République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Monastir
Institut Supérieur d'Informatique de Mahdia
MEMOIRE
N° d'ordre: GLSI-3-24-77

THESIS

Presented to:

Higher Institute of Computer Science of Mahdia

Carried out at:

PerpetualCode

In partial fulfillment of the requirements for the degree of:

Licence of Computer Science:Génie Logiciel et Système d'Information

Elaborated by:

HASSINE MOHAMED ANOUAR

Development of a Product Recommendation System for an E-commerce Platform

Defended on: 09/09/2024, before the jury composed of:

President:	**Bsir Bassem**
Academic Supervisor:	**Ahmed Afef**
Reviewer:	**Zribi Ines**
Professional Supervisor:	**Anouar Kallel**

Academic Year: 2023-2024

Acknowledgements

At the end of this project, I would like to express my sincere gratitude to all the people who played a key role in its success.

First and foremost, my thanks go to the teachers at the ISI Mahdia. Their dedication and constant support have been the cornerstone of this achievement.

I also wish to express my gratitude to Ms. Afef Ahmed, my supervisor. Her wise advice and attentive guidance have been invaluable.

I would like to warmly thank Mr. Kallel Anwer, the CTO at PerpetualCode. His unwavering support and the opportunity he provided me greatly contributed to the success of this project.

My thanks also go to the PerpertualCode team, including Mr. Barboura Amin. Their warm welcome and valuable collaboration were of great help.

A big thank you to Mr. Anouar Kallel and Mrs. Sarra Chaali for the trust they placed in my abilities. Their support was a crucial driving force in the accomplishment of this project.

I do not forget my family, my source of inspiration and constant support. I dedicate this success to my mother, Mrs. Boujday Najwa, and my father, Mr. Hassine Seddik.

Finally, I want to express my deep gratitude to the jury members. Their valuable evaluation was a determining factor in the validation of this final project.

Contents

List of figures

1 General Introduction

As part of earning my licence degree in Computer Science, specializing in Software Engineering and Information Systems at ISI Mahdia, I completed my end-of-studies internship with PerpetualCode, a leading firm in E-commerce, Digital Marketing, and Web Development. This internship was a valuable opportunity to bridge the gap between academic knowledge and practical application.

PerpetualCode, known for its expertise in innovative IT solutions and advanced mobile and web applications, provided the ideal environment for this project. Collaborating with them allowed me to apply the theoretical concepts from ISIMA in a real-world context, gaining practical experience and contributing to the development of cutting-edge solutions.

Our project represents a fusion between Artificial Intelligence (AI) and the field of E-commerce. This innovative approach aims to apply the knowledge acquired in AI to address specific challenges in the E-commerce domain.

In today's rapidly evolving digital landscape, e-commerce platforms face the challenge of offering personalized experiences to an increasingly diverse and demanding customer base. With the proliferation of online shopping options, businesses must go beyond traditional marketing strategies and leverage advanced technologies to cater to individual customer preferences. A well-implemented recommendation system can significantly enhance customer satisfaction, drive sales, and foster customer loyalty by providing tailored product suggestions.

The primary objective of this project is to develop an effective recommendation system for an e-commerce website, enabling it to offer personalized product recommendations to its users. This system is designed to understand customer preferences, predict future purchases, and ultimately contribute to the growth and success of the business.

To achieve this objective, the project was conducted in several key stages. Initially, a thorough understanding of the business mission and the e-commerce environment was established. This involved identifying the specific needs and challenges faced by the business, as well as understanding the customer behavior patterns that could inform the recommendation process.

Following this, the focus shifted to data exploration and visualization. Three datasets, comprising customer information, order details, and product attributes, were meticulously analyzed to uncover insights and patterns that could be leveraged in the recommendation process. This stage was crucial in identifying the relationships between various factors, such as customer demographics, purchasing behavior, and product characteristics.

With a solid understanding of the data, the next step was to preprocess the datasets

for modeling. This involved cleaning the data, handling missing values, and transforming the data into a suitable format for the various modeling techniques to be applied. The preprocessing stage was essential in ensuring the accuracy and effectiveness of the models.

The core of the project centered on the experimentation with different recommendation methods. Various approaches were explored, including content-based filtering, collaborative filtering (both user-based and item-based), and model-based methods such as Factorization Machines. Each method was meticulously evaluated using appropriate evaluation metrics to determine its performance in generating accurate and relevant recommendations.

Finally, after thorough evaluation, the best-performing approach was selected for implementation. This recommendation system was fine-tuned and deployed to provide personalized product suggestions, thereby enhancing the overall user experience on the e-commerce platform.

This project not only demonstrates the application of data science techniques to solve real-world business challenges but also highlights the importance of an iterative and systematic approach to model development. By continuously refining the models and incorporating feedback, the project successfully delivered a recommendation system that aligns with the business's mission and contributes to its long-term success.

2 General Context of the Project

In this first chapter, we will introduce the work environment, specifically the hosting organization. Next, we will outline the project's mission and the methodological framework used. Finally, we will provide a detailed explanation of the proposed solutions.

2.1 Host Agency

2.1.1 Introduction

The host company is a Tunisian-origin web development agency based in France, founded by Anouar Kallel and Amin Barboura. Renowned for its expertise, the company specializes in web development with a range of technologies, including CMS platforms, React, and Angular, as well as mobile application development using Flutter. In addition to their development services, they offer SEO, SEA, and comprehensive Google Analytics solutions.

Figure 2.1: PerpetualCode

The professional team made a significant impression with their guidance and support throughout my journey, keeping me motivated to enhance both my skills and the project. They provided valuable insights into professional conduct and development.

2.1.2 Vision

"To be the go-to partner for creating sustainable and innovative digital solutions, shaping the future of the web and mobile applications with excellence and creativity."

2.1.3 Mission

"At Perpetual Code, our mission is to deliver tailor-made web and mobile development solutions that meet our clients' specific needs. By combining technical expertise, a user-centered approach, and continuous innovation, we create high-performance and intuitive applications that drive our clients' growth and help them achieve their business goals."

