen model in building a recommendation system for the case study?**

  The motivation for proposing Sub-Question 2 stems from the need to assess the performance of a selected recommendation system in addressing the specific requirements and objectives of the case study. Building a recommendation system involves implementing and evaluating various models to determine their performance in generating accurate and relevant recommendations. By examining the performance of a chosen model, we can understand its strengths, weaknesses, and limitations in meeting ING's strategic objectives. The findings from this analysis will guide further improvements and refinements to the recommendation system, ensuring that it remains aligned with the needs of ING and delivers optimal results for the case study.

- ***SQ 3:*** **What is the impact of the diverse features in ING's payment data on the performance of our chosen model?**

  The motivation for proposing Sub-Question 3 arises from the recognition that the performance of a recommendation system is influenced by the characteristics and attributes of the data used. In the case of ING's payment data, there are various features available that can provide valuable insights into customer behavior and product relevance. By investigating the impact of diverse features on the performance of a chosen model, we can gain a deeper understanding of how different aspects of payment data contribute to the performance of the

recommendations generated. This analysis helps in identifying the key factors that significantly influence the model's performance and enables us to optimize the utilization of these features for better recommendation outcomes.

Selecting ING's payment data as a dataset in this thesis is for several reasons.

- By analyzing payment data, we can gain a deeper understanding of customers' financial needs and identify patterns that can be used to enhance our recommendation system.
- Second, the use of payment data aligns with research question 2 (RQ2) of optimizing ING's use of data to achieve strategic objectives and improve business performance. Payments are a fundamental aspect of banking operations.

- ***SQ 4:*** **How does our chosen model compare with two selected alternative models for building a recommendation system?**

  The motivation for proposing Sub-Question 4 stems from the need to evaluate the performance of our chosen model in comparison to other approaches. While the chosen model has been selected based on its suitability for ING's case study, it is important to assess its performance relative to alternative models to gain a comprehensive understanding of its strengths and weaknesses. By comparing our chosen model with two alternative models, namely XGBoost and Sklearn Decision, we can examine how each model handles ING's recommender system. By comparing multiple models, we can make informed decisions about the best approach to building a recommendation system for ING, aligning with the objective of optimizing data usage to achieve strategic objectives.

1.4.4 Research Methodology

To answer the research questions, a methodological approach is required as the foundation. For this research, Design Science Research Methodology (DSRM) [24] based on the thesis of Sajid [27], a methodology in the field of Information Systems (IS) to design IS artifacts has been adopted. Adopting such a method using the scope of any research would be advantageous as it provides a road map for researchers to use design as a research mechanism. DSRM is a 6-step process model aligned in a nominal sequence. It is structured as follows:

1. **Problem Identification and Motivation:** To define the specific research

problem and justify the value of a solution.

2. **Define the objective for a solution:** To define objectives of the solution based on the problem identified and understanding the feasibility of seeing it through.

3. **Design and Development:** To create an artifact i.e. constructs, models, methods, or instantiations.

4. **Demonstration:** To demonstrate the use of the created artifact instances of the problem. This could be in the form of experiments, simulations, case studies, and more.

5. **Evaluation:** To observe and measure how well the artifact supports a solution to the problem using relevant metrics and analysis techniques.

6. **Communication:** To communicate the problem and its importance, the artifact, its utility and novelty, and the rigor of its design to researchers and other relevant audiences.

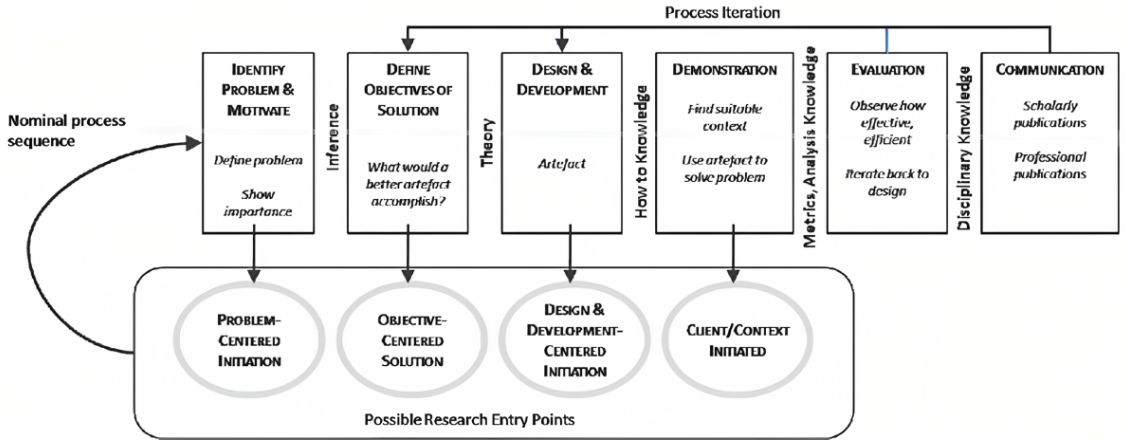


FIGURE 1: *Design Science Research Methodology [24]*

The reason behind selecting DSRM is that it aligns with the parameters of the research objective, which is to develop a technique (artifact). The process model additionally contains an actual implementation of the created artifact, which will be followed by an assessment of its efficiency. This makes it easier to hold the researcher liable for the generated artifact and how well it functions in actual use. The cycle, indicated by arrows, that moves from the evaluation and communication phases back to the phase of defining objectives is another finding that validates the process model.

The researcher initially encountered the challenge of comprehending the purpose behind the "Possible Research Points" box. Through an in-depth review of the relevant literature, it became evident that the process model does not mandate a specific starting point. Instead, researchers are granted the freedom to commence their investigation from any of the available entry points. Pfeffer et al. [24] argue that the process model is designed to accommodate the existence of an artifact that has not yet been formally considered as a solution within the targeted problem area. This methodology provides researchers with flexibility and choice, enabling them to select from various entry points rather than being constrained to a single one. For this research, the author has chosen to adopt a design and development-centered approach as the starting point. This decision aligns with the objective of developing a new technique by studying the existing approach, identifying its limitations and gaps, and addressing them.

1.5 Thesis Outline

This thesis includes six chapters.

Chapter 1 states the research objectives and the research questions and sub-questions. The chapter sets the stage for the subsequent chapters by establishing the context and significance of the research.

Chapter 2 explains a methodology used for conducting the systematic literature review and presents the study selection process and data analysis and aims to provide a comprehensive understanding of relevant research.

Chapter 3 provides an in-depth exploration of the foundational concepts and theoretical background necessary for designing our recommender system. This chapter provides a comprehensive exploration of the key theories, models, and methodologies relevant to the field, equipping the reader with the necessary theoretical understanding.

Chapter 4 explains the design of the chosen model and implementation process and details and the data acquisition process.

Chapter 5 explains the evaluation methodology used to assess the performance of your cross-selling recommender system, presents the results of experiments, and figures out alternative models.

Chapter 6 discusses the conclusions by answering the (sub)research questions, limitations, and recommendations for future research and contributions to practitioners. An overview of the chapter and the related (sub)research questions can be seen in Table 2.

Chapter		(Sub)Research Question
2	Literature Review	RQ 1
3	Foundations for the development of the cross-selling recommender system	SQ1
4	Design and Implementation	RQ2, SQ1, SQ2
5	Evaluation	SQ1, SQ2, SQ3, SQ4
6	Conclusion	RQ 1, RQ 2, SQ1, SQ2, SQ3, SQ4

TABLE 2: *Outline of the Thesis Structure*

2 Literature Review

This chapter presents the systematic literature review (SLR) conducted as part of the thesis, employing the systematic literature review approach. The goal of this literature review is to systematically examine and summarize the existing body of published literature on a specific topic or research question. It aims to provide a comprehensive understanding of the specific requirements, considerations, and approaches for developing recommender systems tailored to the banking sector. The review encompasses papers that discuss new techniques or build upon existing approaches in the field of business impact analytics related to recommender systems in banking. Its purpose is to gather, evaluate, and synthesize relevant information from various sources such as academic journals, books, conference proceedings, and other scholarly publications.

It outlines the methodology employed for conducting the review, including the search strategy, databases utilized, and the data extraction strategy implemented. Furthermore, the chapter provides an extensive overview of the relevant literature in order to establish a comprehensive understanding of recommender systems in the banking industry, as well as the specific recommender algorithms employed within ING.

Table 3 shows the literature review protocol used for this research as adapted from the systematic literature review procedure proposed in a book titled "Guidelines for performing Systematic Literature Reviews in Software Engineering" by Kitchenham and Charters [18].

For this specific review, three prominent digital libraries were chosen: Google Scholar[1], IEEE Explore [2], and Scopus [3] as the digital libraries. These databases were selected for their extensive coverage of scholarly literature in the field. To construct an effective search string, a range of keyword variations and synonyms were evaluated, drawing from relevant terms and concepts identified in the existing literature [7].

2.1 Search Strategy

In this literature review, a search strategy was implemented to gather relevant literature from databases accessible through the University of Twente. Among the

[1] https://scholar.google.com
[2] https://ieeexplore.ieee.org
[3] https://www.scopus.com

Protocol Component	Protocol Sub-component	Component Description	Component Details
Search strategy	Search terms	Includes synonyms, abbreviations, alternative spellings, terms from reference list, etc. Search terms are finalized through trial-and-error by running test search queries with potential search terms.	• Data Analytics Platform (DAP) • model generation, model creation, content creation • Hybrid Recommender system, implementation, design methodology, development process, development workflow • LightFM, Recommender System
	Search queries	Constructed from search terms using ADD, OR, NOT operator.	Each search query is mapped to the relevant research question(s) the query is intended to provide insights for.
	Resources	Databases/ resources where the search queries are executed.	IEEExplore: https://ieeexplore-ieee-org.ezproxy2.utwente.nl/search/advanced Google scholar: https://scholar.google.com/ Google search for looking up industry literature: https://www.google.com/
Study selection criteria	Inclusion criteria	Query results which meet the inclusion criteria proceed to the study quality assessment phase.	• Open access paper
	Exclusion criteria	Query results which meet the exclusion criteria do not proceed to the study quality assessment phase.	• Non-English papers. • Duplicate papers. • Papers which involve the use of Recommender System (RS), LightFM.
Study quality assessment	Quality checklist (i.e., more detailed inclusion/ exclusion criteria)	Query results which satisfy the quality assessment are examined in detail using the data extraction form.	Specific quality checklists for different search queries.
Data extraction strategy	Data extraction form	Used to systematically extract key information from the final list of papers filtered from the initial query results.	The data extraction form outlines attributes to be extracted each paper.

TABLE 3: *Literature Review Protocol*

available databases, IEEE Xplore, Scopus, and Google Scholar were identified as the most pertinent for the study. After selecting the appropriate databases and digital libraries, the next step involved constructing a search string using relevant keywords. To ensure comprehensive and accurate results, multiple combinations of keywords and synonyms related to the thesis scope were experimented with to formulate an effective search string. The aim was to capture a wide range of relevant literature while maintaining relevance and precision. The queries are as follows:

("Recommender System" OR "Hybrid Recommender System " OR "Machine Learning-Based Recommender System") AND ("Cross-Selling" OR "Up-Selling")

("Collaborative Filtering") AND ("Content-Based Filtering")

("LightFM Recommender" OR "LightFM") AND ("Decision Tree Recommender" OR "Decision Tree") AND "XGBoost" OR "XGBoost Recommender")

("Feature Engineering" OR "Feature Selection")

The purpose of this chapter is to delve into the existing literature and gather insights related to the research questions (RQ1, RQ2), as well as the specific sub-questions (SQ1, SQ2, SQ3, SQ4) outlined in section 1.4.3. By conducting a comprehensive review of relevant studies, this chapter aims to provide a comprehensive understanding of the existing knowledge and findings related to the identified research questions.

The following search query has been used in the chosen scientific databases mentioned in Table 4 for searching relevant studies.

2.2 Inclusion and exclusion criteria

The initial selection of studies was based on their titles. The following criteria were utilized to determine which studies were included (ICs) and excluded (ECs).

The inclusion criteria for selecting papers in our review are as follows:

- **IC1:** The paper must be directly related to the subject of our review, including papers that propose new techniques or build upon existing approaches in the field of business impact analytics. We also consider articles that evaluate the current approaches through comparison studies, case studies, and experiments.

Search Query	Databases	Research Question
("Recommender System" OR "Hybrid Recommender System " OR "Machine Learning-Based Recommender System") AND ("cross-selling" OR "up-selling")	• IEEExplore • Scopus • Google Scholar	RQ1 RQ2
("Collaborative Filtering") AND ("Content-Based Filtering")	• IEEExplore • Scopus • Google Scholar	SQ1
("LightFM XGBoost" OR "LightFM") AND ("Decision Tree Recommender" OR "Decision Tree") AND "XGBoost" OR "XGBoost Recommender")	• IEEExplore • Scopus • Google Scholar	SQ2 SQ4
(“Feature Engineering” OR “Feature Selection”)	• IEEExplore • Scopus • Google Scholar	SQ3

TABLE 4: *Search Queries mapped to Research Questions*

- **IC2:** The publication year of the paper must be between 2012 and 2022.
- **IC3:** The paper must be published in a peer-reviewed conference or journal.
- **IC4:** The paper must be written in English.
- **IC5:** The paper must be available for download.

The following exclusion criteria were also applied:

- **EC1:** Articles with the same title or identical content will be excluded and keep only one of them after prioritizing the most relevant or comprehensive version of the papers.
- **EC2:** Articles that are not relevant to the research questions will be excluded.
- **EC3:** Articles that are too brief or incomplete will also be excluded.

2.2.1 Study selection

The study selection process for the literature reviews in this thesis follows the three stages proposed by Kitchenham [18]: Planning, Conducting, and Reporting. The study selection process consists of the following steps:

- Planning Stage:
  - Clearly define the (sub)research questions and objectives of the literature reviews, including the specific aspects related to recommender systems in

the banking sector.

  - Identify the inclusion and exclusion criteria based on the research questions, such as the publication year, relevance to the banking industry, and focus on recommender systems.

  - Determine the sources for literature search, such as academic databases such as Google Scholar, IEEE Xplore, and Scopus.

  - Consider any specific keywords or search terms related to recommender systems, data optimization, and the banking industry to guide the search process.

- Conducting Stage:

  - Perform a comprehensive search using the identified sources and search terms to retrieve relevant literature.

  - Apply the inclusion and exclusion criteria to the retrieved studies based on title, abstract, and full-text screening.

  - Assess the relevance of each study based on its alignment with the (sub)research questions and objectives, as well as its contribution to the understanding of recommender systems.

  - Document the selection process, including the number of studies identified, screened, and included/excluded, along with reasons for exclusions.

- Reporting Stage:

  - Provide a clear description of the study selection process in the thesis, including details on the research questions, inclusion and exclusion criteria, and sources searched.

  - Present a flow diagram or table illustrating the study selection process, from initial identification to final inclusion.

  - Include a summary of the characteristics of the included studies, such as publication year, research methodologies, data sources, and key findings, to provide an overview of the literature landscape.

By following this systematic approach to study selection, the literature reviews aim

to ensure comprehensive and unbiased coverage of relevant studies that contribute to the understanding of recommender systems in the banking sector, specifically addressing ING's objectives.

According to the systematic literature review, Figure 2 represents the flow chart outlining the selection process for the identified articles and the criteria employed to determine the inclusion of primary studies in the review. The flow chart provides a visual representation of the number of records at each phase of the selection process, offering insights into the progression and refinement of the article selection.

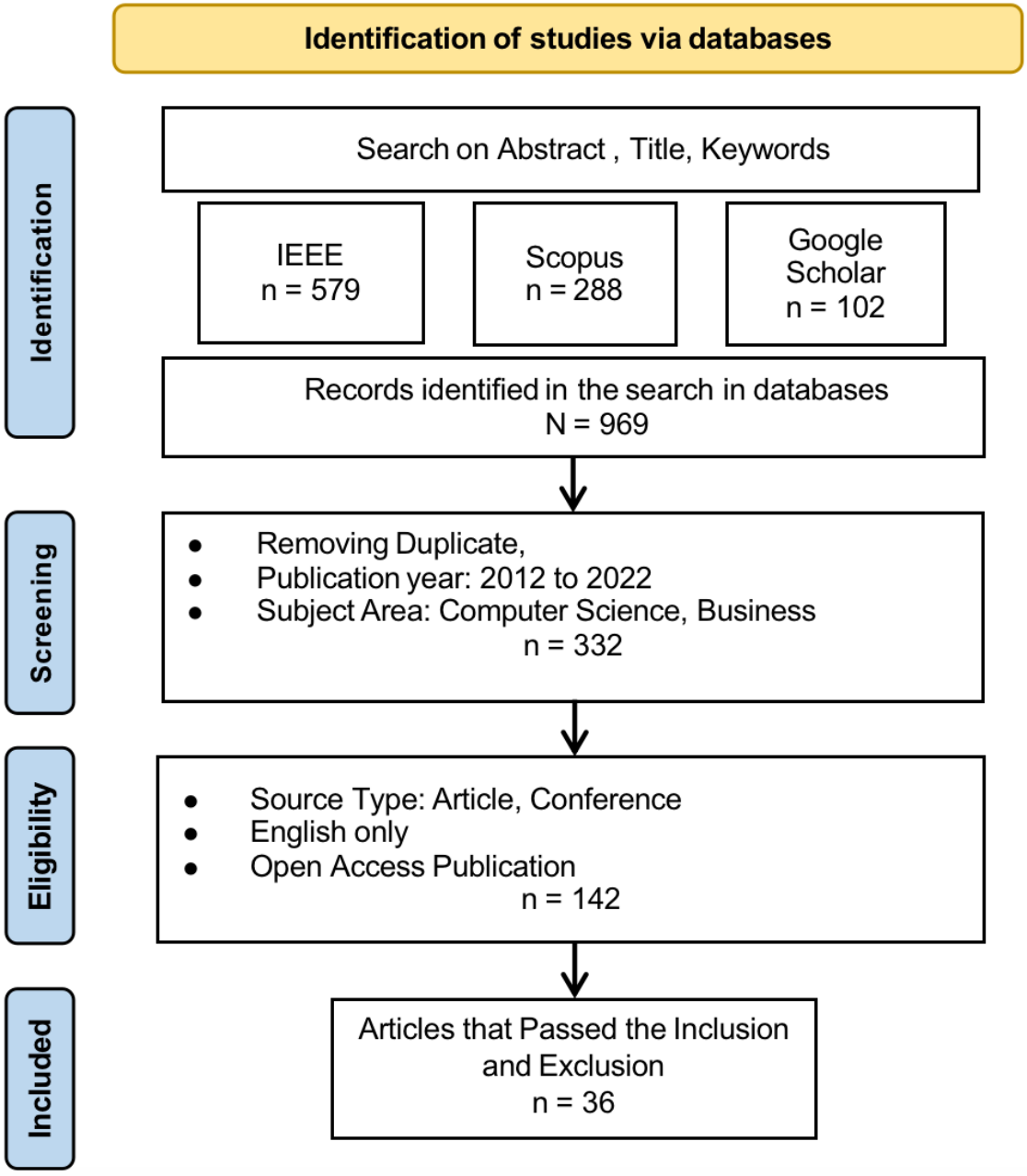


FIGURE 2: *Study Selection Process*

Data Collection and Analysis

The study selection process involves multiple stages that are carefully conducted to ensure the inclusion of relevant studies. Prior to initiating the selection process, clear study selection criteria are defined to identify studies that provide evidence pertaining to the research questions. These criteria, known as inclusion and exclusion criteria, serve as guidelines for the selection process (Section 2.2).

The initial round of exclusions is performed by assessing certain properties of the identified studies, such as the publication year. Studies published before 2012 are excluded from the review for two main reasons. Firstly, the focus of this master thesis is primarily on the most recent literature that pertains to the three main constructs under investigation. Contributions made prior to 2012 are likely to be outdated and less relevant. Secondly, considering all publishing years would exceed the allotted time constraints for this research study. Additionally, studies published in languages other than English are also excluded.

The objective is to include studies that are directly relevant to the research questions, providing valuable evidence for addressing them. During the subsequent stage of selection, studies are assessed based on their title and abstract. Since it can be challenging to determine the adequacy of evidence based solely on the abstract and title, a liberal approach is adopted in interpreting relevance during this stage.

In the final stage of the study selection process, the remaining studies are thoroughly read in their entirety. Once again, the inclusion or exclusion of papers is determined based on their relevance to the research questions. Following this stage, the selected studies proceed to undergo a quality assessment.

In addition to the application of inclusion and exclusion criteria, the quality assessment of individual studies is an important step in this thesis. Assessing the quality of a study serves as a guide to interpret the results rather than a strict inclusion/exclusion criterion. The quality assessment process involves a systematic evaluation of various aspects of each study, including its research design, methodology, data collection methods, analysis techniques, and the overall rigor of the study. Each study is carefully examined to determine the level of methodological soundness, potential biases, and robustness of the findings. The quality assessment also involves critically examining the research findings and conclusions presented in the selected studies. This includes assessing the logical coherence of the arguments, the consistency of the results, and the validity of the conclusions drawn.

2.3 Finding of Literature Review

Through the extensive literature review conducted in the banking industry, several key findings have emerged for developing recommender systems tailored to this domain.

Yahyapour [36] and Asosheh [5] highlighted the need for recommender systems in the banking industry based on the Technology Acceptance Model, revealing a latent demand for these systems among users. This emphasizes the importance of considering user acceptance and adoption when developing recommender systems in the banking context.

Gallego and Huescas [12], along with Vico and Huescas [13] focused on exploiting contextual information in transactional data to generate personalized recommendations using a clustering-based approach. Their work demonstrated the value of leveraging customer data, such as credit card information, to cluster customers and provide targeted recommendations. This underscores the significance of utilizing relevant and context-specific data sources in recommender system development for the banking industry.

Felfernig advocated for knowledge-based recommender systems in the banking domain due to their flexibility in handling multi-criteria-based financial decisions. This approach offers the ability to incorporate various banking products and constraints, highlighting the importance of customization and adaptability in recommender system design for the banking sector [10].

Zhao explored the use of demographic data in generating personalized recommendations by inferring users' purchase intentions from tweets and online reviews. While this approach demonstrated the value of demographic information, it is more suited for e-commerce rather than banking, indicating the need to consider domain-specific data sources and characteristics in recommender system development [37].

Mitra examined the insurance domain and proposed a hybrid recommender system combining collaborative and content-based filtering. However, this technique relies on explicit ratings from users, which are generally unavailable in the banking sector. This finding emphasizes the need to explore alternative data sources and recommendation approaches that align with the unique characteristics of the banking industry [23].

These findings from the literature review shed light on the specific requirements and

considerations for developing recommender systems in the banking domain. They emphasize the importance of user acceptance, leveraging contextual and domain-specific data, flexibility in handling multi-criteria decisions, and exploring alternative recommendation approaches. By incorporating these insights into the development of recommender systems, banks like ING can enhance their ability to generate personalized recommendations, optimize operations, improve customer experiences, and achieve strategic objectives effectively.