2.1.4 Values

Figure 2.2: PerpetualCode-values

1. **Continuous Innovation**: We are committed to staying at the forefront of the latest technologies and trends to offer innovative solutions that anticipate our clients' future needs.
2. **Technical Excellence**: We are engaged to deliver the highest quality solutions by employing best development practices and the most recent technologies.
3. **Client-Centric Approach**: We place our clients' needs and goals at the heart of our development process, ensuring customized solutions that maximize their satisfaction and success.
4. **Active Collaboration**: We believe in the power of teamwork, both internally and with our clients, fostering open communication and encouraging proactive collaboration to achieve the best results.

5. **Agility and Flexibility**: We are always ready to adapt to changes and quickly progress according to project needs and market trends to provide flexible and scalable solutions.
6. **Integrity and Transparency**: We act in honest and transparent ways in all our interactions, providing clear information and being accountable for our commitments.
7. **Exceptional Client Service**: We strive to exceed our clients' expectations by delivering outstanding customer service, being responsive to their needs, and offering continuous support throughout the project lifecycle.

2.2 Introduction to E-commerce

2.2.1 Overview of E-commerce

E-commerce, or electronic commerce, refers to the buying and selling of goods and services over the internet. This sector encompasses a wide range of online transactions, from retail to wholesale, including digital services and subscriptions. The rapid development of e-commerce has revolutionized the way businesses interact with their customers and has significantly altered traditional market dynamics.

2.2.2 Evolution of E-Commerce

Since its inception in the 1990s, e-commerce has undergone significant evolution. Initially limited to simple transactions, it quickly integrated advanced technologies such as mobile commerce (m-commerce), social commerce (s-commerce), and personalized online shopping. Continuous innovations in information technology, data analytics, and user interfaces have fueled this evolution, leading to a more dynamic and customer-centric e-commerce landscape.

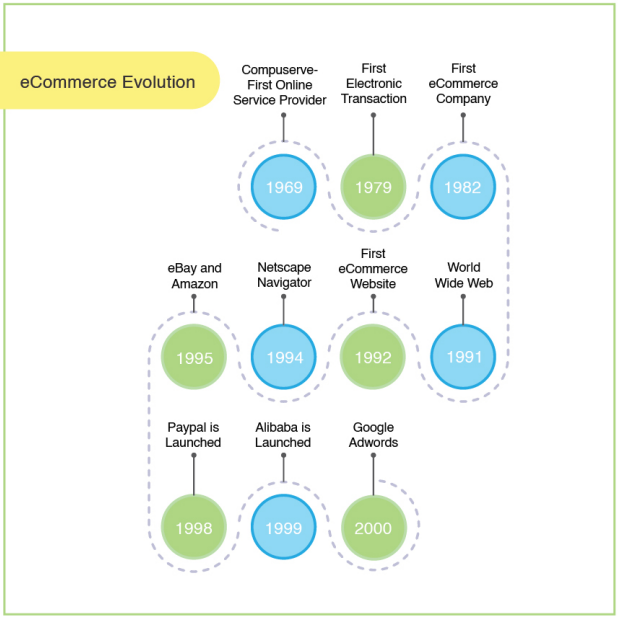


Figure 2.3: E-commerce evolution

2.2.3 The Importance of E-Commerce in Marketing and Sales

1. **Global Reach and Market Expansion** E-commerce enables businesses to reach a global audience without the geographical limitations of physical stores. This global reach allows companies to enter new markets, attract international customers, and diversify their revenue streams, which is crucial in an increasingly globalized market.
2. **Cost Efficiency** Online operations significantly reduce costs associated with physical stores, such as rent, utilities, and inventory management. These savings allow businesses to reallocate resources to other aspects of their marketing strategy, such as online advertising and enhancing the user experience.
3. **Data Analytics and Customer Insights** E-commerce platforms generate vast amounts of data on shopping behaviors, customer preferences, and market trends. Analyzing this data provides valuable insights that help businesses personalize their offerings, refine marketing campaigns, and optimize sales strategies.
4. **Enhanced Customer Experience** E-commerce offers consumers unparalleled flexibility, allowing them to shop anytime and from anywhere. Features such as personalized recommendations, multiple payment options, and accessible online customer service enhance the overall shopping experience, leading to increased customer satisfaction.
5. **Integration with Digital Marketing Strategies** E-commerce is closely integrated with digital marketing strategies. Techniques such as search engine optimization (SEO), social media marketing, email campaigns, and pay-per-click (PPC) advertising can be seamlessly integrated with e-commerce platforms to drive traffic, increase conversions, and strengthen brand visibility.

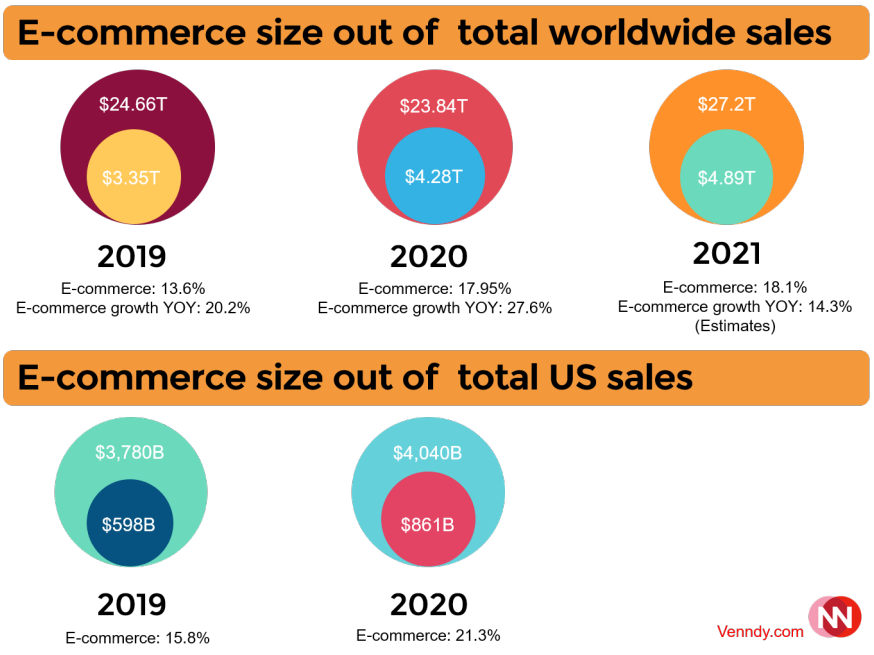


Figure 2.4: E-commerce size worldwide

2.3 Introduction to the Field of Artificial Intelligence (AI) and Recommendation Systems

2.3.1 Definition of Artificial Intelligence

Artificial Intelligence is a set of theories that create systems capable of exhibiting human-like behaviors and reasoning. It also allows systems to mimic human intelligence. AI is divided into several domains, such as Machine Learning and Deep Learning. (1, netapp)

2.3.2 Definition of Machine Learning

Machine Learning is a set of techniques and methods that enable computers to learn from data and make decisions based on models such as logistic regression, random forests, KNN, and SVM. There are thousands of algorithms, each based on a specific approach. Three approaches are commonly used in Machine Learning: a supervised approach means that the data is labeled during the model's training. On the other hand, unsupervised learning trains models from unlabeled data. Semi-supervised learning, meanwhile, requires both approaches to solve certain types of problems. (2, talend)

2.3.3 Definition of Deep Learning

Deep Learning is a branch of Artificial Intelligence inspired by the functioning of the human brain, allowing computers to make decisions. The goal is to solve increasingly complex problems. (3, datascientest)

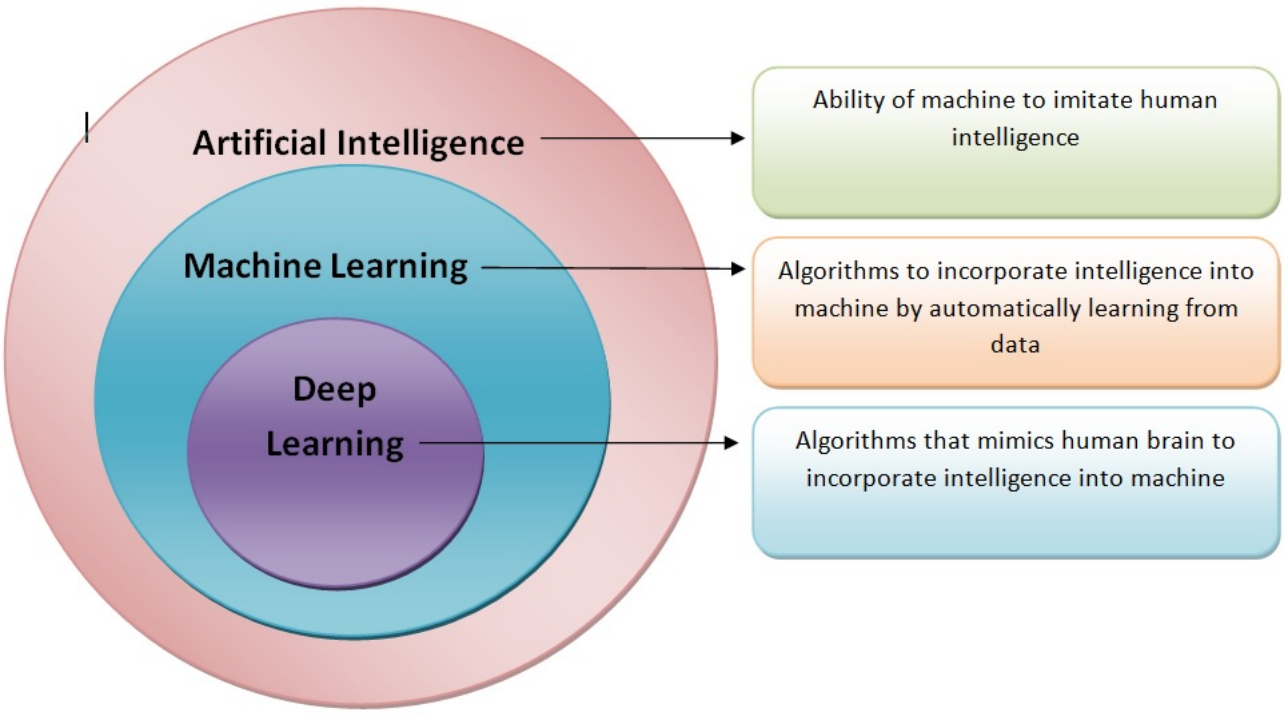


Figure 2.5: AI, ML and DL

2.3.4 Overview of Recommendation Systems

Recommendation systems (also known as a "personalized suggestion engine") have become increasingly prevalent in our daily lives with the advent of websites like YouTube, Amazon, and Netflix .

A recommendation system (or recommender system) is a class of machine learning, these systems are trained to understand the preferences, previous decisions, and characteristics of people and products using data gathered about their interactions. These include impressions, clicks, likes, and purchases. Because of their capability to predict consumer interests and desires on a highly personalized level, recommender systems are a favorite with content and product providers. They can drive consumers to just about any product or service that interests them, from books to videos to health classes to clothing.

2.3.5 Evolution of Recommendation Systems

Recommender systems aren't a new idea. Jussi Karlgren formulated the idea of a recommender system, or a "digital bookshelf," in 1990. Over the next two decades researchers at MIT and Bellcore steadily advanced the technique.(4, Nvidea)

The technology really caught the popular imagination starting in 2007, when Netflix — then in the business of renting out DVDs through the mail — kicked off an open competition with a $1 million prize for a collaborative filtering algorithm that could improve on the accuracy of Netflix's own system by more than 10 percent, a prize that was claimed in 2009.

Over the following decade, such recommender systems would become critical to the success of Internet companies such as Netflix, Amazon, Facebook, Baidu and Alibaba.

2.3.6 The Importance of Recommendation Systems in E-commerce

A recommendation system is crucial for enhancing user experience and driving sales. The effectiveness of such a system can significantly impact customer satisfaction and conversion rates. Research shows that a well-designed recommendation system can increase engagement and sales by providing users with tailored product suggestions that match their preferences and behavior.

In other words, implementing an effective recommendation system can be a valuable investment for businesses, as it improves the likelihood of customers finding the products they want, thereby enhancing their overall shopping experience and boosting the company's revenue. This is achieved through personalized recommendations that reduce search time, increase product relevance, and ultimately contribute to higher customer satisfaction and loyalty.

2.3.7 Future Trends in Recommendation Systems

As recommendation engines advance, we may see some key developments:

More contextual recommendations based on detailed customer data and behaviors. Increased use of deep learning and neural networks for sophisticated recommendations.

Expanded hybrid approaches that blend various data signals. Tighter integration between recommendations and business KPIs like conversion rate. More focus on explainability and transparency behind recommendations. These trends point to more intelligent, personalized, and impactful future recommendation systems. However, core machine learning fundamentals will likely remain relevant.