The literature review revealed several following considerations for developing recommender systems tailored to the banking sector.

- The literature emphasized the need to consider users' attitudes, preferences, and behaviors when designing and implementing these systems. Understanding the factors that influence users, can help developers create recommender systems that meet customers' needs.

- Contextual information emerged as a vital factor for generating personalized recommendations in the banking industry. Studies highlighted the importance of leveraging customer data, such as transaction history, geolocation, and demographic information, to enhance the accuracy and relevance of recommendations. By incorporating contextual factors into the recommendation algorithms, recommender systems can better understand customers' preferences and provide tailored suggestions that align with their financial goals and requirements.

- Flexibility and adaptability were identified as key considerations for recommender systems in the banking domain. The literature emphasized the need for these systems to handle multi-criteria-based financial decisions and accommodate various banking products and constraints. Knowledge-based approaches were highlighted as effective in this regard, as they allow for customization and can incorporate complex decision-making processes specific to the banking industry.

- Domain-specific characteristics of the banking sector also need to be taken into account when developing recommender systems. The banking industry often lacks explicit ratings from users, making collaborative filtering less applicable. Instead, alternative approaches such as content-based filtering, demographic-based inference, or hybrid models may be more suitable. Considering the data sources and characteristics of the banking domain ensures the development

of recommender systems that are aligned with the industry's specific requirements.

2.4 Summary of Literature Review

Examining studies conducted in the banking sector provides valuable insights into the technologies and techniques employed in generating personalized recommendations for clients. By reviewing these studies, we gain a better understanding of the benefits and challenges associated with implementing recommender systems in the banking sector. This knowledge can inform ING in leveraging data to optimize its operations, improve customer experiences, and achieve its strategic objectives.

The studies developed models that focused on exploiting the value of contextual information in transactional data to generate personalized recommendations using a clustering-based method.

The findings from these studies offer insights into the potential impact of recommender systems on business performance and revenue generation. This information can assist ING in exploring new opportunities for utilizing data effectively in its operations. By incorporating the knowledge gained from these studies, ING can enhance its ability to provide personalized recommendations to customers, optimize its services, and ultimately deliver superior customer experiences in the banking industry.

The outcomes of the literature review and analysis can be summarized as follows:

- **Knowledge Gathering:** The literature review allows us to gather knowledge about different technologies and techniques employed in generating personalized recommendations for clients in the banking industry. By reviewing these studies, we can identify best practices and understand the state-of-the-art approaches in this field.

- **Strategic Guidance:** The insights gained from the literature review can provide guidance for ING's strategy in leveraging data to optimize operations, improve customer experiences, and achieve strategic objectives. By understanding the benefits and challenges associated with implementing recommender systems, we can make informed decisions regarding the utilization of data in banking operations.

- **Revenue Generation:** The findings from the reviewed studies can offer valuable insights into the potential impact of recommender systems on business performance and revenue generation in the banking sector. The obtained in-

formation can help ING explore new opportunities for utilizing data effectively and driving financial outcomes.

- **Relevance and Applicability:** The reviewed studies are specifically focused on recommender systems in the banking industry, making them directly applicable to our thesis. By analyzing these studies, we can gain a deep understanding of the specific context, challenges, and opportunities related to personalized recommendations in banking.

- **Identification of Appropriate Approaches:** The literature review highlights various approaches used in recommender systems. Understanding the advantages and limitations of these approaches helps us select the most suitable methods for our thesis in the banking sector.

3 Foundations for the Development of the Cross-Selling Recommender System

In this chapter, we delve into the theoretical aspects of developing a recommender system for cross-selling financial products in the banking sector. This section focuses on the design choices and technical background to create a personalized recommendation system tailored to the needs of ING. By exploring the theoretical aspects of recommender systems, we aim to shed light on the underlying principles and methodologies that inform the proposed solution.

By delving into the theoretical aspects of recommender systems, we aim to equip the reader with a comprehensive understanding of the conceptual framework that underlies the proposed solution. This chapter serves as a bridge between theory and practice, providing a solid foundation for the subsequent chapters that delve into the practical implementation and evaluation of the recommender system. This chapter helps to demonstrate the thought process and rationale behind the proposed solution.

3.1 Recommendation Tools and Technologies

Recommender systems are a tool for predicting the utility or usefulness of items to a particular user. There are various types of recommender systems have been developed to address this requirement. According to Burke (2007) and Ricci, Rokach, and Shapira (2011), recommender systems are crucial for predicting the usefulness of items for a specific user. To meet this requirement, various types of recommender systems have been developed. The taxonomy of recommender systems, as outlined by Burke (2007) and Ricci et al. (2011), consists of Collaborative Filtering (CF), Content-based (CB), and Hybrid recommender systems [9][30]. The algorithmic models for recommendations are explained in detail, respectively, in Sections 3.1.2, 3.1.3, and 3.1.4.

3.1.1 Recommender System Architecture

To provide an overview of the recommendation system, Figure 3 presents a comprehensive summary of their approaches. This visual representation serves as a handy reference guide, allowing readers to quickly grasp the underlying principles and technologies employed by each model. It includes Collaborative Filtering, Content-Based Filtering, and Hybrid systems. This condensed overview provides a quick reference

to understand the underlying principles and technologies of these models [19].

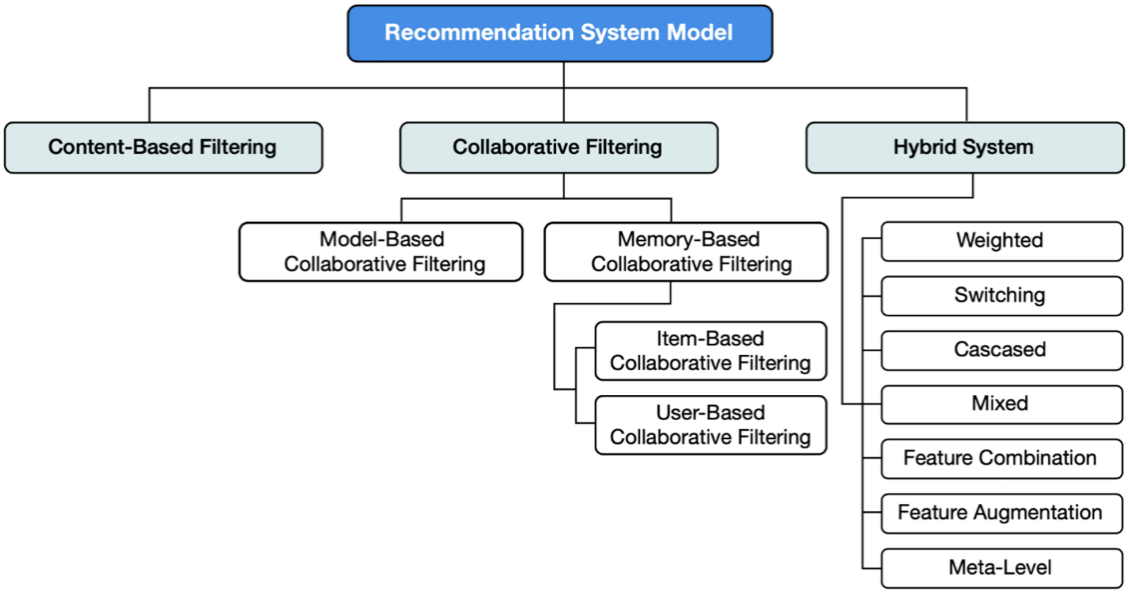


FIGURE 3: *Overview of recommendation approaches [19]*

3.1.2 Collaborative Filtering

CF systems recommend items to a user based on the ratings of other users with similar interests. This approach only requires past information about the rating profiles of different users and can be either memory-based or model-based. "Ratings" refer to the scores or feedback given by users to items they have interacted with. Ratings could be explicit, where users directly provide numerical scores or rankings for items, or implicit, where user behavior such as clicks, purchases, or time spent on an item is used as an indicator of preference. Collaborative filtering systems use these ratings to identify users with similar preferences and recommend items that other similar users have rated highly [22][31].

The model can be divided into two types: Memory-Based Collaborative Filtering and Model-Based Collaborative Filtering.

User-Based Collaborative Filtering and Item-Based Collaborative Filtering are two further subcategories of Memory-Based Collaborative Filtering.

1. **Memory-Based Collaborative Filtering**
   Memory-based collaborative filtering relies solely on the user-item interaction matrix to generate personalized recommendations for users. The recommendation process hinges entirely on a user's past behavior, including ratings and

interactions with items. It includes User-Based Collaborative Filtering and Item-Based Collaborative Filtering.

- **User-Based Collaborative:** User-Based Collaborative Filtering (Figure 4) compares the evaluation data of users on the same item and generates a list of the top N items that suit the user's taste based on the rating of similar users.

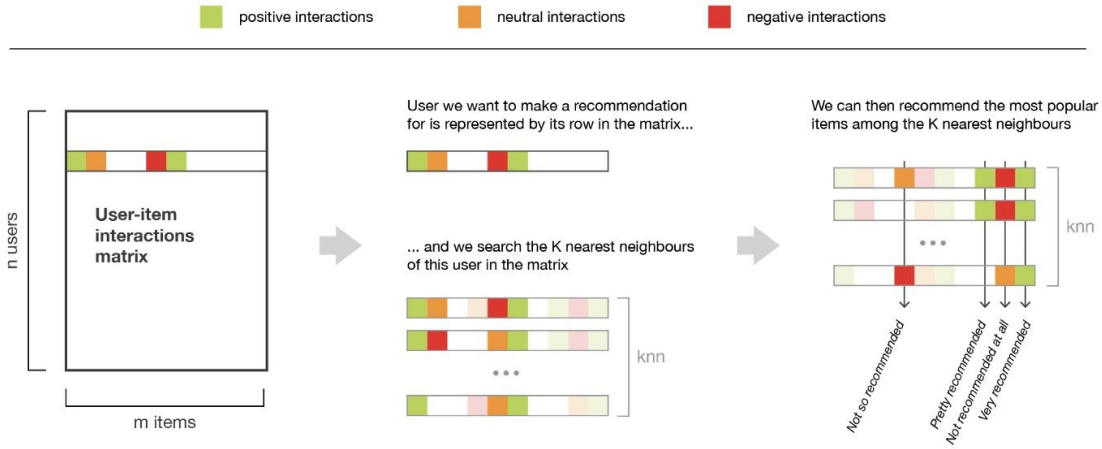


FIGURE 4: *User-Based Collaborative Filtering [26]*

- **Item-Based Collaborative:** Item-Based Collaborative Filtering (Figure 5) predicts an item by creating a rating matrix of users and items and using the similarity between the item and the item selected by the user.

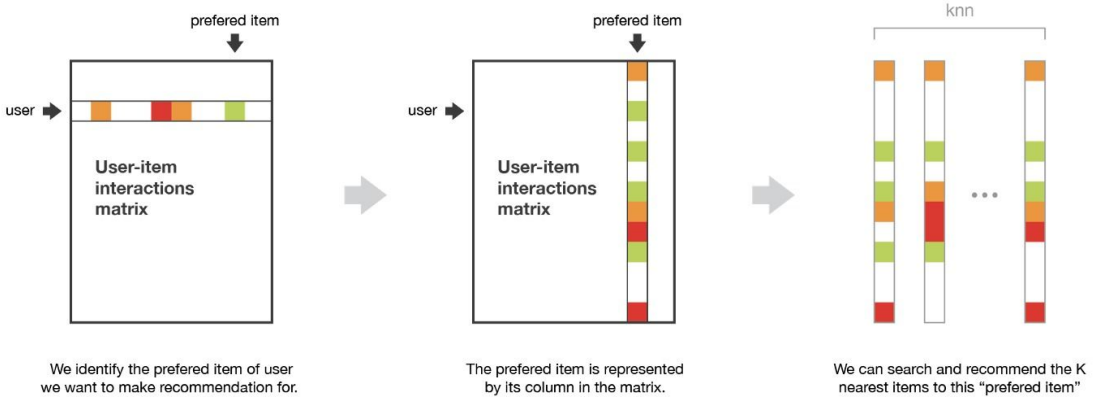


FIGURE 5: *Item-Based Collaborative Filtering [26]*

Memory-Based Collaborative Filtering utilizes various technologies such as

Pearson Correlation, Vector Cosine Correlation, and KNN to create similar groups (neighborhood groups) among users and recommend items to users within the same group.

2. **Model-Based Collaborative Filtering**
   In the model-based collaborative filtering approach, user interactions with items they have not yet interacted with are predicted and ranked using machine learning models. These models are trained using the existing interaction information from the interaction matrix, using various algorithms such as matrix factorization, deep learning, clustering, and more.

   - **Matrix Factorization** Matrix factorization is one of the most widely used techniques in the model-based approach. It allows for the generation of latent features by decomposing the sparse user-item interaction matrix into two smaller and denser matrices of user and item entities. The core idea of matrix factorization is that there is a lower-dimensional latent space of characteristics in which we may represent both users and items that we can compute the dot product of corresponding dense vectors in that space to determine the interaction between a user and an item.

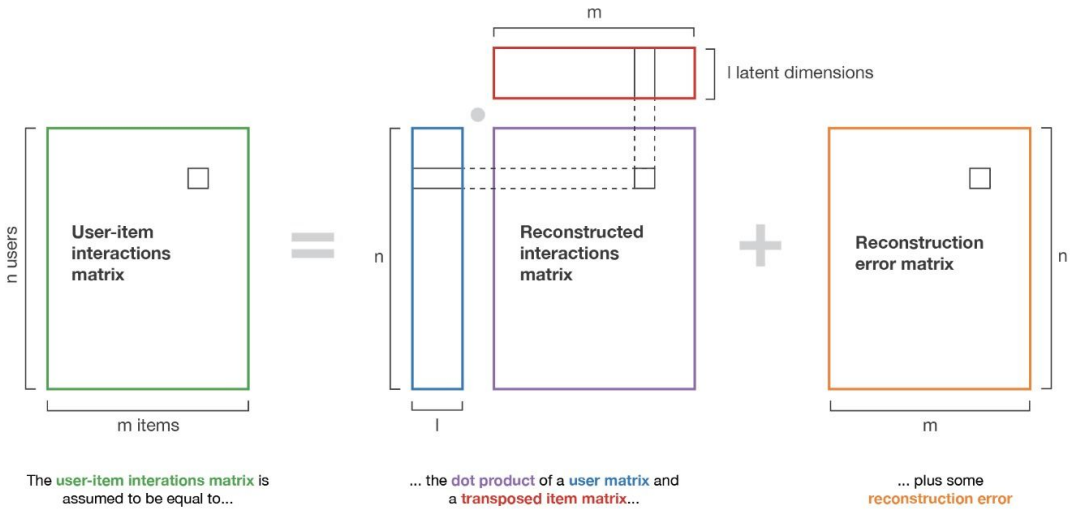


FIGURE 6: *Matrix Factorization [26]*

To better understand this approach, let's take the example of a user-item rating matrix. In order to model the interactions between users and items, we can assume that there are features describing items that can also be used to describe user preferences. However, we do not want

to explicitly give these features to our model. Instead, we let the system discover these useful features by itself and make its own representations of both users and items. As they are learned and not given, these extracted features taken individually have a mathematical meaning but no intuitive interpretation. However, it is not uncommon for structures to emerge from this type of algorithm that is extremely close to the intuitive decomposition that a human could think about. The consequence of such factorization is that close users in terms of preferences, as well as close items in terms of characteristics, end up having close representations in the latent space.

Matrix Factorization Mathematics: To explain matrix factorization, we use a classic iterative method based on gradient descent. This approach allows us to obtain factorizations for large matrices without having to load all the data into memory at once. Suppose we have an interaction matrix M with n rows and m columns representing ratings, where each user has rated only some of the items. Most of the interactions are expressed as "None" to indicate the absence of a rating. The goal is to factorize the matrix into two smaller, dense matrices X and Y^T:

$$M \approx X.Y^T \tag{1}$$

- The user matrix X ($n \times l$) is composed of rows representing the n users.
- The item matrix Y ($m \times l$) is made up of rows representing the m items.

$$user_i \equiv X_i \quad \forall_i \in \{1, ..., n\} \tag{2}$$
$$item_j \equiv Y_j \quad \forall_j \in \{1, ..., m\} \tag{3}$$

The symbol l represents the dimension of the latent space, where users and items will be represented. Our objective is to find matrices X and Y, whose dot product can provide the closest approximation of the existing user-item interactions. Let E be the set of pairs (i, j) where M_{ij} is set (not None). The aim is to minimize the "rating reconstruction error" by finding X and Y.

$$(X, Y) = \arg\min_{X,Y} \sum_{(i,j)\in E} [(X_i)(Y_j)^T - M_{ij}]^2 \tag{4}$$

By adding a regularization factor and dividing the expression by 2, we obtain:

$$(X, Y) = \arg\min_{X,Y} \frac{1}{2} \sum_{(i,j)\in E} [(X_i)(Y_j)^T - M_{ij}]^2 + \frac{\lambda}{2}(\sum_{i,k}(X_{ik})^2 + \sum_{j,k}(Y_{jk})^2) \tag{5}$$

The matrices X and Y can be obtained through a gradient descent optimization process, which has two notable characteristics. Firstly, the gradient does not have to be calculated over all pairs in E at each step, allowing for optimization "by batch". Secondly, the gradient descent can alternate between X and Y at each step, with one matrix fixed and optimized for the other before switching at the next iteration.

After the matrix has been factorized, we can make new recommendations by multiplying a user vector with any item vector to estimate the corresponding rating. Additionally, we can also use user-user and item-item methods with these new representations of users and items. As a result, approximate nearest neighbor searches would no longer have to be performed over large, sparse vectors but over small, dense ones, making some approximation techniques more manageable.

■ CF Advantages and Disadvantages

One advantage of this recommendation system is that it can automatically learn embeddings, without requiring any domain knowledge. Furthermore, the model can also discover new interests among users, making it adaptable to changing user preferences.

However, the model's prediction for a user-item pair is based on the dot product of the corresponding embeddings. A missed item during training would prevent the system from creating an embedding for it, which would prevent the model from offering recommendations for that item. This is commonly referred to as the "cold-

start" [4] problem.

3.1.3 Content-Based Filtering

CB systems utilize the features associated with products and the ratings given by a user to create a user-specific profile and classify items based on product features. In content-based recommender systems, ratings are used to train the system's machine learning model to predict a user's preference for items based on their features.

Content-based methods frame the recommendation problem as a classification or regression task, where the objective is to predict whether a user will "$like$" an item or to predict the rating that a user will give to an item. To achieve this, the model relies on the features of the user and/or item that are available to us, which form the "content" of the "content-based" approach [26].

If we base our classification or regression on item features, the approach is considered user-centered. This means that the modeling, optimization, and computations can be done by the user. In this case, we build and learn one model per user based on item features and figure out the probability that this user will like each item. The model associated with each user is naturally trained on data related to this user, leading to pretty robust models, as many items have been interacted with by the user. However, the interactions considered to learn the model come from various items, and even if these items have similar features, users' preferences can differ.

If we use item features, the method becomes user-centered: modeling, optimization, and computations can be performed by user. We train one model per user based on item features that attempt to find the probability that this user likes each item. We can then associate a model with each user that is trained on their data: the resulting model is more personalized than its item-centered counterpart as it only considers interactions from the particular user.

To train a Bayesian classifier for each item, we aim to input user features and output either "$like$" or "$dislike$". Therefore, in order to accomplish this classification task, we need to compute:

$$\frac{\mathbb{P}_{item}(like|user-features)}{\mathbb{P}_{item}(dislike|user-features)} \tag{6}$$

[4] *A* cold-start problem means that the recommender system cannot make recommendations for a new user with no history[32].

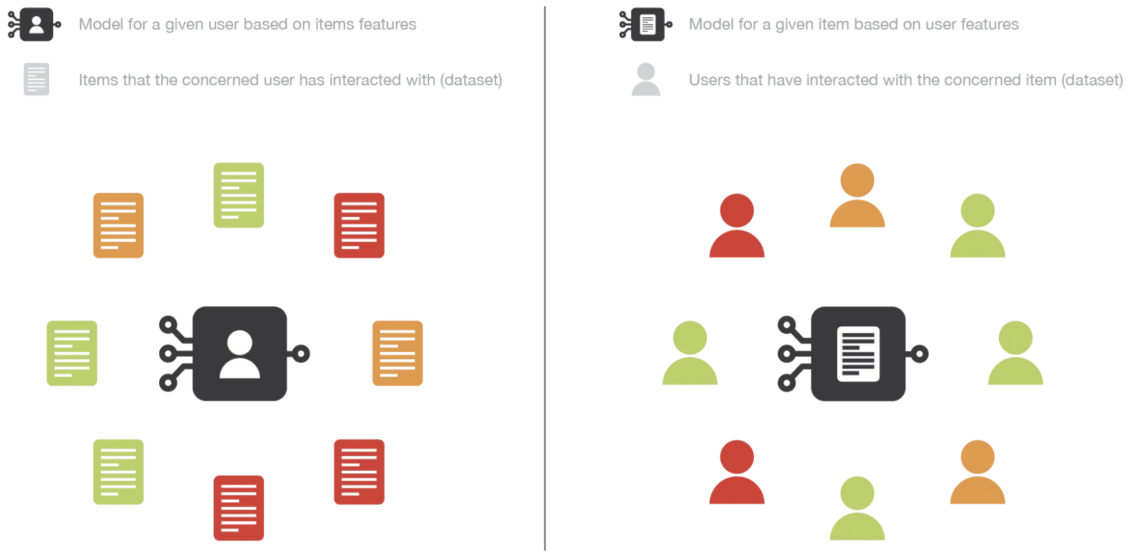


FIGURE 7: *Difference between item-centered and user-centered content based [26]*

■ CB Advantages and Disadvantages

One advantage of the Content-Based is that it can generate personalized recommendations without relying on data from other users. This means that the system can be easily scaled to accommodate a large number of users without the need for additional data.

On the other hand as a disadvantage, it is crucial to note that the recommendation system heavily relies on the domain knowledge of the system. Without a good understanding of the domain, the system may not function effectively, resulting in incorrect recommendations.

3.1.4 Hybrid Recommender

The hybrid recommender system utilized in this master thesis represents a unique approach that integrates both content-based and collaborative filtering techniques, which are previously discussed. By combining the strengths of collaborative filtering and content-based filtering, this hybrid approach improves recommendation performance.

As Figure 8 shows, in a hybrid approach, by leveraging the collaborative filtering method, the system can tap into the collective preferences of a user community,

making accurate recommendations based on similar user behaviors. Simultaneously, content-based filtering allows the system to incorporate item characteristics and user preferences, providing a more comprehensive understanding of user tastes and preferences. Hybrid systems are introduced to take advantage of multiple systems.