So how does this technology work?

Collecting Information

Recommenders work by collecting information — by noting what you ask for — such as what movies you tell your video streaming app you want to see, ratings and reviews you've submitted, purchases you've made, and other actions you've taken in the past

Perhaps more importantly, they can keep track of choices you've made: what you click on and how you navigate. How long you watch a particular movie, for example. Or which ads you click on or which friends you interact with.

All this information is streamed into vast data centers and compiled into complex, multidimensional tables that quickly balloon in size.

They can be hundreds of terabytes large — and they're growing all the time.

That's not so much because vast amounts of data are collected from any one individual, but because a little bit of data is collected from so many.

In other words, these tables are sparse — most of the information most of these services have on most of us for most of these categories is zero.

But, collectively these tables contain a great deal of information on the preferences of a large number of people.

And that helps companies make intelligent decisions about what certain types of users might like.

Content Filtering, Collaborative Filtering

While there are a vast number of recommender algorithms and techniques, most fall into one of two broad categories: collaborative filtering and content filtering.

Collaborative filtering helps you find what you like by looking for users who are similar to you.

So while the recommender system may not know anything about your taste in music, if it knows you and another user share similar taste in books, it might recommend a song to you that it knows this other user already likes.

Content filtering, by contrast, works by understanding the underlying features of each product.

So if a recommender sees you liked the movies "You've Got Mail" and "Sleepless in Seattle," it might recommend another movie to you starring Tom Hanks and Meg Ryan, such as "Joe Versus the Volcano."

Those are extremely simplistic examples, to be sure.

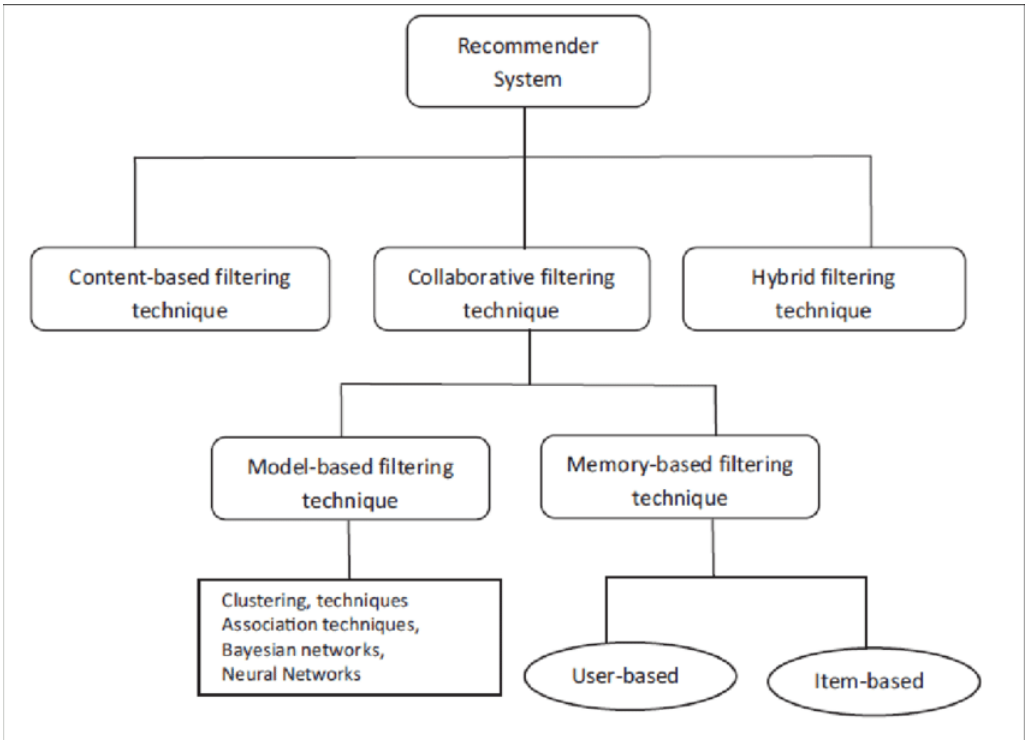


Figure 2.6: The-main-types-of-recommendation-system

3 Project Definition

3.1 Problem Statement

In the rapidly evolving landscape of e-commerce, personalized user experiences have become a critical factor in retaining customers and driving sales. Traditional recommendation systems often fall short in addressing the complex and dynamic nature of user preferences, particularly in scenarios where vast product inventories and diverse customer bases are involved.

The e-commerce platform under study, developed by PerpetualCode, has amassed a substantial dataset comprising customer profiles, order histories, and detailed product information. However, the platform currently lacks an effective recommendation system capable of leveraging this data to provide personalized product suggestions. This gap results in missed opportunities for enhancing user engagement and optimizing sales, as customers are not presented with tailored recommendations that align with their interests and purchasing behavior.

The core problem, therefore, is to develop a recommendation system that can efficiently analyze the available data and deliver relevant, personalized product suggestions to users. This system should improve the customer experience on the platform by understanding individual preferences, predicting future purchases, and ultimately contributing to the business's growth and success.

3.2 Project Objectives

The primary objective of this project is to design and implement a robust recommendation system for the e-commerce platform. This system aims to fulfill the following specific objectives:

1. **Data Integration and Preprocessing:** Integrate and preprocess data from multiple sources (customers, orders, and products) to ensure that the recommendation system can access and utilize relevant information effectively. This involves cleaning the data, handling missing values, and transforming features to be suitable for modeling.

2. **Recommendation Algorithm Development:** Explore and implement various recommendation algorithms, including content-based filtering, collaborative filtering (user-based and item-based), and model-based approaches such as Factorization Machines. The goal is to identify and implement the most effective method(s) for this specific e-commerce context.

3. **Model Evaluation and Optimization:** Evaluate the performance of different recommendation models using appropriate metrics such as precision, recall, and F1-score. The objective is to compare these methods, optimize them, and select the one that best balances accuracy with computational efficiency.
4. **Implementation of the Best Model:** Deploy the selected recommendation model within the e-commerce platform, ensuring it is scalable, efficient, and capable of delivering real-time product recommendations to users.
5. **Documentation and Reporting:** Document the entire process, including the methodologies used, the rationale behind model selection, the challenges encountered, and the results obtained. This documentation will serve as a guide for future improvements and as a final report for academic and professional purposes.

3.3 Scope of the Project

The scope of this project encompasses the entire lifecycle of developing a recommendation system, from initial data exploration to final model deployment. The following key areas define the boundaries of the project:

- **Data Sources:** The project utilizes data extracted from the Shopify platform, specifically focusing on three key datasets: customers, orders, and products. These datasets provide the foundational information required for building user profiles, understanding product characteristics, and analyzing purchasing behavior.
- **Modeling Techniques:** The project explores a range of recommendation techniques, including content-based filtering, collaborative filtering, and model-based methods such as Factorization Machines. However, the scope does not extend to more advanced methods such as deep learning-based recommenders, which are considered out of scope due to resource and time constraints.
- **Evaluation and Optimization:** The project includes a thorough evaluation of the implemented models using established metrics. The scope also covers the optimization of models to enhance performance, but it does not delve into continuous learning or online training approaches.
- **Deployment Considerations:** While the project aims to deploy the recommendation system within the e-commerce platform, the scope is limited to prototype deployment. Full-scale integration with live customer data and real-time processing is beyond the current project's scope and is intended for future work.
- **Business Impact:** The project's focus is on improving user engagement and sales through personalized recommendations. However, it does not include a detailed financial impact analysis or the integration of other business intelligence tools.
- **Documentation and Reporting:** Comprehensive documentation of the project process, findings, and results is included within the scope. This will ensure that the project's outcomes are clearly communicated and can be built upon in future work.

Conclusion: This section defines the problem addressed by the project, outlines the specific objectives that guide its development, and clarifies the boundaries within which the project will operate. By focusing on these elements, the project aims to create a practical and effective recommendation system tailored to the needs of the e-commerce platform, driving both user satisfaction and business success.

3.4 Methodological Framework

To execute and succeed in this project, we need to follow a methodology that will help us accomplish the project step by step in a simplified and structured manner. There are several methodologies for managing data science projects; we will study a few methods to choose the most suitable one.

3.4.1 SEMMA

SEMMA is an acronym that stands for Sample, Explore, Modify, Model, and Assess. It is a method used for carrying out industrial-type data science projects. This method simplifies the application of statistical exploration techniques, the selection and transformation of the most critical predictive variables, the modeling of variables to predict outcomes, and the validation of the model's accuracy.

It consists of 5 steps:

1. **Sample**: Selecting a sample of data for analysis.
2. **Explore**: Understanding the data, the relationships between variables, distribution, etc.
3. **Modify**: Preparing and cleaning the data, such as handling missing values and duplicates.
4. **Model**: Building models using machine learning.
5. **Assess**: Evaluating the models using metrics to choose the most effective one. (5, datascience)

3.4.2 Knowledge Discovery in Databases (KDD)

Knowledge Discovery in Databases (KDD) is a process aimed at extracting useful and meaningful knowledge from large amounts of data. It is a crucial step in the fields of data mining and machine learning. The KDD process generally includes several steps:

Data Selection: This step involves choosing relevant data from various data sources. It is important to select data that is likely to contain useful information for the analysis.

Data Preprocessing: Raw data may contain errors, missing values, or noise. Data preprocessing includes steps such as data cleaning, normalization, imputing missing values, and dimensionality reduction.

Data Exploration: At this stage, data exploration techniques are used to understand the structure of the data, identify trends, patterns, and relationships between variables.

Knowledge Extraction: This is the main step of KDD, where data mining techniques like clustering, classification, association, etc., are applied to extract knowledge from the data.

Model Evaluation: The models generated in the previous step are evaluated to determine their accuracy and relevance. This may involve using performance metrics and cross-validation techniques.

Interpretation and Visualization: The results of the data analysis are interpreted to derive actionable insights. Visualization techniques are often used to present the results in an understandable manner.

Knowledge Utilization: The knowledge extracted from the data is used to make decisions, solve problems, or improve processes in the specific application domain.

Feedback: The KDD process is iterative. The knowledge obtained can be used to improve data quality, adjust models, and refine the analysis objectives. (geeksforgeeks, n.d.) Below is a legend that explains the KDD process:

3.4.3 CRISP-DM

Cross-Industry Standard Process for Data Mining is an approach used to manage data science projects. It is based on 6 key steps:

Understanding the Business Needs: This step aims to gain a general understanding of the business. A good understanding of the business is crucial for effectively using the data. This step is vital for understanding business keywords as well as the specific problem the solution is designed to address.

Understanding the Data: This step involves a thorough analysis of the available data. First, data needs to be collected. Second, it's important to understand the need and establish the format of the data, its nature, the size of the database, and the distribution values to use the data appropriately. Finally, perfect data rarely exists, so it's important to carefully examine the quality of the available data.

Data Preparation: This step involves the activities necessary to build an accurate dataset for analysis from raw data. Specifically, it involves classifying, sorting, cleaning, and especially recoding the data to make it compatible with the algorithms that will be used.

Modeling: This step involves choosing, configuring, testing, and validating different algorithms, as well as their logical sequence to build a model.

Evaluation: This step aims to verify whether the model has achieved its business objectives. It also plays a crucial role in deciding whether to deploy or improve the model.

Deployment: This is the final step of the CRISP process. It involves putting the models obtained into production for the end user, with the goal of presenting the knowledge gained through modeling in an appropriate form and integrating it into the decision-making process. (IBM, n.d.)