There are various ways to combine recommender systems, including Weighted, Switching, and Mixed Hybrid Recommender Systems.

These types of systems consider users' preferences to adjust the system accordingly. By taking into account the users' likes, the system can adapt to their level of preferences, ultimately improving their overall experience.

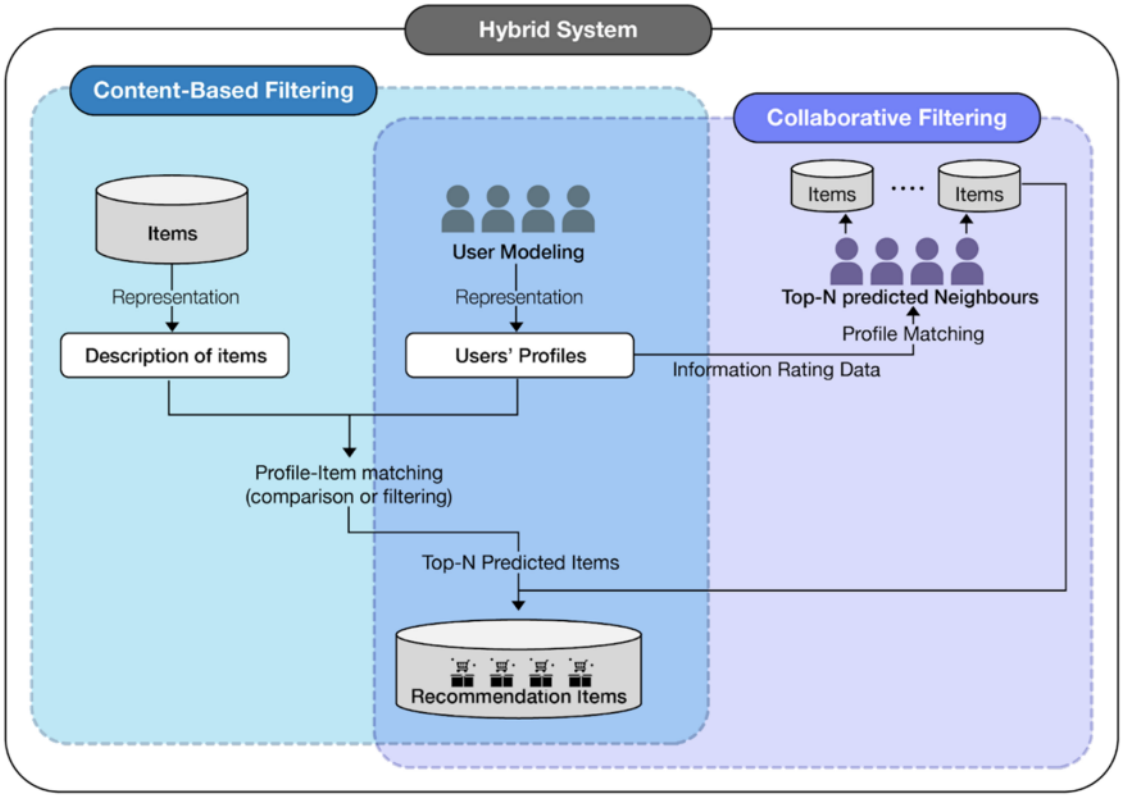


FIGURE 8: *Process of recommendation systems [19]*

Burke [8] presented a taxonomy for the hybrid recommendation systems. He classifies Hybrid Recommendation Systems into the following seven classes:

1. **Weighted:** Scores from different recommendation components are statistically combined using an additive formula.
2. **Switching:** The system selects a particular recommendation component and applies it.

3. **Mixed:** Multiple recommenders provide their recommendations, which are merged and presented as a single rated list.
4. **Feature Combination:** This class has two different recommendation components: contributing and actual. The functioning of the actual recommender depends on the data modified by the contributing one, which throws features of one source onto the other component's source.
5. **Feature Augmentation:** This class is similar to the feature combination hybrids, but the contributor provides novel characteristics, making it more flexible than the feature combination method.
6. **Cascade:** This class serves as a tiebreaker. Each recommender is assigned a priority, and lower-priority recommenders act as tiebreakers over higher-priority ones.
7. **Meta-level:** This class has contributing and actual recommenders, but the former completely substitutes the data for the latter one.

According to Burke's taxonomy, our chosen hybrid model, LightFM (explained further in section 3.2), could be classified as a feature combination hybrid.

Additionally, LightFM is a Python implementation of hybrid recommendation algorithms for both implicit and explicit feedback [21].

Explicit and Implicit Rating

- Explicit ratings are created when users give a score to an item. For example, Facebook and YouTube have popular like and dislike buttons that allow users to provide explicit ratings.
- Implicit ratings are more complex and include a variety of user interactions that are not specifically intended to provide ratings to the system. Instead, these interactions can be analyzed to infer positive or negative ratings.

3.2 Our Chosen Model: LightFM

One of the popular techniques for building recommendation systems is collaborative filtering. Collaborative filtering works on the principle of analyzing the behavior of users and items to identify similarities and provide recommendations. LightFM is a Python library that provides an efficient implementation of collaborative filtering-based recommendation systems. In this thesis, we will explore the use of the LightFM library for building a recommendation system.

LightFM, a hybrid matrix factorization model, utilizes latent vectors (embeddings) to represent users and items, akin to traditional collaborative filtering models. Furthermore, it incorporates linear combinations of embeddings of content features to describe each user and product, similar to a content-based model. LightFM unites the advantages of content-based and collaborative recommenders.

To make a prediction, LightFM utilizes six key components. When considering users:

1. When considering users, the first ingredient is a feature matrix F^U_{uf}, which describes the user's possession of a particular feature f.
2. The second component for users is a feature embedding matrix E^U_{fm}, which provides an M-dimensional embedding for each feature f. It is represented as a vector of M numerical values and is learned by the model. The value of M is provided by us, and in LightFM, it is called "no_components".
3. The feature bias vector B^U_f, is also learned by the model and assigns a numerical bias to each feature f.

   When considering items:

4. The first component for item is a feature matrix F^I_{if}, which indicates the quantity of a particular item-feature f possessed by item i. In our implementation, we do not currently use item features, so LightFM generates the feature matrix as an Identity Matrix. This means that every item becomes its own feature, and if we consider two items, our feature matrix would be $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.
5. A feature embedding matrix, denoted as E^I_{fm}, assigns an M-dimensional em-

bedding, which is a vector of M numerical values, to each feature f. The model learns and optimizes this embedding matrix during the training phase to maximize its accuracy. For instance, when M=3, the model can potentially learn that the feature "item 0" corresponds to the embedding [1, 1, 1], while "item 1" maps to [0, 1, -1].

This allows the model to efficiently process and analyze the input data, ultimately leading to better performance and results.

6. The feature bias vector B_f^I, assigns a numerical bias value to each feature f and is also learned by the model. Together, these three components allow LightFM to accurately predict items.

3.2.1 Sizes of the various matrices and vectors

LightFM uses a total of four matrices and two vectors for making its predictions [3].

- The first matrix is the user feature matrix F_{uf}^U, which has dimensions of users by features. Each row in the matrix corresponds to a user and contains numerical values for each feature. We need to explicitly define this matrix to provide the necessary information for the model.
- The second matrix is the user feature embedding matrix E_{fm}^U, which has dimensions of no-of-user features by M. We do not need to explicitly define this matrix; we only need to specify the value of M.
- The third vector is the user feature bias vector B_f^U, which has dimensions of no-of-user features. It is not necessary for us to explicitly define this vector.
- The fourth matrix is the item feature matrix F_{if}^I, which has dimensions of no-of-items by no-of-items-features. Each row in the matrix corresponds to an item and contains numerical values for each feature. We need to explicitly define this matrix to provide the necessary information for the model. However, if we do not provide this matrix, LightFM will generate an identity matrix with dimensions of no-of-items by no-of-items, treating each item as having its own unique feature.
- The fifth matrix is the item feature embedding matrix E_{im}^I, which has dimensions of no-of-item features by M. We do not need to explicitly define this matrix, as we have already specified the value of M.

- Finally, the sixth vector is the item feature bias vector B_f^I, which has dimensions of no-of-item-features. It is not necessary for us to explicitly define this vector. Together, these matrices and vectors allow LightFM to make accurate predictions.

LightFM generates a score for each pair of user and item by utilizing the six ingredients mentioned earlier.

$$S_{ui} = \sigma(\sum_f F_{uf}^U \sum_{m=1}^{M} E_{fm}^U \sum_g E_{gm}^I F_{ig}^I + \sum_f F_{uf}^U B_f^U + \sum_f F_{ig}^I B_g^I) \tag{7}$$

The score is calculated by using the sigmoid function σ on the dot product of the user and item embeddings, along with the biases for the user and item. The sigmoid function ensures that the resulting score is between 0 and 1.

LightFM can be seen as a logistic regression model, where the weights are the embeddings for the users and items E_{fm}^U, and E_{fm}^I, and the biases are learned from the data B_f^U, and B_f^I. These embeddings and biases are optimized during the training process to generate accurate predictions for user-item interactions.

3.2.2 Learning/fitting the weights and biases

To fit and learn the weights and biases E_{fm}^U, E_{fm}^I, B_f^U, and B_f^I, an interaction matrix y_{ui} must be provided indicating whether a user u liked an item i or not. If the user liked the item, the value of y_{ui} is set to 1, otherwise, it is set to -1 [3].

It is important to note that when working with LightFM and sparse matrices, it is recommended to use $y_{ui} = -1$ instead of $y_{ui} = 0$ to indicate that the user did not like the item. This is because LightFM treats any positive value of $y_{ui} > 0$ as an indication that the user liked the item and sets it to $Z_{ui} = 1$ internally. Similarly, any negative value of $y_{ui} <= 0$ is treated as an indication that the user did not like the item, which is set to $Z_{ui} = 0$ internally.

It is not mandatory to specify a value for y_{ui} for every combination of user u and item i in the interaction matrix. LightFM will consider only the specified values. More information on this is provided in the sparse matrices section 3.2.3.

In LightFM, the model parameters are optimized through an iterative process called

training, which involves adjusting the weights and biases to minimize the difference between the predicted scores and the actual interactions in the training set.

During the training process, the model tries to match the predicted scores S_{ui} with the given interactions Z_{ui}. For every known user-item interaction where the user has expressed a positive preference for an item, the model tries to assign a high score (1) to that item. Conversely, for every known user-item interaction where the user has not expressed a preference for an item, the model tries to assign a low score (0) to that item.

It is important to note that not every user-item interaction needs to be provided in the training set. LightFM can handle sparse data, where only a subset of the interactions is available. In this case, the model only tries to match the predicted scores and actual interactions for the interactions that are present in the training set.
To optimize the model parameters, LightFM uses the stochastic gradient descent algorithm, which calculates the gradients of the loss function with respect to the weights and biases and updates them in small steps. This process is repeated iteratively until the loss function converges to a stable minimum, indicating that the model has learned the underlying patterns and relationships in the training data. (More on this is in the section 3.2.3)

LightFM provides four losses to optimize the model weights and biases to match the interactions [3]. Below, each of the four is explained more in detail:

- **logistic:** useful when both positive (1) and negative (-1) interactions are present.

    1. The goal of the logistic loss function is to minimize the difference between the predicted scores and the actual scores for the interactions between users and items in the training data.

    2. The predicted scores are obtained by applying the LightFM model to the user-item pairs in the training data.

    3. The actual scores are represented by the binary values indicating whether the user interacted with the item or not.

    4. The logistic loss function computes the error between the predicted scores and the actual scores using a formula that takes into account the actual and predicted scores for each interaction in the training data.

5. The goal of the training process is to minimize the value of the logistic loss function by adjusting the parameters of the LightFM model.
6. The optimization is typically done using stochastic gradient descent (SGD) or a similar optimization algorithm.

- **BPR:** Bayesian Personalised Ranking pairwise loss [25]. Maximizes the prediction difference between a positive example and a randomly chosen negative example. Useful when only positive interactions are present and optimizing AUC is desired.
    1. Select a user u randomly.
    2. Select an item i that user u likes.
    3. Select another item j that user u does not like.
    4. Calculate the predicted score for both first item i and the second one j using the user and item factorized matrices.
    5. Compute the difference between the predicted scores of both items i and j.
    6. Pass the difference through a sigmoid function to get a weight.
    7. Use this weight to update the model parameters using stochastic gradient descent.
    8. Repeat steps 1-7 multiple times for different users and items.
    9. The goal is to learn the relative ranking between items for each user, rather than trying to predict the exact rating for each user-item pair.
- **WARP:** Weighted Approximate-Rank Pairwise loss [33]. Maximizes the rank of positive examples by repeatedly sampling negative examples until rank violating one is found. Useful when only positive interactions are present and optimizing the top of the recommendation list is desired. It only updates the parameters when it predicts that a negative item has a higher score than a positive item.
    1. WARP updates parameters only when the model predicts a negative item with a higher score than a positive item.
    2. If this does not happen, WARP keeps trying by drawing more negative samples until it either finds a violation in ranking or hits a threshold for trying.

3. If WARP finds a violation in ranking on the first try, it makes a larger gradient update. This indicates that many negative items are ranked higher than positive items based on the current state of the model.

4. If it takes many sampling attempts to find a violation in ranking, WARP makes a smaller update. This infers that the model is likely to be close to the optimum and should be updated at a lower rate.

- **k-OS WARP:** k-th order statistic loss [34]. A modification of WARP that uses the k-th positive example for any given user as a basis for pairwise updates.

3.2.3 Sparse matrices

In the previous section, we explained the workings of LightFM in terms of matrices. However, LightFM requires sparse matrices instead of regular matrices. A sparse matrix is like a regular matrix but stores its data differently. A regular matrix stores its data as a 2D array of numbers, including zeros. If 99% of the matrix values are zeros, then all these zeros are also stored in the 2D array, leading to a large memory. In contrast, a sparse matrix only stores non-zero values. If 99% of the matrix values are zeros, then a sparse matrix will only store 1% of the values, leading to a much smaller memory.
LightFM uses the coo-matrix (a sparse matrix in COOrdinate format) format to store sparse matrices. This format uses three arrays to store data:

- An array of row indices: This array contains the row indices of all non-zero elements in the matrix.
- An array of column indices: This array contains the column indices of all non-zero elements in the matrix.
- An array of data values: This array contains the values of all non-zero elements in the matrix.

In LightFM, the user-item interaction data is represented using sparse matrices. A sparse matrix is a matrix in which most of the elements are zero, and only non-zero elements are stored. The sparse matrix representation consists of three arrays: row array, column array, and data array.

To access a specific element in the sparse matrix, we need to look at the corresponding position in each of the three arrays. For example, to access the value at row r and column c, we need to find the position in the row and column arrays where the

row number is r and the column number is c. Once we have found the position, we can access the value in the data array at the same position.

If a row-column pair is not present in the row and column arrays, it means that the corresponding value is zero. This is because sparse matrices only store non-zero values, and any missing element is assumed to have a value of zero.

We can build the user feature matrix E_{fm}^{U} as a regular matrix and convert it to coo-matrix format. Internally Python will simply drop all matrix elements that are zero which is acceptable.

However, LightFM has a limitation when it comes to the interaction matrix. It only considers the user-item interactions that are specified, which means you don't have to indicate whether a user liked an item or not for cases where you have no information. It achieves this by looking only at the rows (users) and columns (items) present in the row and column arrays of the interaction matrix in coo-matrix format. As a result, any row-column combination (user-item combination) not found in the row and column arrays of the interaction matrix in coo-matrix format will not be utilized for fitting the weights and biases.

If we convert the interaction matrix y_{ui} from regular format to coo-matrix format, then Python will drop all matrix elements that are zero. However, if we interpret $y_{ui} = 0$ as indicating that user u does not like item i, then these values will be dropped by Python during the conversion to coo-matrix format. Consequently, we would only train on the positive interactions where $y_{ui} = 1$, and we wouldn't have any negative interactions.

Therefore, to train the model on negative interactions, we need to specify a value of $y_{ui} = -1$ to indicate that a user u dislikes an item i. This ensures that the corresponding row-column combination is included in the row and column arrays of the coo-matrix format and is used to fit the weights and biases during model training.

So, to summarize the interaction matrix:

- Set $y_{ui} = 1$ if user u liked item i.
- Set $y_{ui} = -1$ if user u did not like item i.
- Set $y_{ui} = 0$ if we do not know whether or not user u liked item i or we do not

want to take this user-item combination into account.

3.2.4 LightFM Set Up

To set up and train a LightFM model, one needs to provide two things:

1. **A user feature matrix in coo-format.** This can be created by converting a regular user feature matrix to a coo-matrix format.
2. **An interaction matrix in coo-format.** This contains information about which users interacted with which items and how strongly. It can also include information about negative interactions.

- Build an interaction matrix in regular format. Make sure that the values are:

  - Set $y_{ui} = 1$ if user u liked item i.
  - Set $y_{ui} = -1$ if user u did not like item i.
  - Set $y_{ui} = 0$ if we do not know whether or not user u liked item i or we do not want to include it.

- Also make sure that row number r is the same user as row number r in the user feature matrix for all r.
- Convert the matrix to coo-format.
- Finally, it is recommended to eliminate all the zero values in the matrix.

Data preparation is one of the critical steps in building a recommendation system using the LightFM library. The data should be properly formatted into the user-item interaction matrix. The matrix should contain the user and item IDs along with the interaction data. The interaction data can be binary, indicating whether the user has interacted with the item or not, or it can be explicit, indicating the rating given by the user. "Explicit" refers to a type of user-item interaction data that is quantitative in nature and provides a specific rating or score given by the user to the item. This rating can be on a numerical scale, or any other relevant rating scale based on the nature of the items being recommended.

Explicit ratings allow the recommendation system to learn and make more personalized recommendations based on the user's specific preferences and past interactions with the items. However, obtaining explicit ratings from users can be difficult and time-consuming, as it requires users to actively rate or score items. In contrast,

"binary" interactions are simpler and more straightforward, as they only indicate whether a user has interacted with an item or not.

LightFM provides a utility function to create the interaction matrix from the raw data. The function can handle both implicit and explicit feedback. The function takes the raw data, along with the IDs of the users and items, and returns the interaction matrix. The function can also handle missing values in the data and can impute them with zeros (see Section 3.2.2).

Once the data is prepared, we can use LightFM to build the recommendation system. The model is trained using the interaction matrix, and the embeddings of the users and items are learned through matrix factorization. The embeddings represent the latent factors that influence the user-item interactions.

LightFM provides several options for model training, including the number of latent factors, the loss function, and the regularization parameters. After the model is trained, we can use it to generate recommendations for the users. The recommendations can be generated by either ranking the items based on their predicted scores for the user or by using the nearest neighbors of the user in the embedding space.

In the LightFM recommender system, it is possible to train and test the model without any additional features, as LightFM could create indicator features during the training process. However, this approach may limit the system's ability to make recommendations for users whose interactions were not present during the training phase.

4 Design and Implementation

This chapter focuses on the design and implementation of the model, providing a detailed exploration of the steps. Within this chapter, we provide an in-depth analysis of the design choices made, including the selection of appropriate algorithms, and techniques to achieve the desired objectives.

4.1 Data Acquisition

Each company aims to leverage the power of data in a data-driven world to add value to its operations. However, data is often scattered across various departments, source systems, and databases, making it difficult to access and analyze. To make matters worse, the necessary analytics tools and skills are often not available within the same environment. As a result, there is an increasing need for the democratization of data and analytics, enabling simple access to and analysis of data by all stakeholders.

Centralizing data stores, harmonizing data definitions, and ensuring good governance is essential for achieving this goal. At ING, the Chief Data Officer and Data Management organization are currently working towards building an Enterprise Data Lake to achieve this objective.

4.1.1 Data Sources

To facilitate the democratization of data and analytics [6], ING WB Advanced Analytics has developed a platform called the Data Analytics Platform (DAP). The DAP provides a centralized location for storing, managing, and analyzing data from various sources.

In our case study, data sources include:

- The cards payments dataset comes from cardprocessor TSYS TS2 platform.
- The payments data source is the Domino project and the database on DAP is prd-domino and the table x-transactions.
- The Grid table comes from the One Client Core team which the database and table are on DAP.

By harmonizing data definitions and ensuring good governance, the DAP enables users to access and analyze data, regardless of their technical background [6].

4.1.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an important step in any data analysis project, including the recommender system we are developing. Our EDA aimed to discover basic statistics about the data and understand the characteristics of the items and users in our dataset. By conducting EDA, we can identify important insights and relationships in the data, which can guide the development of our recommender system [15][16].

In our case, EDA can help us understand the behavior of users and items in our dataset. By analyzing user behavior, we can identify patterns in their preferences, which can guide our choice of algorithms and models for the recommendation. Similarly, by analyzing item characteristics, we can identify key features that are important to users and incorporate these features into our recommendation algorithm. During this stage, we did not have a specific data set in mind and explored user-item and product features equally to gather as much information as possible.

In particular, we focused on the following aspects:

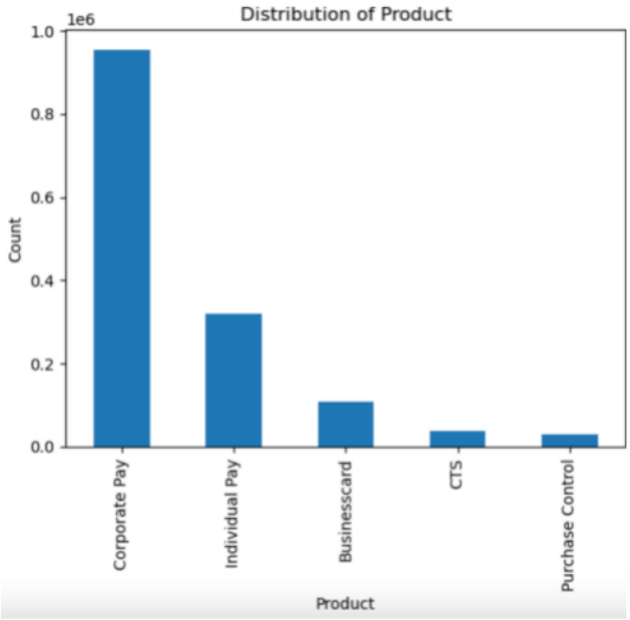
Products: We analyzed the characteristics of the products in our dataset, such as their categories, ratings, and popularity. We also explored how frequently the products were rated and the distribution of ratings across products.

Number of products by users: We investigated the number of products each user rated and how many unique products were rated in the dataset. This information can help us understand the diversity of users' preferences and the sparsity of the data.

Distribution of Product: In the recommender system, the distribution of products among users plays a crucial role in providing personalized recommendations. Figure 9 shows the frequency distribution of products among users. The chart helps to identify the most popular products among users. This information is valuable for the recommender system, as it can be used to recommend the most popular products to new users, as well as to recommend complementary products to users who have already used the popular products. By analyzing the frequency distribution of products among users, we can identify patterns of product usage and user preferences.