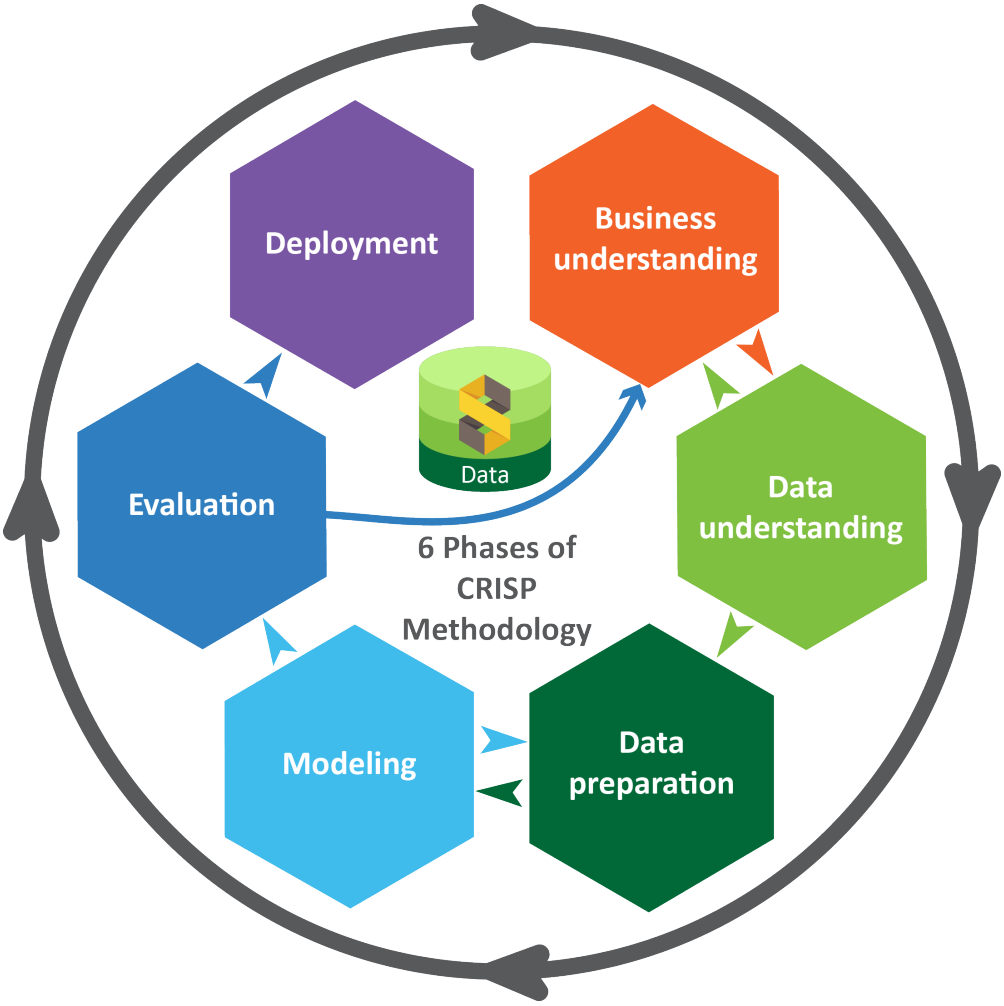


Figure 3.1: phases of CRISP-DM

3.4.4 Comparison: (KDD/SEMMA/CRISP-DM)

KDD and SEMMA are almost identical in that each step of KDD directly corresponds to a step in SEMMA. The CRISP-DM process combines the steps of selection and pre-processing (KDD) or sample exploration (SEMMA) with the data understanding step. It also incorporates business understanding and deployment steps. The table presents a comparative study between the three approaches:

A significant difference between CRISP-DM and the other two methodologies is that the transitions between the steps in CRISP-DM can be reversed. Additionally, CRISP-DM differs during data exploration by adding business understanding and deployment steps, making the process resemble product development more than research.

3.4.5 SCRUM

Scrum is an agile framework used for managing and executing projects, particularly in the fields of software development and product management. It is designed to help teams work collaboratively and adaptively, focusing on delivering high-value results incrementally. The Scrum framework is based on iterative development and is organized around several key components:

Sprint Planning: Each project is divided into time-boxed iterations called sprints, typically lasting two to four weeks. During sprint planning, the team selects and prioritizes tasks from the product backlog, setting clear objectives and defining the work to be accomplished in the upcoming sprint.

Daily Scrum: Also known as the daily stand-up, this is a brief meeting held each day of the sprint. Team members discuss their progress, upcoming tasks, and any obstacles they are facing, ensuring that the project stays on track and issues are addressed promptly.

Sprint Review: At the end of each sprint, a review meeting is held to showcase the completed work to stakeholders. This allows for feedback, ensures that the project is meeting its objectives, and provides an opportunity to make adjustments based on stakeholder input.

Sprint Retrospective: Following the sprint review, the team conducts a retrospective to reflect on the sprint process. This meeting focuses on identifying what went well, what could be improved, and how the team can enhance its practices in future sprints.

Product Backlog: This is a dynamic list of features, enhancements, and bug fixes required for the project. It is continually updated and prioritized by the product owner, ensuring that the team works on the most valuable tasks.

Sprint Backlog: Derived from the product backlog, this is a list of tasks selected for a specific sprint. The sprint backlog details the work to be completed within the sprint and is managed by the team.

Recommended Methodological Framework

The final decision is to apply the CRISP-DM methodology to the project. This approach was selected because of its structured and comprehensive nature, making it particularly well-suited for data science projects. CRISP-DM is an ideal fit for the objectives of this project, which center on developing a robust and effective recommendation system for an e-commerce platform.

Key reasons for choosing CRISP-DM:

Structured Approach: CRISP-DM provides a well-defined process that covers the entire lifecycle of a data science project, from understanding the business problem to deploying the final solution. This ensures that each phase of the project is systematically addressed, leading to more reliable and actionable results.

Flexibility and Iteration: CRISP-DM's iterative nature allows for flexibility in revisiting and refining each phase as needed. This is particularly important for a project of this scope, where understanding customer preferences and predicting future purchases require ongoing analysis and adjustment.

Alignment with Project Objectives: The CRISP-DM framework aligns well with the goal of building a recommendation system that can drive business growth. By following this methodology, we ensure that the project remains focused on delivering a solution that is both technically sound and aligned with the business needs.

Comprehensive Process: CRISP-DM's inclusion of steps such as business understanding, data preparation, modeling, and evaluation ensures that the project addresses

all critical aspects of developing a recommendation system. This comprehensive approach minimizes the risk of overlooking important factors that could impact the system's performance.

Continuous Improvement: The CRISP-DM framework emphasizes the importance of evaluating and refining the model throughout the project. This continuous improvement process is crucial for ensuring that the recommendation system is not only effective but also adaptable to changing business requirements.

In conclusion, the CRISP-DM methodology was chosen for its structured, flexible, and comprehensive approach, making it the ideal framework for guiding the development of an advanced recommendation system tailored to the needs of an e-commerce platform

Conclusion

In summary, applying the CRISP-DM steps to this project provides a systematic, adaptable and well-structured approach for carrying out our data analysis project. This allows us to optimize efficiency, manage resources wisely, and maximize the quality of results at each stage of the process.