The table and a bar chart (Figure 9) show the distribution of product counts. The table displays the name of each product in the 'product' column and the number

of occurrences of each product in the 'count' column. According to the output, the 'Corporate Pay' product has the highest count.



	Product	Count
0	Corporate Pay	954682
1	Individual Pay	319005
2	Businesscard	107986
3	CTS	38239
4	Purchase Control	28381

FIGURE 9: *Distribution of Product*

4.1.3 Data Preparation

The data pre-processing phase in this research involves data quality assessment and feature aggregation. Pandas' library is utilized for data wrangling, processing, and analysis. Specifically, pandas are used for manipulating numeric data [11].

Data Consistency

The following steps were taken for data quality assessment:

- The research involved an evaluation of the raw dataset for data quality. During the assessment, it was discovered that the dataset had a number of impurities such as missing values, inconsistent values, and null values.
  Handling missing data is an important aspect of data cleaning and analysis. Missing data can be indicated by various representations such as NA, and NaN. The approach to dealing with missing values depends on the specific goals and requirements of the study. In the case of this thesis, the approach for dealing with missing values is to replace them with 0.
- To ensure data consistency and handle null values, records with null values were removed from the dataset.
- The evaluation also revealed that there were no duplicate or inconsistent values

in the dataset. This step ensured that the data used in the research was of high quality and could be relied upon for accurate analysis and results.

Train-Test-Split

The dataset used in this research project is divided into two subsets, namely the training set and the testing set. The training set is utilized to develop and train the proposed models, while the testing set is used to evaluate the performance of the models. The testing set generates a list of recommendations and estimated ratings for each user, and evaluation metrics such as Area Under the Curve (AUC) (explained further in section 5.1.1) score are calculated to compare the performance of the models. These metrics provide a quantitative measure of how well the models are performing in terms of predicting user preferences and making accurate recommendations. By carefully evaluating and comparing the performance of different models (e.g., CF, CB) using various evaluation metrics, we can select the best model that can provide the most accurate and effective recommendations for the users.

For the Train/Test split, the dataset was split into train and test sets after feature encoding. The purpose of this step was to prepare the data for training with the model. The split ratio used in this research was 70:30, where 70% of the data was used as the train set and the remaining 30% was used as the test set.

The train set was used to train the LightFM models, while the test set was used to measure the accuracy of the models on real-world data. For the purpose of this work, we will work with LightFM. It will be introduced in more detail in section 3.2. Additionally, a validation set could also be used for model selection and hyperparameter tuning. The test set plays an essential role in the evaluation of the model's performance, as it provides a measure of how well the model can generalize to new, unseen data.

4.1.4 Feature Selection

For feature selection, the following steps helped us:

1. **Understand the domain:** Gain a solid understanding of the domain in which the recommender system will be used. This help identify the most relevant features for the system's users and items.
2. **Identify potential features:** Once the domain is understood, brainstorm potential features that could be useful for the recommender system.
3. **Evaluate feature relevance:** Once potential features have been identified,

evaluate their relevance to the recommendation task. Analyze how much variation in user preferences can be explained by each feature to determine its relevance. Features that explain more variance are more relevant to the task.

In our case, we considered the following features:

- **User features**
  - Transaction amount: This feature helps to identify the purchasing power of the customer and how much money they are spending on a particular product. It can also help to identify trends in customer spending, such as high-value transactions, and can be used to identify patterns in customer behavior.
  - Transactions count: This feature helps to identify how frequently a customer is using their account and can be used to identify changes in customer behavior over time
- **Item (product) features**

  The mentioned item features are various types of cards offered by ING to their clients as products. These cards include the Business card, which is designed for business-related expenses, CTS (Corporate Travel Solutions) card, which caters to corporate travel needs, Corporate Pay card, used for corporate payment purposes, Individual Pay card, intended for individual payment transactions, and Purchase Control card, which provides enhanced control and management over purchasing activities. These cards serve different purposes and are tailored to meet the specific needs of ING's clients.
  - Business card
  - CTS
  - Corporate Pay
  - Individual Pay
  - Purchase control

4.2 Implementation

In this section, we delve into the implementation details of building the recommendation system. The approach we adopt can be summarized into four main steps as can be seen in Figure 10.

1. The first step involves loading and preprocessing the dataset. This step includes cleaning the data and removing any impurities such as missing values, null values, and inconsistent values. Once the dataset is cleaned, it is preprocessed to prepare it for use in model building.

2. The second step is model building. The preprocessed dataset is split into a training set and a testing set. The best parameters for the model are selected, and the model is built using the training set. The model is designed to predict the utility or usefulness of items to a particular user, based on the dataset.

3. The third step is model testing. The model is tested using the test dataset, and the predicted values are compared against the actual values. This step is important in ensuring that the model is accurate and reliable.

4. The fourth and final step is to generate the output. The output includes a list of recommendations based on the predictions for the given user. The area under the curve (AUC) is calculated to evaluate the performance of the model (Section 5). A high AUC score indicates a better-performing model, while a low score indicates the need for further improvements.

Overall, this approach is a generic and widely used framework for building models, allowing for efficient and effective development and implementation of recommender systems.

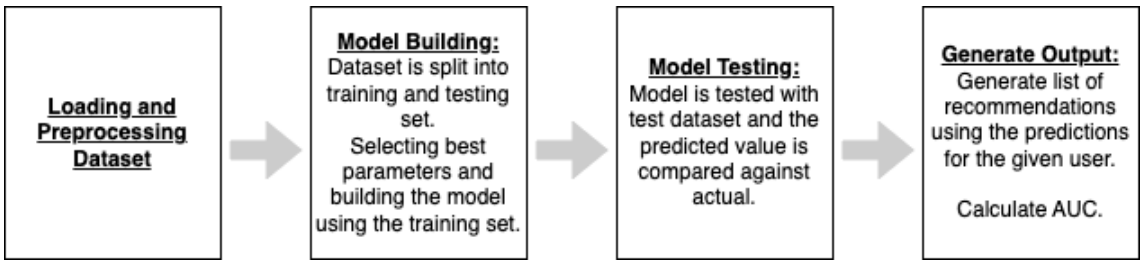


FIGURE 10: *Generic approach for building the models*

The pipeline implementation consists of the following steps:

- **Step 1: Data Preprocessing**
  The first step is to preprocess the interaction and user feature data. The

interaction data contains information about users' interactions with items, while the user feature data contains information about the attributes of the users. In this step, we filter the data to include only the relevant features and remove any missing values.

In the first step of the pipeline, the data preprocessing phase is performed. The code begins by reading a .csv file and storing the data in a *DataFrame*. Subsequently, the *DataFrame* is grouped by the user's id column, and the number of unique products associated with each payer is calculated. To facilitate further analysis, the *DataFrame* is pivoted to transform it into an interaction matrix structure, where rows correspond to users and columns correspond to products. Additionally, any missing values in the interaction matrix are filled with zeros, ensuring a complete and consistent dataset for subsequent steps. This data preprocessing step prepares the foundation for subsequent stages of the pipeline.

- **Step 2: Building the Interaction Matrix**
  The interaction matrix is a binary matrix that indicates whether a user has interacted with an item or not. In this step, we build the interaction matrix from the filtered interaction data.

  In the second step, the focus shifts to building the User FeatureMatrix. This involves extracting the necessary user feature data from the *DataFrame*. The code accomplishes this by selecting the rows from *DataFrame* that have a matching user's id present in the interaction table. By isolating the relevant user features, this step ensures that the subsequent stages of the pipeline can incorporate the necessary information for effective modeling and analysis.

- **Step 3: Building the User Feature Matrix**
  The user feature matrix is a matrix that contains the attributes of the users. In this step, we build the user feature matrix from the filtered user feature data.

  This involves extracting the pertinent interaction data from the *DataFrame*, with a criterion based on the user's id present in the user feature table. By filtering the data based on this criterion, the code ensures that only the relevant interactions are included in the analysis. This step is important for constructing the Interaction Matrix accurately, as it provides the necessary information for modeling and analyzing the user-item interactions effectively.

- **Step 4: Normalizing the User Feature Matrix**

The user feature matrix needs to be normalized before it can be used in the model. The matrix undergoes a normalization process that scales its values between 0 and 1. This normalization step aids in training the model effectively. By scaling the values within a specific range, the model can better understand and process the user feature data, leading to more accurate and meaningful results. The normalization of the User Feature Matrix ensures that the input data is appropriately adjusted and prepared for further analysis and modeling stages of the pipeline.

- **Step 5: Building the Train-Test Split**
  To evaluate the performance of the model, we split the interaction matrix into train and test sets. This splitting process enables the division of the data into separate train and test sets, which serve distinct purposes in the model evaluation. By randomly selecting a fraction of elements from the interaction matrix, the test set is formed. These elements are then set to 0 in the train set, indicating their exclusion from the training process. This division allows for the model to be trained on the train set while reserving the test set for evaluating its performance. The Train-Test Split step is used in assessing the model's ability to generalize to unseen data and provides insights into its overall predictive capabilities.

- **Step 6: Building the LightFM Model**
  In the sixth step of the pipeline, the LightFM model is constructed and configured with the designated parameters. These parameters include the loss function, number of components, learning rate, and user alpha. Subsequently, the model is trained using the train set of the interaction matrix. Notably, the LightFM model has the capability to incorporate the user feature matrix during the training process. By integrating user-specific features, the model can capture more nuanced patterns and tailor recommendations to users. The building of the LightFM model is a pivotal step in creating a hybrid recommendation system that combines both collaborative filtering and content-based approaches, enhancing the accuracy and relevance of the generated recommendations.

- **Step 7: Predicting the Train Set Scores**
  In the seventh step of the pipeline, the trained LightFM model is employed to generate predictions for the train set of the interaction matrix. By leveraging the learned patterns and relationships within the data, the model assigns scores to the interactions between users and items in the train set. These

scores reflect the model's estimation of the preference or likelihood of user-item interactions. The prediction of train set scores enables the evaluation of the model's performance and its ability to capture the underlying patterns in the training data. It serves as a crucial step in assessing the accuracy of the LightFM model in capturing user preferences and generating relevant recommendations.

- **Step 8: Computing the Train Set AUC Score**
  In the eighth step of the pipeline, the true labels for the train set are obtained from the train set of the interaction matrix. These labels indicate whether a user-item interaction occurred or not. The predicted scores generated in the previous step, along with the true labels, are utilized to calculate the area under the ROC curve (AUC) score. The AUC score serves as a metric to assess the model's performance on the train set. It means the training AUC serves as a means to evaluate the performance during the training process. If the test AUC significantly falls below the training AUC, it indicates that the model has been overtrained. It measures the model's ability to distinguish between positive and negative interactions accurately. A higher AUC score indicates better discrimination and predictive power of the model in capturing user preferences and generating relevant recommendations for the train set data.

- **Step 9: Predicting the Test Set Scores**
  In Step 9 of the pipeline, the trained model is applied to predict scores for the test set of the interaction matrix. By leveraging the learned patterns and relationships between users and items, the model generates predictions for the test data, indicating the likelihood of user-item interactions. These predicted scores serve as an essential metric for evaluating the model's performance on unseen data. The predictions are calculated based on the trained model's understanding of user preferences and item characteristics, allowing for an assessment of how well the model generalizes to new instances.

- **Step 10: Computing the Test Set AUC Score**
  In the tenth step of the pipeline, the true labels for the test set are extracted from the test set of the interaction matrix, indicating the presence or absence of user-item interactions. The predicted scores obtained from the model, along with the true labels, are employed to compute the AUC score. This score serves as a measure to evaluate the performance of the model on the test set.

- **Step 11: Repeating the Process**

In the eleventh and final step of the pipeline, the process from steps 5 to 10 is repeated for a total of 40 iterations. This repetitive procedure aims to obtain a more robust assessment of the model's performance by calculating an average AUC score for both the train and test sets. By conducting multiple iterations, potential variations in the performance metrics can be captured, allowing for a more reliable evaluation of the model's predictive capabilities. The AUC scores and corresponding predicted scores are stored in separate lists, facilitating subsequent analysis and comparison to gain insights into the models in generating recommendations and predicting user-item interactions.

The reason for iteration is that the performance of the model can vary depending on the randomly selected elements in the train-test split. By repeating the process, we can obtain a more reliable estimate of the model's performance on both the train and test sets. Additionally, repeating the process with different train-test splits can help to identify the overfitting or underfitting of the model.

5 Evaluation

In this chapter, we focus on evaluating the designed model within the specific context of ING. The evaluation process plays a crucial role in validating the performance and practicality of our proposed approach. By conducting a thorough assessment, we aim to determine how well the model performs in achieving the desired outcomes and meeting the objectives outlined in the earlier sections of the thesis [35].

It is important to highlight that all the tables presented in this chapter are the outcome of executing the coding procedures, which have been comprehensively explained in relevant sections, encompassing a series of detailed steps.

5.1 Evaluation Metrics

When evaluating the performance of our model, we rely on a specific metric called AUC (Area Under the Curve), which is provided by the LightFM framework. AUC is an evaluation metric that assesses the model's ability to correctly rank positive instances higher than negative ones, indicating its effectiveness in making accurate recommendations [3].

In this research study, we place particular emphasis on the calculation and interpretation of AUC scores as the primary evaluation metric for our model. AUC scores provide a quantitative measure of how well our model performs in terms of ranking and predicting user-item interactions. By analyzing and comparing the AUC scores obtained in different experiments and scenarios, we can gain valuable insights into the model's performance and its ability to generate relevant recommendations.

5.1.1 Area Under Curve (AUC)

The ROC curve represents the relationship between FPR and TPR and is commonly used to compare the performance results of Precision and Recall visually. While the ROC curve is difficult to quantify, the AUC index is typically used to measure the accuracy of the model. AUC represents the area under the ROC curve, and a higher AUC value indicates a more accurate model. An AUC value closer to 1 is indicative of excellent performance. Typically, an AUC value of 0.8 or higher is considered to be a high-accuracy model [20].

In this thesis, the performance of the models was evaluated using the Area Under Curve (AUC) metric. AUC can be interpreted as the probability that a randomly

chosen positive example is ranked higher than a randomly chosen negative example. LightFM provides built-in evaluation methods to calculate the AUC score. In our experiments (sections 5.2.1 & 5.2.2), the best AUC score achieved was 84% (0.842), indicating the high accuracy of our models.

5.1.2 LightFM Hyperparameter

In order to achieve an optimum result, an extensive hyperparameter tuning process was conducted. Hyperparameter tuning involves systematically exploring different combinations of hyperparameter values to find the configuration that maximizes the performance of the model.

To determine the optimal hyperparameters for the LightFM model, a grid search technique was employed. A range of values was defined for each hyperparameter, and the model was trained and evaluated for each combination of hyperparameters.

Multiple iterations of the grid search were performed, adjusting the values of key hyperparameters. The performance of the model was carefully monitored and compared across different hyperparameter configurations. This approach aimed to find the best possible configuration that maximizes the accuracy of the recommendation system. In the case of our LightFM model, we focused on tuning five key hyperparameters to improve its performance:

- **Loss:** The LightFM model incorporates several hyperparameters that play a crucial role in shaping its performance. One such hyperparameter is the "*Loss*" function, which determines the type of loss function employed during the training process. In our research project, we focused on three specific options for the Loss function: Logistic, Bayesian Personalized Ranking (BPR), and Weighted Approximate-Rank Pairwise (WARP). These options were carefully selected and examined to assess their impact on the model's training and recommendation outcomes. By exploring and evaluating these different Loss function choices, we aimed to identify the most suitable approach that would yield optimal results in terms of accuracy.
    - LightFM offers three loss functions: Bayesian Personalized Ranking (BPR), Weighted Approximate-Rank Pairwise (WARP), and the standard binary cross-entropy, along with traditional matrix factorization methods. The approach revolves around sampling positive and negative items and conducting pairwise comparisons. For a given user, a positive and negative item are sampled, predictions are made for both, and the difference

is passed through a sigmoid function, which is then used as a weight to update all model parameters via stochastic gradient descent (SGD).

- **No_components:** One hyperparameter for LightFM is "$No\_components$" which governs the number of components or dimensions employed for generating user and item embeddings. These embeddings play a role in capturing the latent factors and characteristics of users and items, enabling the model to make accurate and personalized recommendations. By adjusting the value of "$No\_components$" can explore the trade-off between model complexity and representation capacity. Selecting an optimal number of components allows for striking the right balance between capturing intricate interactions and avoiding overfitting, ultimately enhancing the model's ability to provide meaningful recommendations tailored to individual users' preferences and item characteristics.

- **Learning_rate:** The "$learning\_rate$" governs the speed at which the optimizer adapts and updates the model's parameters during the training process. It is used in determining how quickly the optimizer converges to an optimal solution. A smaller learning rate tends to result in more precise and accurate weights but requires a longer training time to reach convergence. On the other hand, a larger learning rate may allow for faster convergence but runs the risk of overshooting the optimal solution or failing to converge at all. Selecting an appropriate learning rate is essential to strike a balance between convergence speed and the quality of the learned model parameters, ultimately influencing the model's recommendation performance and efficiency.

- **Item_alpha:** One important hyperparameter in the LightFM model is - $item\_alpha$, which controls the strength of L2 regularization applied to the item embedding weights. L2 regularization is a technique used to prevent overfitting by adding a penalty term to the loss function, encouraging smaller and more generalizable weights. By adjusting the item_alpha hyperparameter, we can control the amount of regularization applied to the item embeddings. Higher values of item_alpha increase the regularization strength, resulting in smaller weights and potentially reducing overfitting. On the other hand, lower values of item_alpha reduce the regularization effect, allowing the model to assign more importance to individual item features. Selecting an appropriate value for item_alpha is crucial to strike a balance between preventing overfitting and capturing the unique characteristics of the items in the recommendation process.

- **User_alpha:** This hyperparameter controls the strength of L2 regularization applied to the user embedding weights. L2 regularization is a technique used to prevent overfitting by adding a penalty term to the loss function, encouraging smaller and more generalizable weights. By adjusting the user_alpha hyperparameter, we can control the amount of regularization applied to the user embeddings. Higher values of user_alpha increase the regularization strength, leading to smaller weights and potentially reducing overfitting. Conversely, lower values of user_alpha decrease the regularization effect, allowing the model to assign more importance to individual user features. Selecting an appropriate value for user_alpha is for strike a balance between preventing overfitting and capturing the unique characteristics of the users in the recommendation process.

In the process of tuning our model, determining the appropriate number of epochs was also important. To determine the optimal number of epochs for our final model, we trained our models one epoch at a time during validation to observe when the validation AUC starts to decrease.

It is important to emphasize once more that the particular research study at hand solely concentrates on utilizing AUC scores as the evaluation metric for our model.

5.2 Experiments

To assess the performance of the model, we have conducted two experiments, which are described in detail below. These experiments are designed to provide valuable insights into the performance of the model.

5.2.1 Experiment 1

Experiment 1 is a simulation of a scenario where a model is trained on randomly sampled user-item interaction and tested on remaining user-item interaction. The experiment is repeated several times to obtain an average measure of the model's performance.

The interactions exclusively consist of likes and dislikes, excluding any instances of "not applicable", "not provided", or "unknown" responses. Our approach stemmed from the fact that LightFM lacks a built-in mechanism to handle responses like "not applicable", "not provided", or "unknown" in the dataset. Therefore, we made a decision to address this by considering the absence of card usage as an implicit dislike during training. This was crucial in ensuring that our training data contained both positive and negative feedback, enabling the model to learn effectively. It is important to acknowledge that this approach impacts the interpretation of our experiments, as we treated the absence of card usage as equivalent to disliking a card.

The experiment aims to evaluate how well the model can predict if an existing user, who has already provided *likes* or *dislikes* for some (but not all) cards, would *like* or *dislike* other cards. Training and testing the model on different random selections allow for a more comprehensive evaluation of the model's performance.

The subsequent steps encompass the analysis and interpretation of the provided Python codes for Experiment 1. In this phase, we thoroughly examine the codes and delve into their functionalities, aiming to gain deeper insights and meaningful interpretations. This process contributes to our understanding of the experimental results and facilitates the extraction of valuable conclusions from the conducted analysis.

- **Step 1: Input and copy interaction matrix**
  The function receives an interaction matrix, which encapsulates the user-item interactions. This matrix serves as the primary input for the subsequent operations. To preserve the integrity of the original data, a copy of the interaction

matrix is made, ensuring that any modifications made during the process do not affect the original dataset. This step is for maintaining the integrity and consistency of the data throughout the train-test split process.

- **Step 2: Mapping non-zero entries**
  The function proceeds to map the non-zero entries in the interaction matrix to -1. This mapping operation is essential for differentiating between observed interactions, which were initially represented by non-zero entries, and unobserved interactions, which were denoted by zero entries. By assigning -1 to the previously non-zero entries, the function establishes a clear distinction between these two types of interactions. This distinction plays a significant role in the subsequent train-test split process, where the mapped values will be utilized to differentiate the train and test sets accurately.

- **Step 3: Creating train and test matrices**
  The function proceeds to create two matrices: the train matrix and the test matrix. Both matrices are initialized as copies of the modified interaction matrix, where the non-zero entries have been mapped to -1. These matrices serve as the foundations for the train and test sets in the experimental setup. The train matrix captures the portion of the interaction data used for training the model, while the test matrix represents the remaining portion that will be used for evaluating the model's performance. By creating separate matrices for the train and test sets, the function enables distinct manipulation and analysis of the data in each set.

- **Step 4: Generating random indices for test set**
  In this step, generate random indices that will be used to select a subset of elements from the train matrix for the test set. The number of elements to be selected is determined by the test fraction parameter, which indicates the fraction of random matrix elements to be allocated to the test matrix. To ensure that each element is chosen only once, the indices are generated without replacement. This random selection process guarantees a representative sample from the train matrix, allowing for an accurate evaluation of the model's performance on unseen data.

- **Step 5: Setting test set elements to 0 in train matrix**
  Set the elements in the train matrix that correspond to the randomly selected indices to 0. This action effectively removes the interactions represented by those elements from the train set. By eliminating these interactions, the test set is ensured to contain only unseen interactions, allowing for an accurate

evaluation of the model's ability to generalize to new data. This step effectively simulates the separation of train and test sets, ensuring the independence of the two datasets for robust model evaluation.

- **Step 6: Setting complementary indices to 0 in test matrix**
  Utilizes the complementary indices, which are obtained by subtracting the randomly selected indices from the complete set of indices. These complementary indices represent the remaining elements that were not included in the train matrix. The function sets the elements in the test matrix that correspond to these complementary indices to 0. This action ensures that the test matrix exclusively contains the interactions that were not included in the train matrix. By isolating these remaining interactions in the test matrix, the evaluation of the model's performance on unseen data is facilitated, allowing for a comprehensive assessment of its generalization capabilities.