4 Data Understanding

4.1 Data Acquisition

In this section, we begin with the data acquisition phase and then move on to the data description.

4.1.1 Source of the Data

The data used in this project is sourced from a CMS platform named Shopify. Shopify is a widely used e-commerce platform that allows businesses to manage their online stores, including product listings, customer data, and order management. For this project, the data is categorized into three distinct datasets: products, customers, and orders. These datasets contain essential information that forms the foundation for building the recommendation system, such as product attributes, customer demographics, and order history.

4.1.2 Data Collection Process

The data collection process involves exporting relevant information directly from Shopify's admin interface. The steps to acquire the data are as follows:

1. **Navigating to Product Listings**: From the Shopify admin panel, go to the section labeled *Products* and then select *All products*. This will provide access to the complete list of products available in the store.
2. **Filtering and Selecting Products**: If only specific products are required, filters can be applied to narrow down the selection. Users have the option to export the current page of products, all products, selected products, or products matching a specific search and filter criteria.
3. **Exporting Data**: Once the products are selected, the *Export* option is chosen. In the export dialog box, the user selects the type of CSV file they wish to export. For this project, a *Plain CSV* file format is chosen, which is compatible with most data analysis tools.
4. **Final Output**: After completing the export process, a CSV file containing the selected product data is generated. This file can then be downloaded and inspected to ensure that the data meets the project's requirements.

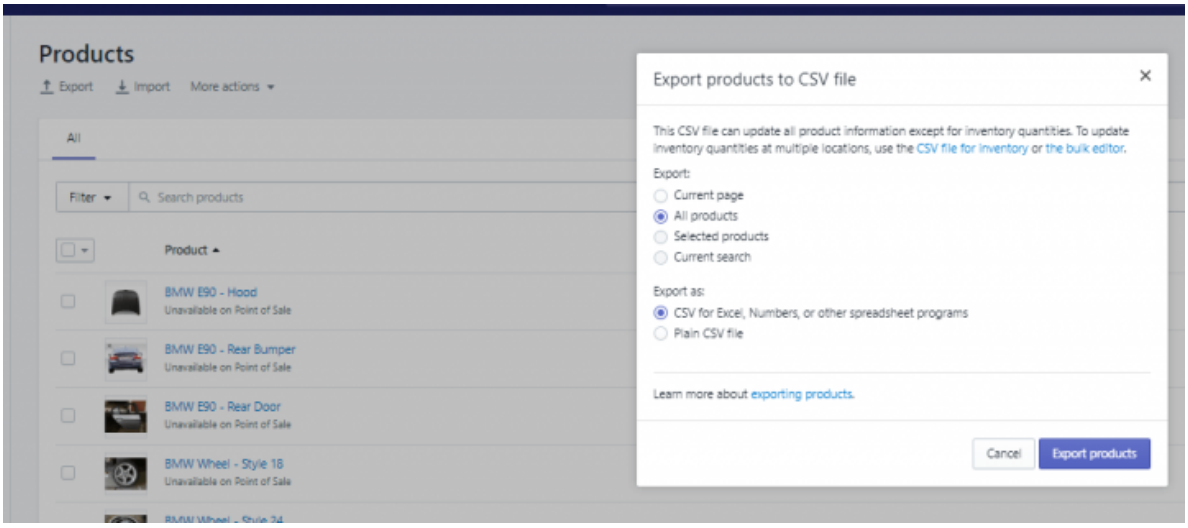


Figure 4.1: data exportaion

This process is repeated for the other datasets (customers and orders) to ensure a comprehensive collection of data for analysis and modeling.

4.2 Description of the Datasets

Before starting the data analysis phase, it is crucial to understand the data at our disposal. This means knowing what information is available, what it signifies, and what its sources are. This preliminary understanding is essential for properly addressing the problem we are trying to solve.

4.2.1 Overview of df_products.csv

Feature Name	Explanation
Handle	Unique identifier for the product.
Title	The name of the product.
Body (HTML)	HTML description of the product.
Vendor	The vendor or manufacturer of the product.
Product Category	Category under which the product is listed.
Type	Type of the product.
Tags	Tags associated with the product.
Published	Indicates whether the product is published.
Option1 Name	Name of the first product option.
Option1 Value	Value of the first product option.
Option2 Name	Name of the second product option.
Option2 Value	Value of the second product option.
Option3 Name	Name of the third product option.
Option3 Value	Value of the third product option.
Variant SKU	SKU for the product variant.
Variant Grams	Weight of the product variant in grams.
Variant Inventory Tracker	Inventory tracking method for the product variant.
Variant Inventory Qty	Quantity of the product variant in inventory.
Variant Inventory Policy	Policy for inventory management.
Variant Fulfillment Service	Fulfillment service for the product variant.
Variant Price	Price of the product variant.
Variant Compare At Price	Compare-at price for the product variant.
Variant Requires Shipping	Indicates if the product variant requires shipping.
Variant Taxable	Indicates if the product variant is taxable.
Variant Barcode	Barcode for the product variant.
Image Src	URL of the product image.
Image Position	Position of the image in the gallery.
Image Alt Text	Alt text for the product image.
Gift Card	Indicates if the product is a gift card.
SEO Title	SEO title for the product.
SEO Description	SEO description for the product.
Google Shopping / Google Product Category	Google Shopping product category.
Google Shopping / Gender	Target gender for Google Shopping.
Google Shopping / Age Group	Age group for Google Shopping.

Google Shopping / MPN	Manufacturer Part Number for Google Shopping.
Google Shopping / Condition	Condition of the product for Google Shopping.
Google Shopping / Custom Product	Custom product flag for Google Shopping.
Google Shopping / Custom Label 0	Custom label 0 for Google Shopping.
Google Shopping / Custom Label 1	Custom label 1 for Google Shopping.
Google Shopping / Custom Label 2	Custom label 2 for Google Shopping.
Google Shopping / Custom Label 3	Custom label 3 for Google Shopping.
Google Shopping / Custom Label 4	Custom label 4 for Google Shopping.
Variant Image	Image for the product variant.
Variant Weight Unit	Weight unit for the product variant.
Variant Tax Code	Tax code for the product variant.
Cost per item	Cost of the product per item.
Included / France	Price included for France.
Price / France	Price for France.
Compare At Price / France	The original price of the product variant in the French market.
Included / International	Indicates whether the product variant is included in international markets.
Price / International	The price of the product variant in international markets.
Compare At Price / International	The original price of the product variant in international markets.
Included / Italie	Indicates whether the product variant is included in the Italian market.
Price / Italie	The price of the product variant in the Italian market.
Compare At Price / Italie	The original price of the product variant in the Italian market.
Status	The current status of the product (e.g., "Active", "Archived").