- **Step 7: Reshaping matrices**
  Performs a reshaping operation on the flattened versions of the train and test matrices. This reshaping process is carried out to restore the matrices to their original structure and dimensions, matching the shape of the original interaction matrix. By reshaping the matrices, their arrangement is reverted to align with the original layout, ensuring consistency and compatibility for further analysis and evaluation. The restored structure allows for a proper interpretation of the results and facilitates comparisons with the original data, enabling a comprehensive understanding of the model's performance.

- **Step 8: Return train and test matrices**
  Returning the train and test matrices as the final output. These matrices serve as essential components for the subsequent stages of the experiment in the study. The train matrix is utilized for training the model, allowing it to learn from the observed interactions between users and items. On the other hand, the test matrix is employed for evaluating the model's performance by measuring its ability to accurately predict unseen interactions. By providing the train and test matrices as output, the function enables the seamless continuation of Experiment 1, facilitating the assessment of the model and its generalization capabilities to unseen data.

Execution Outcomes of Experiment 1:

Upon executing the mentioned steps, we have acquired the result presented in Table 5. Table 5 displays the average AUC scores for three different loss functions (WARP,

BPR, and Logistic) evaluated in Experiment 1. The experiment is repeated 40 times to obtain an average measure of the model's performance. A higher AUC score indicates better performance. Additionally, We calculated the means standard deviation (the second tiny value (+/-) in the tables). The standard deviation is used to indicate the variability of the AUC scores obtained during the training of the model. Standard deviation is a measure of how spread out the data is from the mean. A small standard deviation means that the data points are close to the mean, while a large standard deviation indicates that the data points are spread out over a wider range of values.

Experiment 1-Train Set				
AUC	**No_component**	**warp**	**bpr**	**logistic**
	1	0.837 +/- 0.013	0.834 +/- 0.010	0.522 +/- 0.077
	2	0.839 +/- 0.009	0.841 +/- 0.013	0.498 +/- 0.062
	3	0.843 +/- 0.010	0.844 +/- 0.012	0.520 +/- 0.075
Experiment 1-Test Set				
AUC	**No_component**	**warp**	**bpr**	**logistic**
	1	0.832 +/- 0.013	0.832 +/- 0.008	0.519 +/- 0.076
	2	0.837 +/- 0.008	0.842 +/- 0.009	0.502 +/- 0.088
	3	0.834 +/- 0.009	0.843 +/- 0.007	0.517 +/- 0.076

TABLE 5: *AUC value for Experiment 1*

5.2.2 Experiment 2

Training on random rows means selecting a random subset of the rows from a given dataset. The primary objective of Experiment 2 is to evaluate the model's performance specifically for new clients who have not provided any explicit likes or dislikes. This experiment allows us to assess how well the model performs in recommending items to clients with no prior interaction history.

This approach is sometimes used in order to create more diverse training sets or to prevent overfitting specific patterns in the data. In this experiment, the training set is obtained by randomly selecting a subset of the rows from the original dataset, rather than random sampling from the entire interaction matrix. The purpose of experiment 2 is to evaluate how well the model performs when trained on a more diverse set of user-item interactions. Training and testing the model on different subsets of the data allows for a more comprehensive evaluation of the model's ability to generalize to new and unseen data. The same LightFM model with the same hyperparameters is used in both experiments to ensure consistency and comparability between the results. The interactions are limited to expressing likes and dislikes, and we do not consider responses such as "not applicable", "not provided", or "unknown".

The following steps encompass the analysis and interpretation of the provided Python codes for Experiment 2. This analysis contributes to the broader understanding of the experiment's results and aid in drawing meaningful conclusions.

1. **Step 1: Input and copy interaction matrix**
   In Experiment 2, the function accepts an interaction matrix, as an input. This matrix captures the relationships between users and items based on their interactions. To preserve the integrity of the original data, a copy of the interaction matrix is made and stored in the variable. This ensures that any modifications made during the process do not affect the original matrix, allowing for accurate analysis and comparison.

2. **Step 2: Mapping non-zero entries**
   Similar to Experiment 1, the function maps the non-zero entries in the interaction matrix to -1. This mapping is crucial in differentiating observed interactions, which were initially represented by non-zero entries, from unobserved interactions, which were denoted by zero entries. By assigning -1 to the non-zero entries, the function establishes a clear distinction between these

two types of interactions, facilitating the subsequent train-test split process.

3. **Step 3: Creating train and test matrices**
   To conduct Experiment 2, the function initializes two matrices: train matrix and test matrix. Both matrices are created as copies of the modified interaction matrix obtained in the previous step. These matrices will serve as the train and test sets, respectively, enabling the training and evaluation of the model on a subset of randomly selected rows from the original dataset. By initializing these matrices, the function prepares the necessary data structures for subsequent steps in Experiment 2.

4. **Step 4: Generating random indices for test set**
   In order to create the test set for Experiment 2, the function first determines the total number of rows in the interaction matrix. It then calculates the specific number of rows that should be included in the test set based on the provided test fraction parameter. Random indices are generated without replacement, ensuring that each index corresponds to a unique row in the interaction matrix. These randomly selected indices represent a subset of rows that will be included in the test set, allowing for the evaluation of the model's performance on unseen data.

5. **Step 5: Setting test set rows to 0 in train matrix**
   To simulate the exclusion of certain rows from the training process in Experiment 2, the function sets the corresponding rows in the train matrix to 0 using the randomly generated test indices. By setting these rows to 0, the interactions represented by those rows are effectively removed from the training data. This ensures that the test set contains only unseen interactions, allowing for a reliable evaluation of the model's performance on novel data.

6. **Step 6: Setting complementary indices to 0 in test matrix**
   To ensure that the test matrix in Experiment 2 exclusively contains rows that were not included in the train matrix, the function sets the corresponding rows in the test matrix to 0 using the complementary indices. These complementary indices are obtained by subtracting the randomly selected test indices from the complete set of indices. By setting these complementary rows to 0, the function guarantees that the test matrix only consists of rows that were not used for training the model. This separation between the train and test matrices facilitates an unbiased evaluation of the model's performance on unseen data.

7. **Step 7: Return train and test matrices** The function concludes by re-

turning the train and test matrices as the final output. These matrices play a crucial role in Experiment 2, where the objective is to train the model using a randomly sampled subset of rows and assess its performance on the remaining unseen rows. The train matrix represents the modified dataset specifically designed for model training, containing a subset of randomly selected rows. On the other hand, the test matrix serves as the dataset for evaluating the model's generalization abilities, encompassing the remaining unseen rows. By returning these train and test matrices, the function enables the subsequent training and evaluation steps in Experiment 2 of the study.

Execution Outcomes of Experiment 2:

Upon executing the mentioned steps, we have acquired the result presented in Table 6. Table 6 presents the average AUC scores obtained from Experiment 2, where a LightFM model was trained on a randomly selected subset of rows from a given dataset using three different loss functions: WARP, BPR, and logistic. The results show that the model trained with the logistic loss function achieved the highest AUC score, indicating better performance in predicting whether a user would like or dislike a card based on their previous interactions. This suggests that the logistic loss function might be a more suitable choice for training the LightFM model when using random row selection as a method to prevent overfitting or increase the diversity of the training set.

Experiment 2-Train Set				
AUC	**No_component**	**warp**	**bpr**	**Logistic***(user_alpha=0)*
	1	0.495 +/- 0.134	0.602 +/- 0.108	0.844 +/- 0.001
	2	0.502 +/- 0.127	0.561 +/- 0.143	0.842 +/- 0.005
	3	0.509 +/- 0.122	0.569 +/- 0.106	0.843 +/- 0.006
Experiment 2-Test Set				
AUC	**No_component**	**warp**	**bpr**	**logistic***(user_alpha=0)*
	1	0.490 +/- 0.135	0.604 +/- 0.109	0.843 +/- 0.011
	2	0.503 +/- 0.126	0.566 +/- 0.140	0.838 +/- 0.010
	3	0.507 +/- 0.121	0.574 +/- 0.101	0.841 +/- 0.008

TABLE 6: *AUC value for Experiment 2*

■ Experiments comparison

Experiment 1 and Experiment 2 differ in the way the training set is constructed. In experiment 1, the training set is created by randomly sampling from the entire interaction matrix, while in experiment 2, the training set is obtained by randomly

selecting a subset of rows from the original dataset.

Additionally, Experiment 1 evaluates the performance of the LightFM model in predicting user-item interactions by training the model on random samples of user-item interactions and testing it on the remaining user-item interactions. On the other hand, Experiment 2 aims to evaluate how well the LightFM model can generalize to new and unseen data by training the model on a more diverse set of user-item interactions.

In terms of results, the AUC scores obtained in the experiments are different. The AUC scores for both training and test sets are higher in Experiment 1 compared to Experiment 2 for all loss functions, indicating that the model trained on randomly sampled user-item interactions performs better than the model trained on randomly selected subsets of rows. However, the AUC scores for the logistic loss function in Experiment 2 are higher than the other loss functions, suggesting that the logistic loss function might be a more suitable choice for training the LightFM model when using random row selection as a method to prevent overfitting or increase the diversity of the training set.

5.3 Building the Baseline Recommendation Systems Model

The goal of building a baseline model is to establish a simple and reasonable performance benchmark for comparison with other more complex models. A baseline model is often a simple model that uses basic techniques to make predictions without considering the complex user-item interactions. The performance of other models can be compared with this baseline model to evaluate if the additional complexity and computational cost of the more advanced models are justified. A good baseline model should be simple enough to be easily interpretable, but not too simple that it is practically useless. It serves as a reference point for measuring the performance of more sophisticated models and helps to identify the additional value they bring to the recommendation problem.

Our approach for building a baseline model is to create a simple implementation that assumes the rating given to an item is binary (either 0 or 1). The model predicts the rating of an item for a given user by calculating the average like for each item. The Baseline Model class has two main functions: fit and predict. The fit function takes a sparse interaction matrix as input and performs the following steps:

1. Convert the sparse matrix to a compressed sparse column format (CSC).

   In the baseline model's fit function, one of the initial steps involves converting the sparse interaction matrix into the compressed sparse column format (CSC). This conversion is performed to enhance the efficiency of memory usage and improve the computational capabilities of the model. By employing the compressed sparse column format, the representation of the interaction matrix becomes more compact, enabling more efficient storage and manipulation of the data. This format's advantages include reduced memory footprint and improved performance during subsequent computations and operations on the matrix. The conversion to the compressed sparse column format ensures that the baseline model can effectively handle the sparse nature of the interaction data while maintaining optimal resource utilization.

2. Extract the index pointer (indptr), indices, and data from the CSC matrix.

   The next step involves decomposing the compressed sparse column (CSC) matrix into three distinct arrays. This decomposition process extracts essential information from the CSC matrix, enabling efficient access to the non-zero elements and their corresponding row and column indices. The index pointer array specifies the range of column indices for each column in the matrix,

facilitating efficient traversal and retrieval of the elements. The indices array stores the row indices associated with the non-zero elements, allowing for direct referencing and retrieval of specific entries. Lastly, the data array holds the actual values of the non-zero elements, providing the necessary information for subsequent computations and calculations. This decomposition step plays a crucial role in facilitating efficient data access and manipulation within the baseline model.

3. Create a copy of the data and replace all negative values with 0 and all positive values with 1.

   After extracting the index pointer, indices, and data arrays, the next step involves creating a copy of the data array and applying value transformations. The purpose of this step is to modify the values within the data array to facilitate a simplified modeling process. Specifically, any negative values present in the data array, representing a lack of interaction, are set to 0. This transformation signifies the absence of interaction between users and items. On the other hand, positive values, indicating the presence of interaction, are transformed to 1. This binary representation simplifies the modeling approach by converting the interaction matrix into a more straightforward form, where the focus lies on the presence or absence of interactions rather than the specific values. By performing this data copy and value transformation, the baseline model establishes a simplified and binary representation of the interaction matrix, setting the foundation for subsequent computations and calculations.

4. Calculate the average *like* for each item by iterating over the index pointer array and taking the mean of the corresponding data array elements.

   Once the data has been copied and transformed, the baseline model proceeds to calculate the average *like* for each item. This process involves iterating over the index pointer array, which serves as a guide for accessing the relevant data array elements associated with each item. By utilizing the index pointer array, the model efficiently retrieves the data elements specific to each item. Subsequently, the mean value is computed for these data elements, capturing the average propensity of users to interact with the item. The resulting average *like* provides valuable insight into the overall user-item interactions and serves as a measure of the item's attractiveness or popularity among users. This calculation of the average *like* allows the baseline model to establish a reference point for making predictions based on the observed user-item interactions.

5. Save the calculated average *like* for each item.

   The baseline model stores the computed average *like* values for each item in the "average_like" attribute of the class instance. This storage mechanism ensures that the average *like* values are readily available for future prediction tasks. By retaining these values, the model can efficiently make predictions by accessing the average *like* corresponding to the items of interest. Storing the average *like* values within the class instance enables easy retrieval and utilization throughout the model's lifespan, ensuring that the predictions can be generated promptly and accurately.

The predict function takes a list of users and a list of items as input and returns the predicted rating for each item. It does so by using the previously calculated average like values for each item.

The results (Table 7) show that the baseline model in Experiment 1 achieved good AUC scores on both sets with a relatively small number of epochs (4). Therefore, these results can be used to compare the performance of other more complex models and evaluate if the additional complexity and computational cost of these models are justified.

Experiment 1		
Baseline -Train Set		
AUC	0.843 +/- 0.009	epochs=4
Baseline -Test Set		
AUC	0.841 +/- 0.004	epochs=4

TABLE 7: *AUC value for Baseline-Experiment 1*

The results (Table 8) show that there is a slight improvement in the AUC score in Experiment 2 which suggests that the baseline model is relatively stable and not sensitive to changes in the data or implementation.

Experiment2		
Baseline -Train Set		
AUC	0.842 +/- 0.002	epochs=4
Baseline -Test Set		
AUC	0.843 +/- 0.007	epochs=4

TABLE 8: *AUC value for Baseline-Experiment 2*

It is also worth noting that the standard deviations in the AUC scores are relatively small, which indicates that the model's performance is consistent across different

runs of the experiment.

5.4 Adding more features

After implementing the LightFM model with two initial features (transaction amount and transaction count), there is a hypothesis that the model's performance is not optimal, and there may be room for improvement. One way to potentially improve the model's performance is to add more features to the model. By adding more features, we may be able to capture additional information about users and items Adding more features to the model could provide additional information about users and items that may improve the model's ability to make accurate recommendations.

There are several features that could consider adding, such as currency, country, and industry (NAICS [5]code).

After incorporating the new features into the model, we conducted an evaluation and found that there was minimal impact on AUC. In other words, the AUC did not significantly change with the addition of these new features. Although, adding new features to the model can still provide valuable insights into the recommendation problem and help identify relevant features for future work.

Reasons

There could be several reasons for the tiny change in AUC after adding new features to the LightFM model.

One possible reason is that the new features are not informative or relevant to the recommendation. In other words, the new features may not capture important patterns or trends in user behavior or item characteristics that are useful for making accurate recommendations.

Another possible reason is that the newly added features may have interacted with the existing features in unexpected ways. This could be due to complex relationships between features or interactions between features that the model was not able to capture effectively. Finally, the tiny decrease in AUC may be due to the model's inability to handle the increased complexity of the feature space. Adding more features to the model can lead to a higher-dimensional feature space, which can make it more difficult for the model to learn patterns and make accurate recommendations.

[5] *T*he North American Industry Classification System (NAICS) is the standard used by Federal statistical agencies in classifying business establishments for the purpose of collecting, analyzing, and publishing statistical data related to the US. business economy.

5.5 Replacing transaction amount and count by Range of Amount

We started by defining the ranges to use to divide the transaction amounts. In this case, 6 ranges are defined that start from 0 and cover the entire value of 1.0E7.

To categorize each transaction in the dataset, we compared the transaction amount to the upper and lower boundaries of predefined ranges. If the transaction amount falls within a certain range, mark it as belonging to that range.

Once we have assigned a range to each transaction, group the transactions by range and count the number of unique clients in each group. This will give the number of clients that have a transaction amount in each assigned range.

Based on the results of the two experiments, it appears that using the range of transaction amounts as a user feature can have a moderate impact on the performance of recommendation algorithms, as measured by AUC.

Experiment 1-Train Set				
AUC	**No_component**	**warp**	**bpr**	**logistic**
	1	0.484 +/- 0.126	0.521 +/- 0.094	0.492 +/- 0.094
	2	0.518 +/- 0.115	0.507 +/- 0.093	0.492 +/- 0.074
	3	0.527 +/- 0.113	0.539 +/- 0.133	0.489 +/- 0.084
Experiment 1-Test Set				
AUC	**No_component**	**warp**	**bpr**	**logistic**
	1	0.484 +/- 0.126	0.517 +/- 0.093	0.490 +/- 0.094
	2	0.514 +/- 0.116	0.506 +/- 0.095	0.490 +/- 0.075
	3	0.525 +/- 0.111	0.537 +/- 0.132	0.489 +/- 0.084

TABLE 9: *AUC value for ranges of amount-Experiment 1*

In the first experiment (Table 9), we see that all three recommendation algorithms (BPR, logistic, and WARP) perform slightly better on the training data when the range of transaction amounts is included as a user feature. However, on the test data, only the BPR algorithm shows a slight improvement, while the logistic and WARP algorithms perform slightly worse when the range of transaction amounts is included as a user feature.

In the second experiment, Table 10, the results are somewhat similar, with the logistic and WARP algorithms performing slightly worse on both the training and test data when the range of transaction amounts is included as a user feature.

By comparing two experiments with ranges of amounts, it appears that including

Experiment 2-Train Set				
AUC	**No_component**	**warp**	**bpr**	**logistic**
	1	0.477 +/- 0.112	0.493 +/- 0.084	0.479 +/- 0.090
	2	0.496 +/- 0.099	0.501 +/- 0.057	0.483 +/- 0.105
	3	0.454 +/- 0.117	0.501 +/- 0.057	0.475 +/- 0.082
Experiment 2-Test Set				
AUC	**No_component**	**warp**	**bpr**	**logistic**
	1	0.480 +/- 0.109	0.492 +/- 0.084	0.477 +/- 0.092
	2	0.490 +/- 0.099	0.502 +/- 0.058	0.482 +/- 0.107
	3	0.449 +/- 0.119	0.500 +/- 0.053	0.474 +/- 0.085

TABLE 10: *AUC value for ranges of amount-Experiment 2*

the range of transaction amounts as a user, feature can have a small to moderate impact on the performance of recommendation algorithms, with the BPR algorithm showing the most consistent improvement across both experiments. However, the impact of this feature on performance is not particularly strong, and it may be more beneficial to focus on other users' features or optimization techniques to further improve the performance of recommendation algorithms.

Defining the ranges has two important benefits:

- Identifying potential outliers: Dividing transaction amounts into several ranges can help identify potential outliers or transactions that fall outside of the expected range. These transactions can then be further analyzed to understand why they fall outside of the expected range.
- Simplification of analysis: Dividing transaction amounts into several ranges can simplify the analysis of transaction data by grouping similar transaction amounts together. Through the use of these, it may be possible to find trends and patterns in transaction data that can be used to guide business choices. It would be challenging to spot any trends or patterns, for instance, if the dataset consisted of millions of transactions with various quantities, merely by looking at the individual transactions. However, by dividing the transactions into ranges, we can group similar transactions together and analyze them collectively.

5.6 Exploring the performance of other models

In this section, we delve into the exploration of alternative models. We selected XGBoost and Sklearn Decision models. Regarding our search context and ING domain, the selection of these two models for comparison in the ING recommender system is based on their relevance, popularity, and performance in the field of machine learning and recommendation systems. XGBoost is a widely used gradient-boosting framework known for its high accuracy and performance in handling large-scale datasets. It has been successfully applied in various domains, including recommendation systems, and has shown promising results in terms of predictive performance. Sklearn Decision, on the other hand, is a decision tree-based algorithm implemented in the popular scikit-learn library. Decision trees are interpretable and capable of handling both categorical and numerical features, making them suitable for ING recommendation tasks.

Exploring the performance of two other models namely XGBoost and Sklearn Decision Tree is for several reasons:

- First, it helps us evaluate the performance of different algorithms in the context of our recommender system. By comparing the performance of these models with the baseline LightFM model, we can gain insights into their strengths and weaknesses in generating accurate recommendations.
- Second, it allows us to assess the suitability of alternative approaches and determine if they outperform the LightFM model in terms of prediction accuracy or other performance metrics. This information is valuable for decision-making and selecting the most appropriate model for our specific use case.
- Lastly, exploring other models provides a broader perspective on the capabilities of different recommendation algorithms, enabling us to explore different techniques and potentially discover novel insights that can further improve the accuracy and performance of our recommender system.

5.6.1 XGBoost Model

In the thesis, we decided to compare the performance of the LightFM model with another popular model, XGBoost. While LightFM is a hybrid recommender system that can incorporate both user and item features, XGBoost is a powerful gradient-boosting algorithm that can handle large and complex datasets. Therefore, by comparing the performance of both models, we could gain a better understanding of

the strengths and limitations of each model, and select the most appropriate model for the recommendation system.

XGBoost model has the capability of handling high-dimensional sparse data and can effectively deal with the cold-start problem by incorporating side information such as user demographics and item descriptions. We anticipate that XGBoost can provide better accuracy and scalability in predicting user-item interactions [29].

The model is trained and tested on a user-feature matrix and an interaction matrix where each row represents a user and each column represents an item (in our case, a card).

1. In the initial step of data preparation, we retrieve the user feature matrix and the interaction matrix for a specific item. The interaction matrix is transformed into a binary representation, whereby positive values are encoded as 1 to signify an interaction between the user and the item, while all other values are assigned as 0, indicating the absence of an interaction. This binarization process simplifies the subsequent analysis by focusing on the presence or absence of interactions, enabling us to evaluate the performance of the models in predicting user-item interactions accurately.

2. Following data preparation, the dataset is divided into training and testing sets using the train_test split function from the scikit-learn library. This step ensures an unbiased evaluation of the model's performance by allocating 70% of the data to the training set and the remaining 30% to the testing set.

3. To enhance the performance of the XGBoost model, feature scaling is applied to the training data. The MinMaxScaler from the scikit-learn library is utilized for this purpose. By scaling the features, we ensure that they all have a comparable scale, preventing any particular feature from dominating the learning process due to its larger magnitude. This normalization step promotes stable and efficient training of the XGBoost model by bringing the features to a standardized range.

4. The XGBoost model is configured by setting a range of parameters to control its behavior. These parameters include the choice of objective, which in this case is binary logistic regression, and the evaluation metric used to assess model performance(AUC). Additionally, the maximum depth of the decision trees, regularization parameters (lambda and alpha), learning rate (eta), subsampling ratio (subsample), and column subsampling ratio (colsam-

ple_bytree) are specified. These parameter settings are crucial in determining the model's complexity, regularization, and overall learning dynamics (section 7.2).

5. The XGBoost model is trained using the scaled training data and the corresponding target label. During the training process, the model learns the underlying patterns and relationships between the features and the target variable. By fitting the model to the training data, it optimizes the specified objective function and adjusts the parameters of the decision trees to minimize the loss. This training step is crucial in enabling the model to capture complex relationships and make accurate predictions on unseen data.

6. The trained XGBoost model is employed to generate predictions for the training set (train_score) and the testing set (test_score). These predicted labels are subsequently utilized to compute AUC for both the training set and the testing set.

The goal of the XGBoost model is to predict the likelihood of a user interacting with a specific card based on its features. The model implements a binary classification problem where a user is either likely to interact with the card or not. XGBoost works by building multiple decision trees sequentially, where each subsequent tree corrects the errors made by the previous tree, with the goal of minimizing the loss function. The AUC score for the XGBoost model on the testing set is also computed (Table 11).

After tuning the XGBoost parameters (See the section 7.2), we have obtained the following result:

XGBoost-Train Set		
AUC	0.683 +/- 0.010	Objective=binary:logistic
XGBoost-Test Set		
AUC	0.627 +/- 0.196	Objective=binary:logistic

TABLE 11: *AUC value for XGBoost*

Through parameter tuning, it was discovered that the XGBoost model outperformed the LightFM model in predicting user-item interactions, resulting in a higher AUC score on the testing set. Specifically, the XGBoost model achieved a Test AUC score of 0.627, surpassing the LightFM model's Test AUC score of 0.519 on "*logistic*".

5.6.2 Sklearn Decision Tree Model

Another comparison is LightFM with Sklearn Decision Tree. Sklearn's Decision Tree Model offers a flexible solution for analyzing and predicting complex data. The decision tree forms a predictive model which maps the input to a predicted value based on the input's attributes. Each interior node in the tree corresponds to an attribute and each arc from a parent to a child node represents a possible value or a set of values of that attribute. The construction of the tree begins with a root node and the input set. An attribute is assigned to the root and arcs and sub-nodes for each set of values are created. The input set is then split by the values so that each child node receives only the part of the input set which matches the attribute value as specified by the arc to the child node. The process then repeats itself recursively for each child until splitting is no longer feasible. Either a single classification (predicted value) can be applied to each element in the divided set or some other threshold is reached [14].

Compared to the LightFM recommender system, which focuses on collaborative filtering and factorization techniques, the decision tree model offers a more rule-based approach to making predictions. In this thesis, we explore the performance of Sklearn's Decision Tree Model in comparison to LightFM in the context of the recommendation system.

1. Similar to other models, to assess the performance of the recommendation system, a train-test split was conducted on the interaction matrix. This split was achieved using the buildTrainTest function, which partitioned the matrix into training and testing sets. The function randomly assigned a fraction of 0.3 to the test set, thereby allocating 30% of the rows for evaluation purposes.

2. In the data preparation phase, the user feature matrix was carefully extracted for both the training and testing sets. This extraction process was based on the non-zero entries of the corresponding interaction matrix, ensuring that only the relevant user features were considered for the subsequent prediction task. Furthermore, to create the target labels, the values of the interaction matrix were flattened. This step allowed for a streamlined representation of the user-item interactions, facilitating the modeling process and enabling accurate predictions.

3. The configuration of the Decision Tree model involved fine-tuning several hyperparameters to optimize its performance. These hyperparameters included the criterion used for splitting nodes, the strategy employed for splitting nodes

(such as "best" or "random"), the maximum depth allowed for the tree, the minimum number of samples required to split an internal node, the minimum number of samples required to form a leaf node, the maximum number of features considered when searching for the best split, the random state parameter to ensure reproducibility and the maximum number of leaf nodes allowed in the tree. By carefully selecting these hyperparameters, the Decision Tree model could be customized to effectively capture the underlying patterns and relationships within the data.

4. The trained Decision Tree model was utilized to make predictions on both the training and testing sets. These predictions were obtained in the form of probability scores, representing the estimated likelihood of a positive interaction based on the learned decision rules of the model. Subsequently, AUC was computed for both the training set and the testing set.

The Sklearn Decision Tree Model is a non-hybrid model that uses only the interaction data between users and items. The results showed (Table 12) that the Sklearn Decision Tree Model had a higher Train AUC score of 0.658 compared to LightFM's Train AUC score of 0.522. However, the Test AUC score for the Sklearn Decision Tree Model was lower at 0.430 compared to LightFM's Test AUC score of 0.519. Therefore, it can be concluded that LightFM performs better than the Sklearn Decision Tree Model, despite having a lower Train AUC score.

Decision Tree-Train Set		
AUC	0.658 +/- 0.014	Criterion=gini
Decision Tree-Test Set		
AUC	0.430 +/- 0.006	Criterion=gini

TABLE 12: *AUC value for Sklearn Decision Tree*

5.7 Reflection on Validity Threats

In this evaluation work, there are several validity threats that should be taken into consideration when interpreting the results.

- Firstly, the construction of the training set in Experiment 1 and Experiment 2 introduces a potential bias. Randomly sampling from the entire interaction matrix in Experiment 1 may not accurately represent the real-world user-item interactions, while randomly selecting a subset of rows in Experiment 2 may lead to the exclusion of important data points. These differences in training set construction could impact the generalizability and reliability of the results.
- Another validity threat arises from the evaluation metrics used, specifically the AUC scores. While AUC is a commonly used metric for evaluating recommendation systems, it only captures the ranking performance of the models and may not fully reflect their overall effectiveness in real-world scenarios. Other metrics, such as precision, and recall could provide a more comprehensive understanding of the models' performance and should be considered in future research.
- Additionally, the impact of adding new features to the model is found to be minimal in terms of AUC improvement. This suggests that the selected features, such as currency, country, and industry, may not have strong predictive power in the given context. Exploring and selecting more informative features or alternative feature engineering techniques could be a potential avenue for future investigation.

5.8 Key Learning

In terms of key learnings, the results highlight the importance of carefully constructing the training set to ensure its representativeness and diversity. Experimenting with different methods, such as random sampling and subset selection, provides insights into the performance variations of the recommendation algorithms. It also emphasizes the need for robust evaluation metrics that align with the specific objectives of the recommender system and capture a more holistic view of its performance.

Furthermore, the comparison between different models, such as XGBoost and Sklearn Decision Tree, provided insights into their strengths and weaknesses. While XGBoost outperformed LightFM in predicting user-item interactions, Sklearn Decision

Tree had a higher Train AUC score but performed less effectively on the test set. These findings emphasize the importance of carefully selecting and tuning models based on specific objectives and datasets.

6 Conclusion

This thesis focuses on developing a cross-selling recommender system using the LightFM library, which is a hybrid matrix factorization model that combines the benefits of content-based and collaborative recommenders. It follows a four-step approach for building the models, including splitting the interaction matrix into training and testing sets, specifying hyperparameters, training the model, and evaluating its performance using AUC values.

Regarding the LightFM model, this master thesis includes two experiments. Comparing the two experiments, we can observe that both experiments have a similar number of components and the same hyperparameters.

Moreover, the results of experiment 2 suggested that training the model on a more diverse set of user-item interactions can lead to better performance. This is because selecting a random subset of the rows from the original dataset can create a more diverse training set, which helps the model to generalize better to new and unseen data.

There is a hypothesis that shows the two loss functions may be more suitable for different types of training data. The BPR loss function may work better for randomly sampled user-item interactions, while the logistic loss function may perform better on a more diverse set of user-item interactions.

The thesis also includes a baseline model, which predicts the rating of an item for a given user by calculating the average like for each item. The baseline model serves as a performance benchmark for comparison with more complex models.

The research project concludes with a comparison of the LightFM model with XGBoost and Sklearn Decision Tree, popular models in recommendation systems. The comparisons provide insights into the strengths and weaknesses of each model and their suitability for specific recommendation tasks.

6.1 Answers to Research Questions

RQ1: What are the specific requirements and considerations for developing recommender systems tailored to the banking domain, according to published literature?

By examining studies conducted in the literature review, valuable insights are obtained regarding the optimization of recommender systems for generating personalized recommendations. This knowledge helps identify strategies to improve the implementation of recommender systems in the banking context, resulting in more tailored and relevant recommendations for clients.

The specific requirements and considerations for developing recommender systems tailored to the banking domain, according to published literature, include leveraging contextual information from transactional data, incorporating customer preferences and financial goals, optimizing data utilization for business performance and revenue generation, and addressing the unique challenges and opportunities of personalized recommendations in the banking sector.

According to the results conducted in the banking industry, the implementation of recommender systems needs some requirements to generate personalized recommendations for clients through several strategies:

Through data collection and integration, banks should collect and integrate diverse data sources. By capturing a wide range of customer data, banks can gain a deeper understanding of individual preferences and behaviors, enabling more accurate and personalized recommendations.

Employing advanced recommendation algorithms, such as collaborative filtering, content-based filtering, and hybrid approaches, can improve the performance and relevance of recommendations. Banks can leverage machine learning and artificial intelligence techniques to analyze customer data and predict their preferences, enabling personalized recommendations.

RQ2: In what way can ING optimize its use of data to achieve their strategic objectives and improve business performance and possibly lock in untapped revenue potential?

To optimize the use of data, ING can implement advanced analytics techniques, such as predictive and prescriptive analytics, to uncover patterns and trends in consumer

behavior. This can help ING forecast which products or services a client is likely to be interested in and make data-driven decisions to maximize revenue.

Implementing a recommender system can also play a role in optimizing data usage. By analyzing customer data and preferences, ING can develop personalized recommendations for its clients, suggesting appropriate banking products that align with their needs and interests. This can enhance customer satisfaction, increase cross-selling and up-selling opportunities, and ultimately improve business performance.

Furthermore, ING can explore alternative recommendation algorithms and models, such as the LightFM model, XGBoost, and Decision Tree, to build a robust and effective recommender.

SQ1: Which recommendation algorithms are suitable for making recommendations on our case study?

Based on the results obtained from our experiments, we have successfully addressed sub-question 1, which aimed to identify suitable recommendation algorithms for our case study dataset. In our evaluation, we utilized the Area Under Curve (AUC) metric, which measures the probability of correctly ranking positive examples higher than negative examples. Our first experiment with the LightFM model yielded promising results, achieving a maximum AUC score of 84% (0.842), indicating the high accuracy of our models. Furthermore, experiment 2 revealed that the LightFM model trained with the logistic loss function outperformed other variations, suggesting its suitability for predicting user preferences based on previous interactions.

Moving on to the XGBoost model, after careful parameter tuning, we observed a higher AUC score on the testing set compared to the LightFM model. This suggests that XGBoost demonstrates superior performance in terms of prediction accuracy for our dataset. However, it is important to note that the LightFM model achieved a respectable AUC score of 0.519, indicating its competence in making accurate recommendations.

In contrast, our evaluation of the Sklearn Decision Tree Model showcased a higher Train AUC score but a lower Test AUC score compared to LightFM. This suggests that the Decision Tree Model might be overfitting the training data and struggling to generalize well to new instances. Consequently, despite its higher Train AUC score, LightFM outperforms the Decision Tree Model in terms of Test AUC score,

highlighting its superiority in making accurate recommendations.

SQ2: How is the performance of our chosen model in building a recommendation system for the case study?

Based on the results obtained from our experiments, it can be concluded that LightFM is an effective algorithm for building a recommendation system. The evaluation of LightFM models using the AUC metric demonstrated high accuracy in predicting user preferences and generating relevant recommendations. In Experiment 1, the best AUC score achieved was 84%, indicating the strong performance of the models. Furthermore, in Experiment 2, the model trained with the logistic loss function outperformed other variations, suggesting that this choice of loss function enhances the predictive capabilities of LightFM.

SQ3: What is the impact of the diverse features in ING's payment data on the performance of our chosen model?

Based on the results obtained from our experiments, the impact of different features on the performance of recommendation algorithms, as measured by the AUC metric, appears to be moderate. In our investigation, we focused on adding features related to transaction amounts, such as defining ranges and assigning transactions to these ranges based on their amounts. The inclusion of this feature showed mixed results across the recommendation algorithms tested. In Experiment 1, the BPR algorithm exhibited slight improvements on both the training and test data when the range of transaction amounts was considered as a user feature. However, the logistic and WARP algorithms showed slightly worse performance on the test data when this feature was added. Similar trends were observed in Experiment 2. These findings suggest that while the inclusion of transaction amount ranges may have some impact on the model's performance, it does not consistently lead to significant improvements. Therefore, further exploration and incorporation of other features, such as currency, country, and industry, may be necessary to enhance the recommendation system's performance and capture more relevant information about users and items.

These findings indicate that the effect of different features on model performance can vary depending on the specific recommendation algorithm employed.

SQ4: How does our chosen model compare with two selected alternative models for building a recommendation system?

Based on the results and analysis conducted in this master thesis, we have identified several alternative models that can be considered for building a recommendation system. The first model, LightFM, demonstrated high accuracy and performance in our experiments, achieving an AUC score of 0.842. LightFM is a hybrid recommender system that incorporates both user and item features, making it suitable for capturing complex patterns and providing personalized recommendations. However, it is important to note that the logistic loss function showed the best performance in predicting user preferences in the LightFM model.

The second model, XGBoost, also exhibited promising results with a higher AUC score of 0.627 on the testing set compared to LightFM's 0.519. XGBoost is a powerful gradient-boosting algorithm that can handle large and complex datasets. It leverages an ensemble of decision trees to make accurate predictions and has the flexibility to handle various features and optimize performance through hyperparameter tuning.

Lastly, the Sklearn Decision Tree Model, though not performing as well as LightFM and XGBoost in terms of AUC scores, offers a rule-based approach to making recommendations. It achieved a Train AUC score of 0.658, indicating its ability to capture patterns in the training data. However, it should be noted that its Test AUC score was lower at 0.430, suggesting a potential issue of overfitting.

In conclusion, the alternative models examined in this research project, namely LightFM, XGBoost, and the Sklearn Decision Tree Model, each possess unique strengths and limitations. LightFM excels in capturing complex patterns and providing accurate recommendations, while XGBoost offers powerful gradient-boosting capabilities for handling large datasets. The Sklearn Decision Tree Model, despite the lower performance, follows a rule-based approach. The choice of an alternative model for building a recommendation system depends on the specific requirements of the application, such as the nature of the dataset, available features, and desired interpretability versus accuracy trade-offs. Further exploration and experimentation with other alternative models can provide additional insights and help identify the most suitable model for a given recommendation system.

6.2 Limitation

While this master thesis has provided valuable insights into the different recommendation algorithms and the impact of various features on model performance, there are certain limitations that should be acknowledged.

- One of the limitations of this thesis is the lack of a domain expert who possesses extensive knowledge of this particular data and its corresponding area. Without a domain expert's input, there may be challenges in fully understanding. Having a domain expert could provide valuable insights and ensure a more comprehensive understanding of the data and its context, ultimately enhancing the quality and reliability of the findings.
- The research project focused on a specific case study dataset, which may limit the generalizability of the findings to other datasets.
- The master thesis primarily evaluated the performance of the models using the AUC metric. While AUC provides a measure of ranking accuracy, it may not capture the full picture of a recommendation system's performance. Additional evaluation metrics, such as precision, recall, or user satisfaction measures, could provide a more comprehensive understanding of the models' performance and user experience.
- Another limitation is the exclusion of certain features that could potentially impact the recommendation quality. Incorporating these additional features could lead to improved recommendations and enhance the overall performance of the models.
- AB tests need to be conducted to determine the success of the models. Nevertheless, there are always limitations to the present state of recommender systems that can be researched and addressed. Thus, more research is necessary to identify and address such limitations.
- Lastly, the master thesis focused on a comparison of three specific models: LightFM, XGBoost, and the Sklearn Decision Tree Model. While these models were selected based on their relevance and popularity, there are numerous other recommendation algorithms available that were not considered.

Considering these limitations, future research (Section 6.4) can address these issues and further enhance our understanding of recommendation algorithms and their application in different domains.

6.3 Practical Implications

The implications to practitioners in the thesis would include the following:

- Adoption of Recommender Systems: Practitioners in the banking industry, such as ING, can consider adopting recommender systems to enhance their customer experience and improve business performance. The literature review (Chapter 2) highlighted the potential benefits of recommender systems in the banking sector.
- Integration of Predictive and Prescriptive Analytics: Practitioners can recognize the importance of integrating predictive and prescriptive analytics in their data-driven decision-making processes. By leveraging advanced analytics techniques, practitioners can uncover patterns in consumer behavior, forecast future outcomes, and make optimized decisions to achieve strategic objectives and maximize revenue.
- Evaluation of Recommendation Algorithms: Practitioners should consider evaluating and comparing different recommendation algorithms to identify the most suitable approach for their specific context. The experiments conducted in the thesis, using models like LightFM, XGBoost, and Decision Tree, provide insights into the performance and effectiveness of these algorithms in building a recommendation system. This evaluation can guide practitioners in selecting the most appropriate algorithm to implement in their organization.
- Importance of Data Optimization: The thesis highlights the importance of optimizing the use of data to drive business outcomes. Practitioners should focus on collecting, managing, and analyzing data to uncover valuable insights. By leveraging data analytics and recommender systems, practitioners can utilize customer data to generate personalized recommendations.

6.4 Future Work

There is a number of potential directions for future research to expand and enhance the work done, based on the findings and limitations of this research project.

- Firstly, future studies could investigate the performance of the recommendation systems on different datasets from diverse domains. This would help validate the generalizability of the models and their performance across various contexts. Additionally, exploring datasets with different user preferences, and item categories can provide valuable insights into the models' adaptability and performance in real-world scenarios.
- Secondly, incorporating additional features that were not considered in this research project could be explored. Investigating the impact of these features on model performance can lead to more accurate and personalized recommendations.
- Furthermore, conducting a more comprehensive hyperparameter tuning process can be beneficial. Optimal hyperparameter configurations can significantly influence model performance. This can potentially improve the models' predictive power and recommendation accuracy.
- In addition to the recommendation systems evaluated in this master thesis, there are numerous other algorithms and techniques available in the field of recommender systems. Future work can explore and compare the performance of alternative models.

  By addressing these aspects in future work, researchers can contribute to the advancement of recommendation systems and improve their practical utility in various domains and applications.

Bibliography

[1] The data analytics platform - ING WB. April 24, 2023. URL: https://www.ingwb.com/en/insights/empowering-with-advanced-analytics/the-data-analytics-platform.

[2] ING wholesale banking in the netherlands. April 24, 2023. URL: https://www.ingwb.com/en/network/emea/netherlands.

[3] LightFM's documentation. URL: https://making.lyst.com/lightfm/docs/home.html.

[4] Ian F Alexander. A taxonomy of stakeholders: Human roles in system development. *International Journal of Technology and Human Interaction (IJTHI)*, 1(1):23–59, 2005. doi:10.4018/jthi.2005010102.

[5] Abbas Asosheha, Sanaz Bagherpour, and Nima Yahyapour. Extended acceptance models for recommender system adaption, case of retail and banking service in Iran. *WSEAS transactions on business and economics*, 5(5):189–200, 2008.

[6] Nikoletta Bozika. The data analytics platform: Towards data and analytics democratization. 2019.

[7] Faiza Allah Bukhsh, Zaharah Allah Bukhsh, and Maya Daneva. A systematic literature review on requirement prioritization techniques and their empirical evaluation. *Computer Standards & Interfaces*, 69:103389, 2020.

[8] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12:331–370, 2002.

[9] Robin Burke. Hybrid web recommender systems. *The adaptive web: methods and strategies of web personalization*, pages 377–408, 2007.

[10] Alexander Felfernig and Robin Burke. Constraint-based recommender systems: technologies and research issues. pages 1–10, 2008.

[11] Ebubeogu Amarachukwu Felix and Sai Peck Lee. Systematic literature review of preprocessing techniques for imbalanced data. *IET Software*, 13(6):479–496, 2019.

[12] Daniel Gallego and Gabriel Huecas. An empirical case of a context-aware mobile recommender system in a banking environment. pages 13–20, 2012.

[13] Daniel Gallego Vico, Gabriel Huecas Fernández Toribio, and Joaquín Salvachúa Rodríguez. Generating context-aware recommendations using banking data in a mobile recommender system. 2012.

[14] Amir Gershman, Amnon Meisels, Karl-Heinz Lüke, Lior Rokach, Alon Schclar, and Arnon Sturm. A decision tree based recommender system. *10th International Conferenceon Innovative Internet Community Systems (I2CS)–Jubilee Edition 2010–*, 2010.

[15] Jessica Hullman and Andrew Gelman. Designing for interactive exploratory data analysis requires theories of graphical inference. *Harvard Data Science Review*, 3(3), 2021.

[16] Andrew T Jebb, Scott Parrigon, and Sang Eun Woo. Exploratory data analysis as a foundation of inductive research. *Human Resource Management Review*, 27(2):265–276, 2017.

[17] Sangram Kapre. Common metrics to evaluate recommendation systems. 2021.

[18] Staffs Keele et al. Guidelines for performing systematic literature reviews in software engineering. 2007.

[19] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.

[20] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. pages 1748–1757, 2020.

[21] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*, 2015.

[22] Bradley N Miller, Joseph A Konstan, and John Riedl. Pocketlens: Toward a personal recommender system. *ACM Transactions on Information Systems (TOIS)*, 22(3):437–476, 2004.

[23] Sanghamitra Mitra, Nilendra Chaudhari, and Bipin Patwardhan. Leveraging hybrid recommendation system in insurance domain. *International Journal of Engineering and Computer Science*, 3(10):8988–8992, 2014.

[24] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and S. Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24:45–77, 01 2007. doi:https://doi.org/10.2753/MIS0742-1222240302.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[26] Baptiste Rocca. Introduction to recommender systems: Overview of some major recommendation algorithms. 2019. URL: https://towardsdatascience.com.

[27] Shruthi Sajid. A methodology to build interpretable machine learning models in organizations. Master's thesis, University of Twente, 2023.

[28] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. Setting goals and choosing metrics for recommender system evaluations. 23:53, 2011.

[29] Zeinab Shahbazi and Yung-Cheol Byun. Product recommendation based on content-based filtering using xgboost classifier. *Int. J. Adv. Sci. Technol*, 29:6979–6988, 2019.

[30] Bracha Shapira, Lior Rokach, and Francessco Ricci. Recommender systems handbook. 2022.

[31] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[32] Fu Jie Tey, Tin-Yu Wu, Chiao-Ling Lin, and Jiann-Liang Chen. Accuracy improvements for cold-start recommendation problem using indirect relations in social networks. *Journal of Big Data*, 8(1):1–18, 2021.

[33] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. 2011.

[34] Jason Weston, Hector Yee, and Ron J Weiss. Learning to rank recommendations with the k-order statistic loss. pages 245–248, 2013.

[35] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. Experimentation in software engineering. 2012.

[36] Nima Yahyapour. Determining factors affecting intention to adopt banking recommender system: case of iran. 2008.

[37] Xin Wayne Zhao, Yanwei Guo, Yulan He, Han Jiang, Yuexin Wu, and Xiaoming Li. We know what you want to buy: a demographic-based system for product recommendation on microblogs. pages 1935–1944, 2014.

7 Appendix

7.1 Appendix A

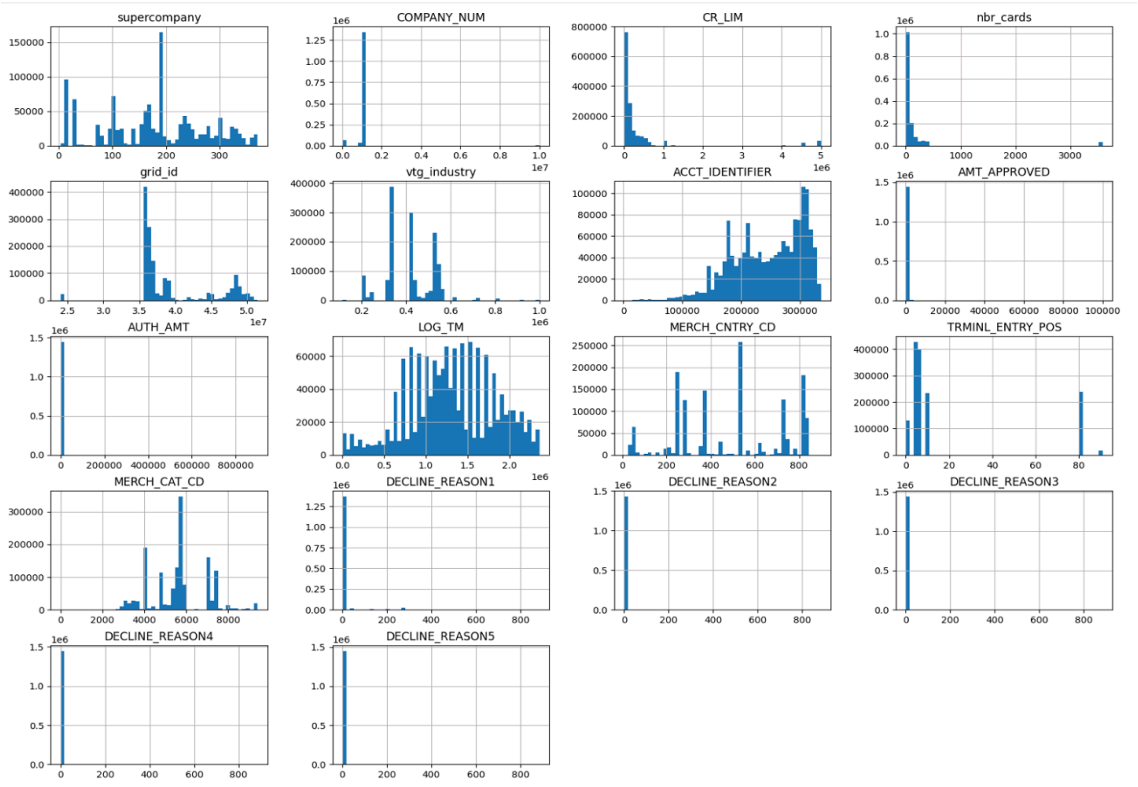


FIGURE 11: *Exploratory Data Analysis*

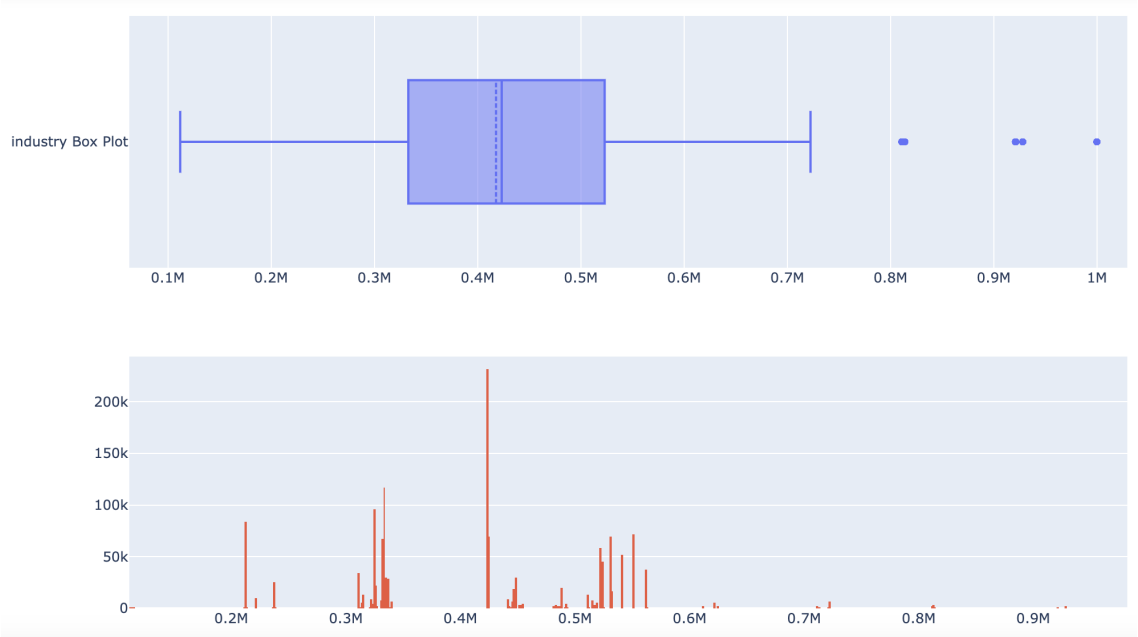


FIGURE 12: *Distribution of Industry*

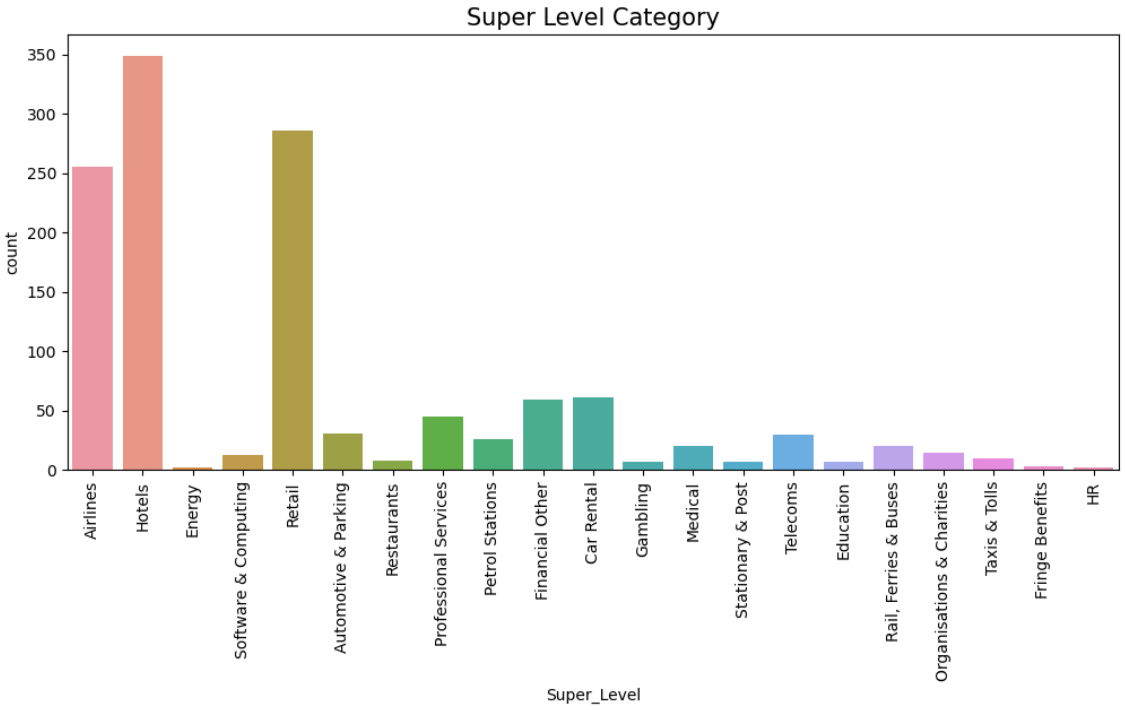


FIGURE 13: *Super Level Category*

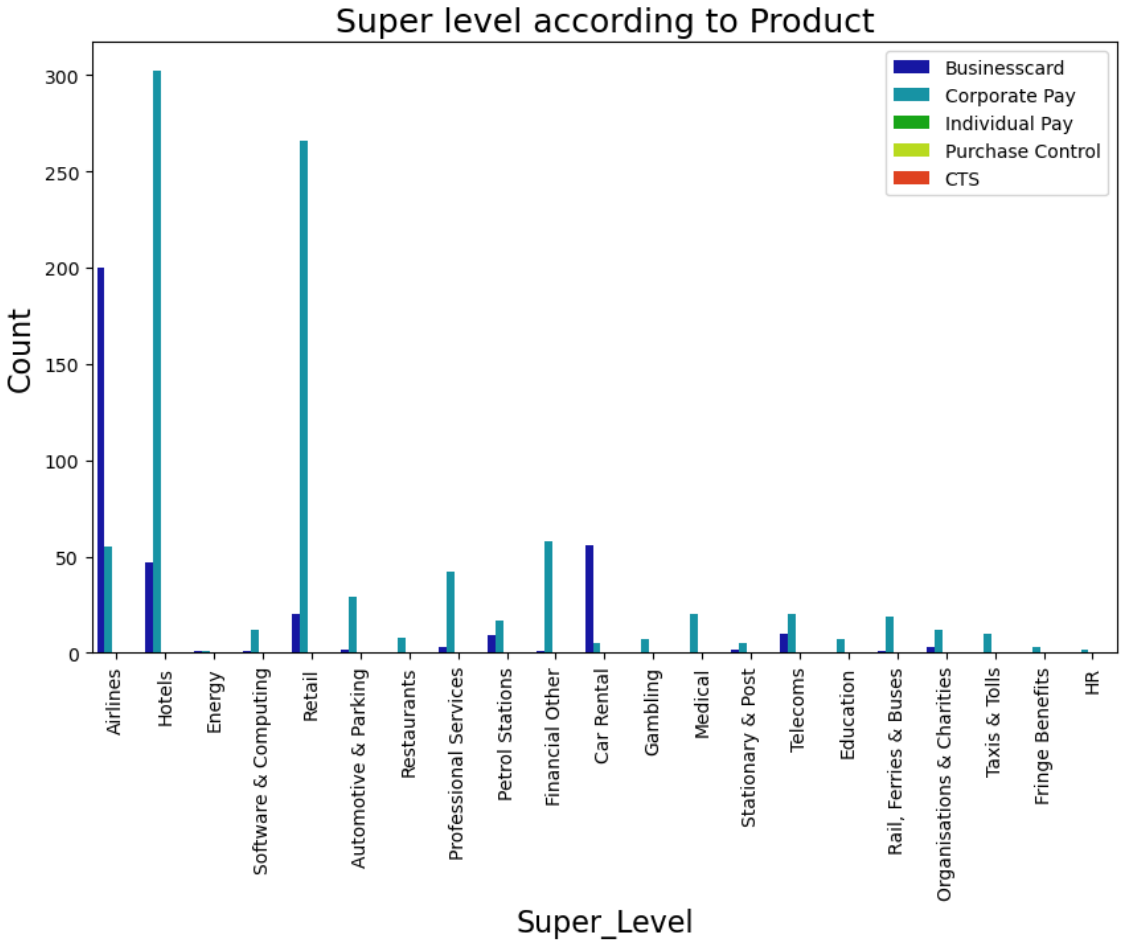


FIGURE 14: *Super Level According to Product*

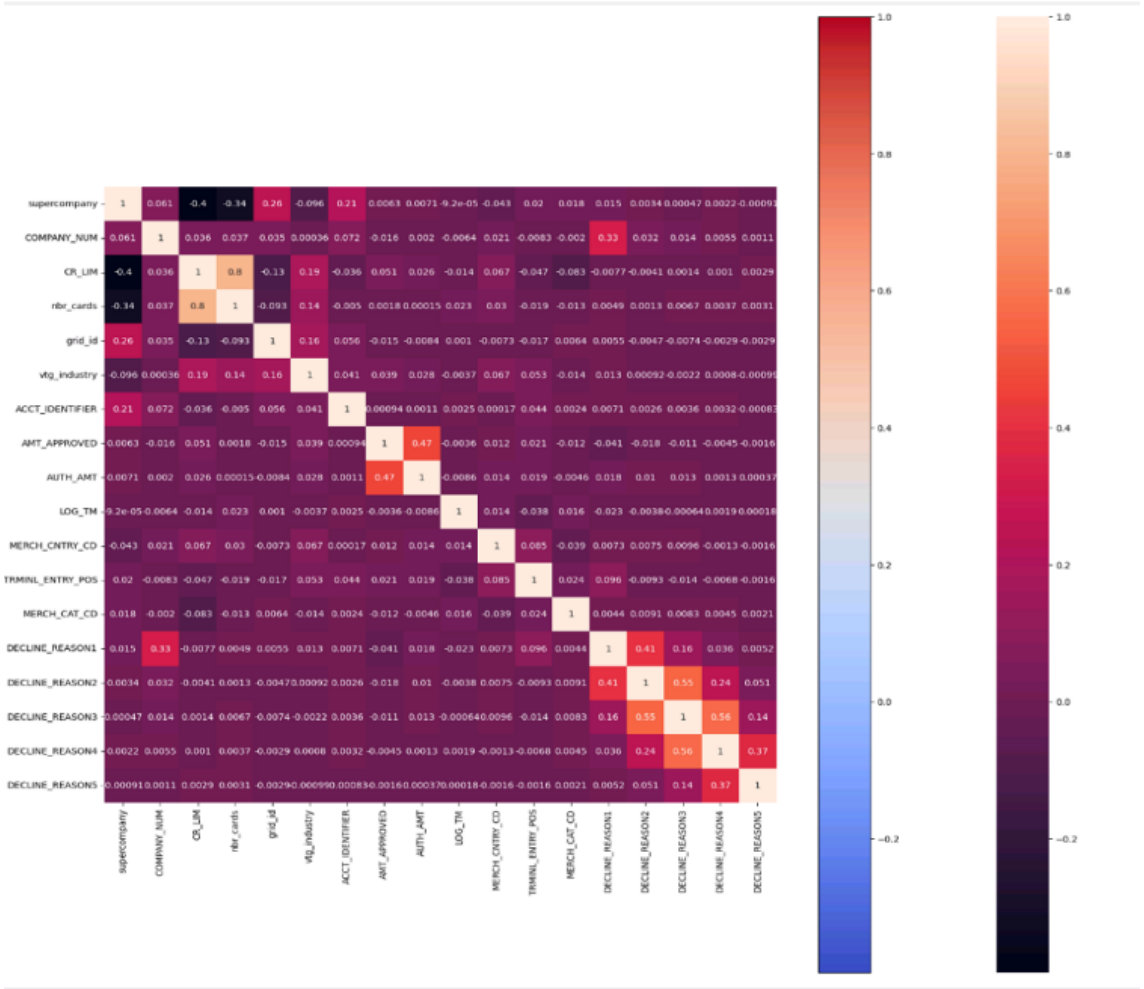


FIGURE 15: *Correlation*

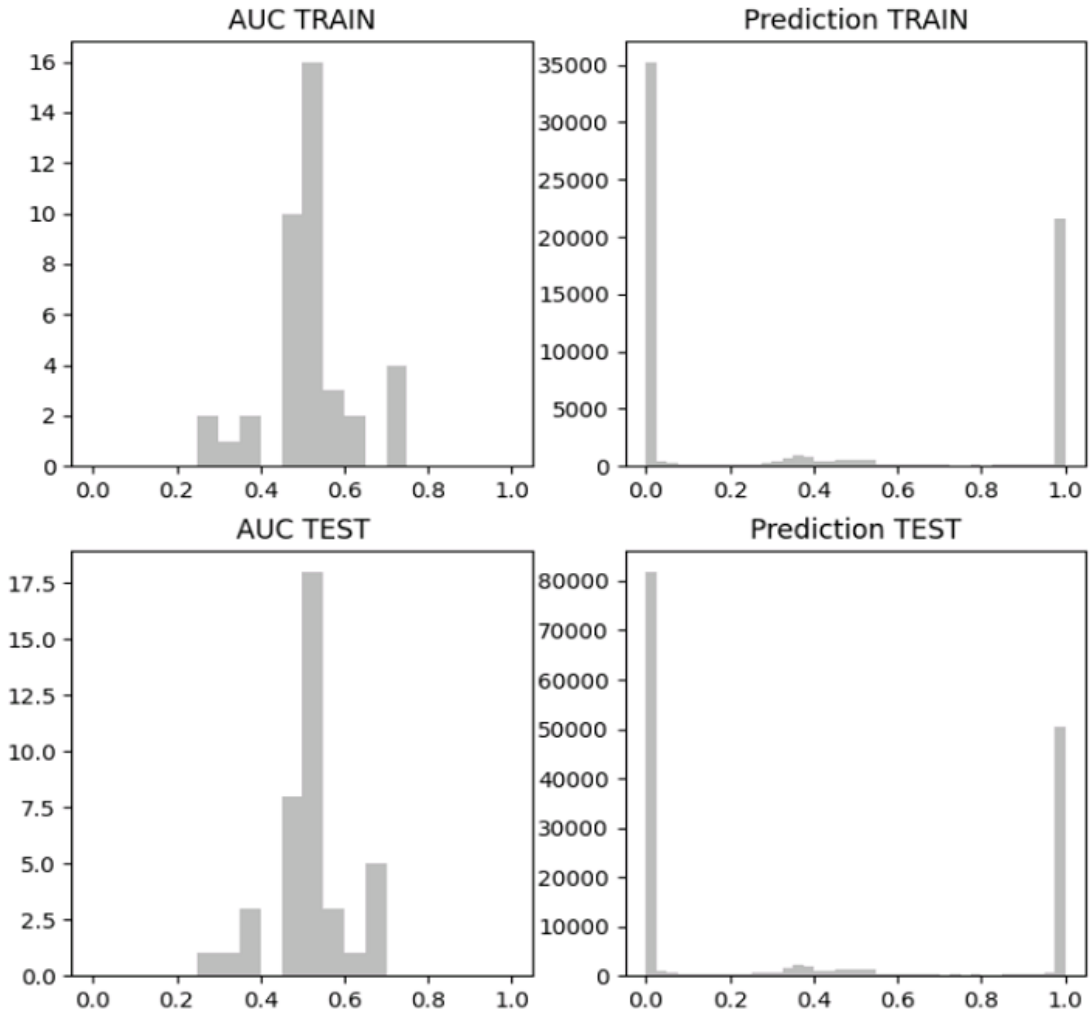


FIGURE 16: *AUC-LightFM model-Experiment 1*

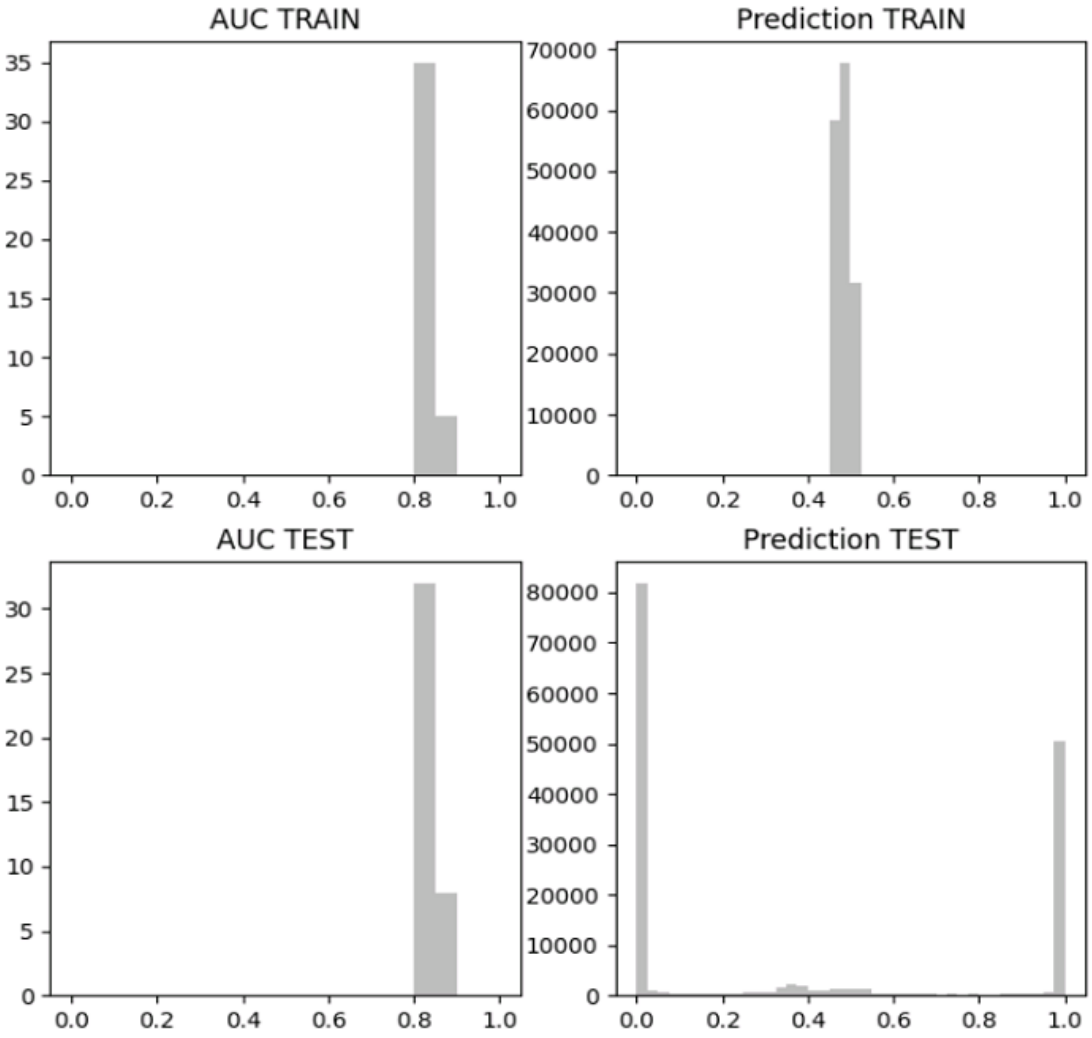


FIGURE 17: *AUC-LightFM model-Experiment 2*

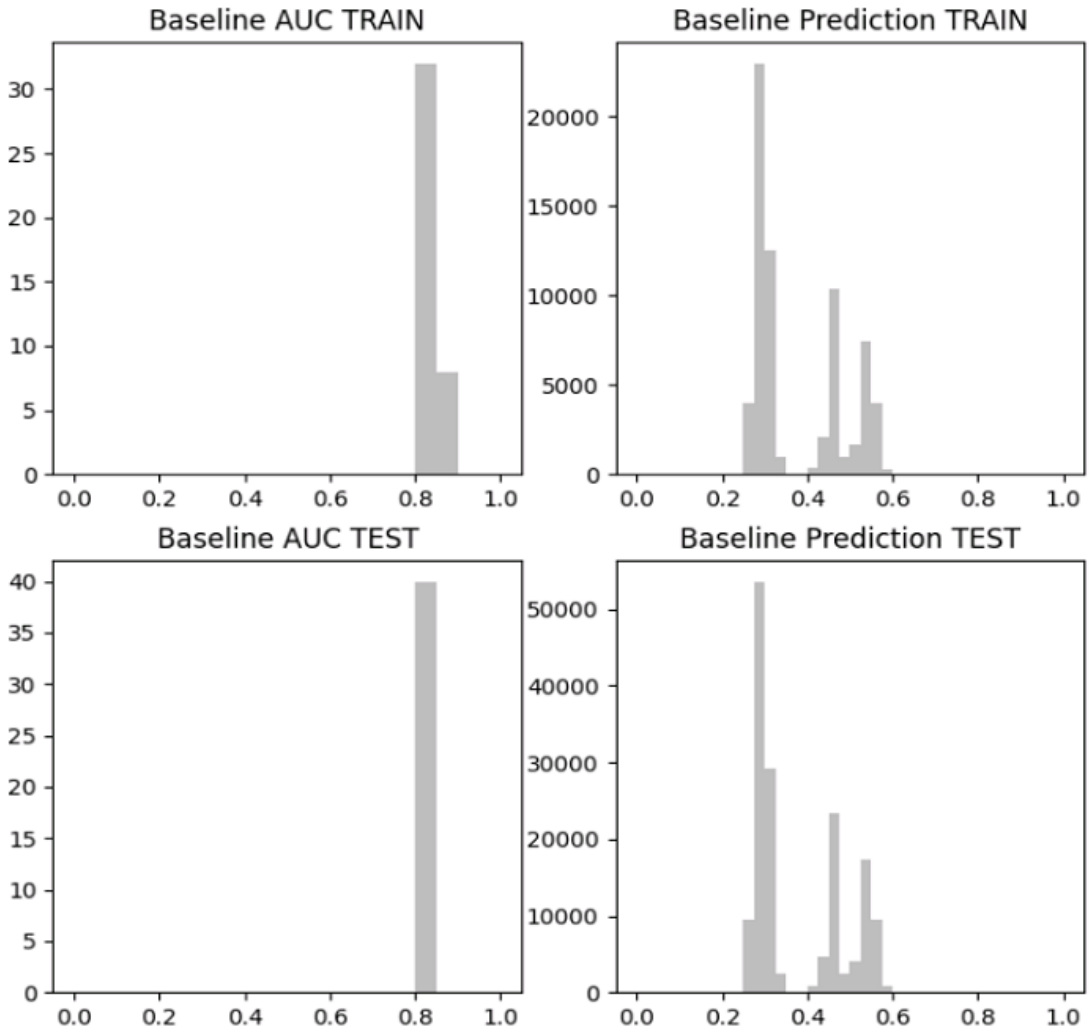


FIGURE 18: *AUC-Baseline model-Experiment 1*

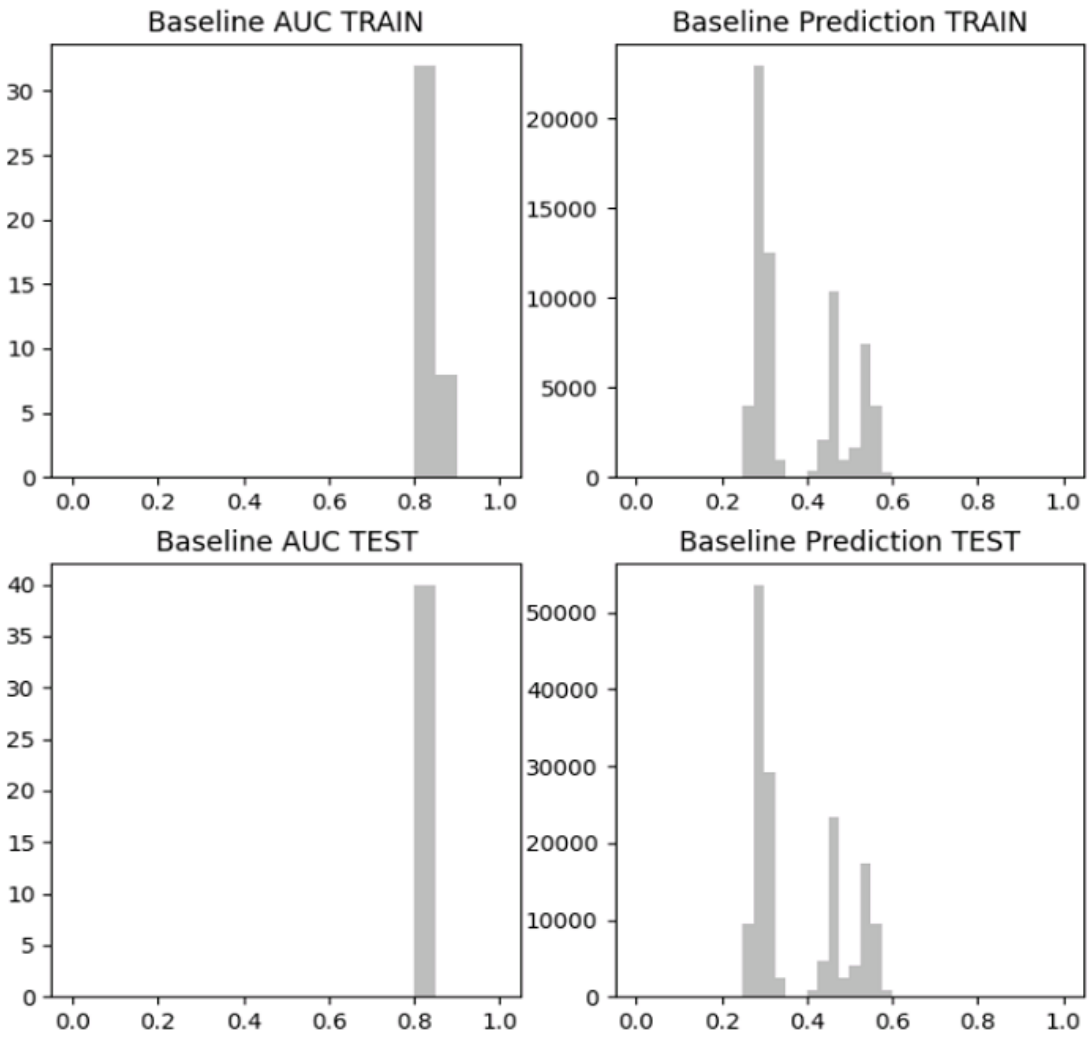


FIGURE 19: *AUC-Baseline model-Experiment 2*

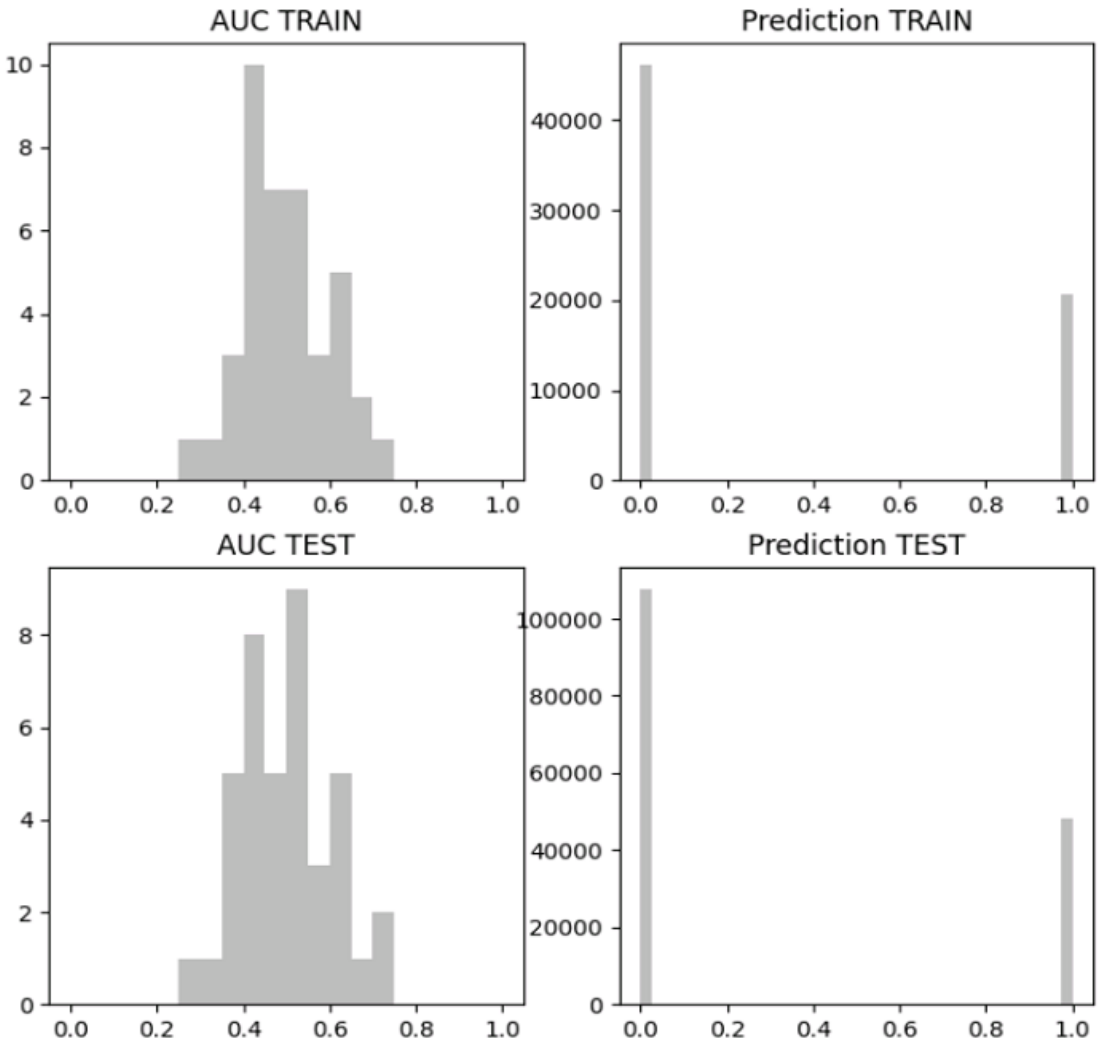


FIGURE 20: *AUC-LightFM model-Experiment 1 by Range of Amount*

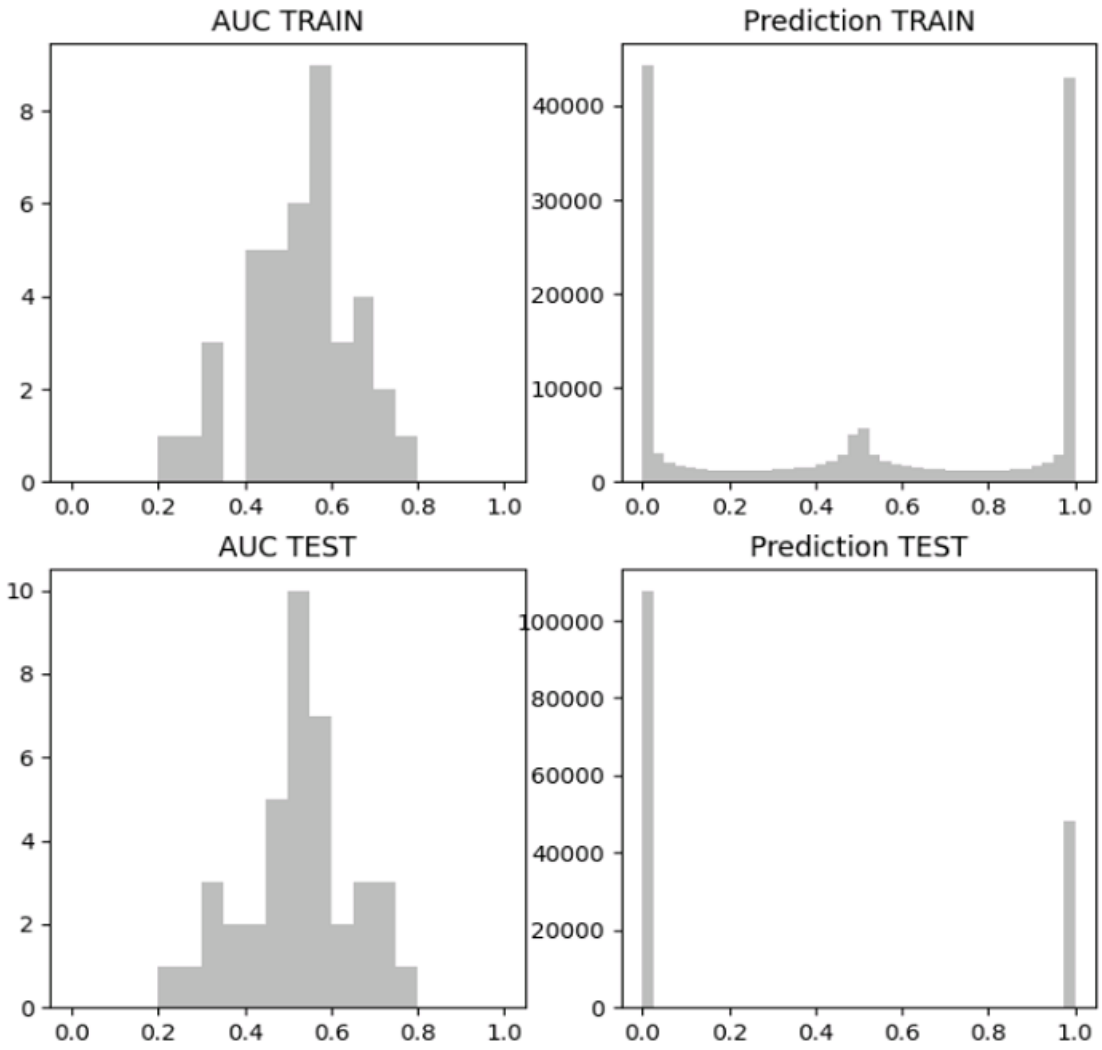


FIGURE 21: *AUC-LightFM model-Experiment 2 by Range of Amount*

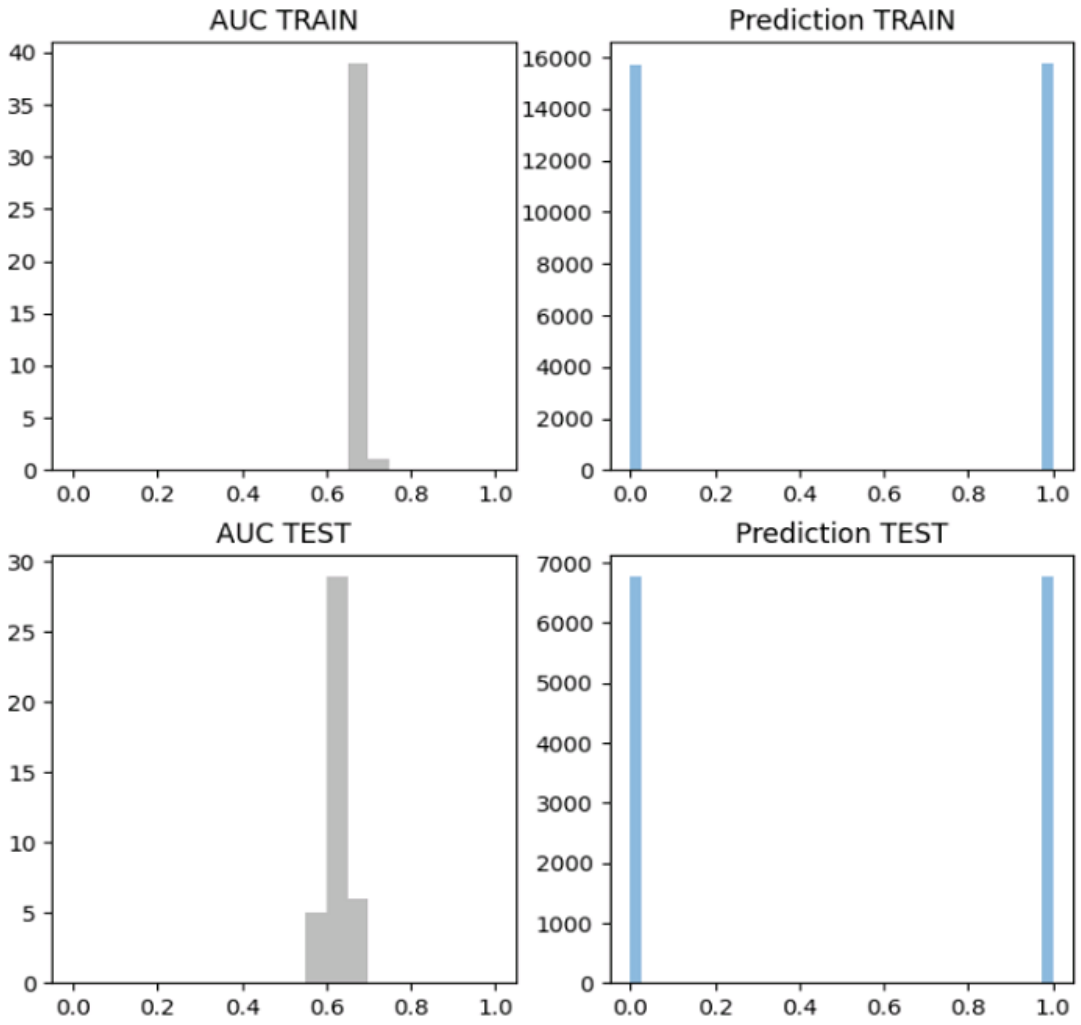


FIGURE 22: *AUC-XGBoost*

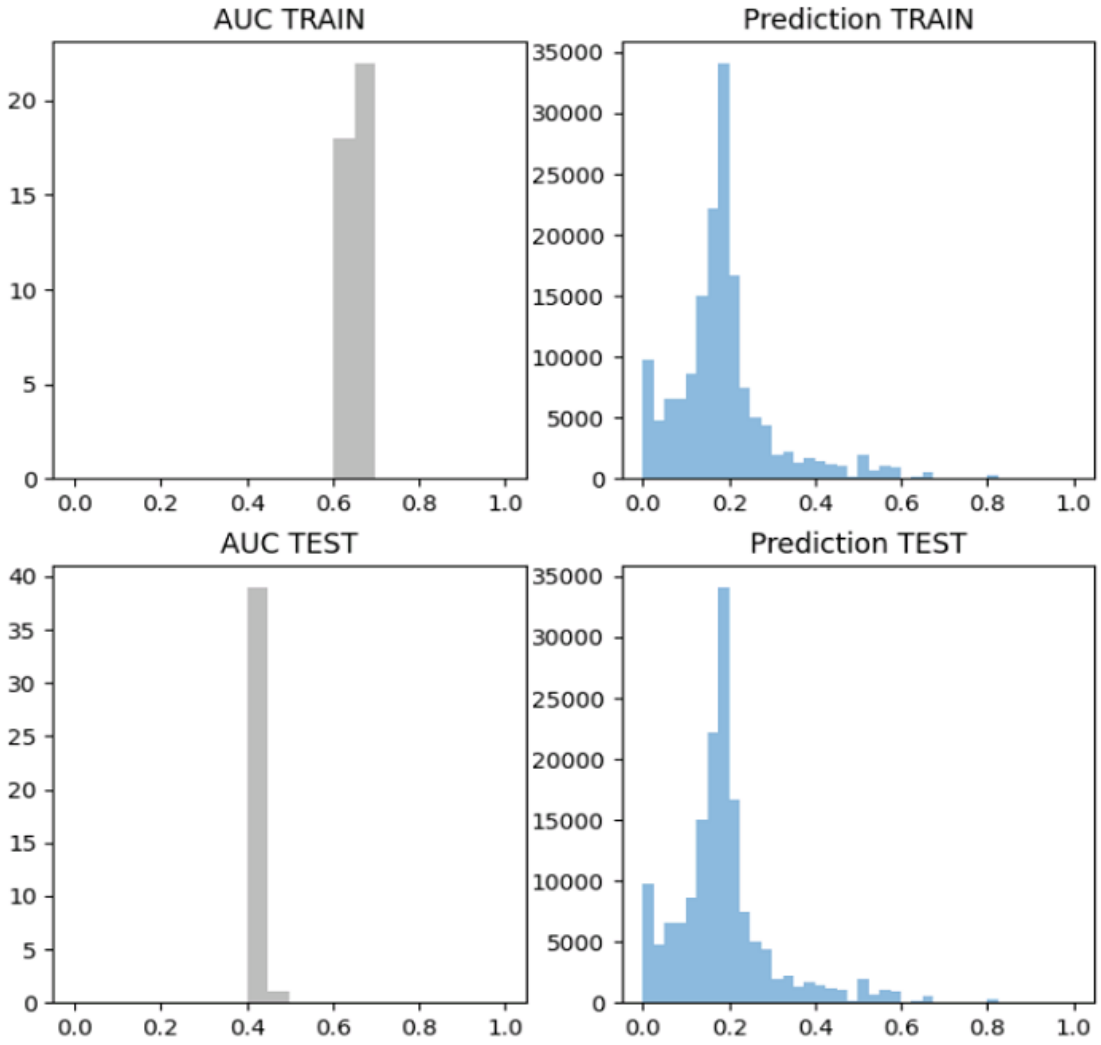


FIGURE 23: *AUC-Sklearn Decision Tree*

7.2 Appendix B

LightFM parameters tuning:

```
params = {'loss' : 'warp/bpr/logistic',
'no_components' : 1,
'learning_rate' : 0.01,
'user_alpha' : 1
}
```

XGBoost parameters tuning:

```
params = {'objective' : 'binary:logistic',
'eval_metric' : 'auc',
'eta' : 0.01,
'max_depth' : 4,
'lambda' : 2,
'alpha' : 0.4,
'nthread' : 5,
'n_estimators' : 7,
'subsample' : 0.7,
'colsample_bytree' : 1
}
```

Decision Tree parameters tuning:

```
criterion = "gini",
splitter = "best",
max_depth = 50,
min_samples_split = 15,
min_samples_leaf = 5,
max_features = 7,
random_state = 20,
max_leaf_nodes = 44
```