Table 4.1: Features and Explanations for df_products Dataset

4.2.2 Overview of df_customers.csv

Feature Name	**Explanation**
Customer ID	A unique identifier assigned to each customer.
First Name	The first name of the customer.
Last Name	The last name of the customer.
Email	The email address of the customer.
Accepts Email Marketing	Indicates whether the customer has opted in to receive email marketing communications.
Default Address Company	The company name associated with the customer's default address.
Default Address Address1	The primary address line for the customer's default address.
Default Address Address2	The secondary address line for the customer's default address (if applicable).
Default Address City	The city of the customer's default address.
Default Address Province Code	The province or state code of the customer's default address.
Default Address Country Code	The country code of the customer's default address.
Default Address Zip	The postal code or zip code of the customer's default address.
Default Address Phone	The phone number associated with the customer's default address.
Phone	The primary phone number of the customer.
Accepts SMS Marketing	Indicates whether the customer has opted in to receive SMS marketing communications.
Total Spent	The total amount of money spent by the customer on the e-commerce platform.

Table 4.2: Features and Explanations for df_customer Dataset

Total Orders	The total number of orders placed by the customer.
Note	Any additional notes or remarks associated with the customer.
Tax Exempt	Indicates whether the customer is exempt from taxes.
Tags	Any tags or labels assigned to the customer for categorization or identification purposes.

Table 4.3: Features and Explanations for df_customer Dataset

4.2.3 Overview of df_orders.csv

Feature Name	Explanation
Name	Customer's name.
Email	Customer's email address.
Financial Status	Payment status of the order.
Paid at	Date and time when payment was made.
Fulfillment Status	Status of order fulfillment.
Fulfilled at	Date and time when the order was fulfilled.
Accepts Marketing	Indicates if the customer accepts marketing.
Currency	Currency used for the order.
Subtotal	Total price of items before shipping and taxes.
Shipping	Shipping cost.
Taxes	Tax amount applied to the order.
Total	Total amount of the order including taxes and shipping.
Discount Code	Code used for applying discounts.
Discount Amount	Amount of discount applied.
Shipping Method	Method of shipping chosen for the order.
Created at	Date and time when the order was created.
Lineitem quantity	Quantity of items in the order.
Lineitem name	Name of the product item.
Lineitem price	Price of the product item.
Lineitem compare at price	Compare-at price of the product item.
Lineitem sku	SKU of the product item.
Lineitem requires shipping	Indicates if the item requires shipping.
Lineitem taxable	Indicates if the item is taxable.
Lineitem fulfillment status	Fulfillment status of the product item.
Billing Name	Name of the person being billed.
Billing Street	Street address for billing.
Billing Address1	Address line 1 for billing.
Billing Address2	Address line 2 for billing.
Billing Company	Company name for billing.
Billing City	City for billing address.
Billing Zip	Zip code for billing address.
Billing Province	Province for billing address.
Billing Country	Country for billing address.
Billing Phone	Phone number for billing contact.
Shipping Name	Name of the person receiving the shipment.
Shipping Street	Street address for shipping.
Shipping Address1	Address line 1 for shipping.
Shipping Address2	Address line 2 for shipping.
Shipping Company	Company name for shipping.
Shipping City	City for shipping address.
Shipping Zip	Zip code for shipping address.
Shipping Province	Province for shipping address.
Shipping Country	Country for shipping address.
Shipping Phone	Phone number for shipping contact.
Notes	Additional notes for the order.
Note Attributes	Attributes related to the order notes.
Cancelled at	Date and time when the order was canceled.

Table 4.4: Features and Explanations for df_orders Dataset

Feature Name	Explanation
Payment Method	Method used for payment.
Payment Reference	Reference for the payment.
Refunded Amount	Amount refunded to the customer.
Vendor	Vendor associated with the order.
Id	Unique identifier for the order.
Tags	Tags associated with the order.
Risk Level	Risk level associated with the order.
Source	Source from where the order was placed.
Lineitem discount	Discount applied to the product item.
Tax 1 Name	Name of the first tax applied.
Tax 1 Value	Value of the first tax applied.
Tax 2 Name	Name of the second tax applied.
Tax 2 Value	Value of the second tax applied.
Tax 3 Name	Name of the third tax applied.
Tax 3 Value	Value of the third tax applied.
Tax 4 Name	Name of the fourth tax applied.
Tax 4 Value	Value of the fourth tax applied.
Tax 5 Name	Name of the fifth tax applied.
Tax 5 Value	Value of the fifth tax applied.
Phone	Phone number associated with the order.
Receipt Number	Receipt number for the order.
Duties	Duties applied to the order.
Billing Province Name	Name of the billing province.
Shipping Province Name	Name of the shipping province.
Payment ID	Unique payment identifier.
Payment Terms Name	Payment terms associated with the order.
Next Payment Due At	Date and time for the next payment due.
Payment References	References related to the payment.

Table 4.5: Features and Explanations for df_orders Dataset

Conclusion: From the dataset description, it is evident that the customer_id feature is not present in the df_orders dataset. However, alternative features such as Email, Billing Phone, and Shipping Phone can serve as linking attributes to associate orders with the appropriate customers. The initial step in our project involves accurately linking the orders to their corresponding customers using these alternative features. Additionally, it is crucial to remove confidential information, including names, emails, and addresses, to ensure data privacy and confidentiality. This will guarantee that our recommendation system is built on properly linked and anonymized customer-order relationships.

4.2.4 Establishing Customer-Order Relationships and Handling Sensitive Data

In this section, we address two crucial tasks: linking orders to the appropriate customers and ensuring the confidentiality of sensitive data. This involves enriching the dataset with missing customer information and removing unnecessary or confidential columns.

Linking Orders to Customers

Initially, the dataset lacked a direct `customer_id` in the `df_orders` dataset. To establish a connection between orders and customers, we used alternative features such as `Email`, `Billing Phone`, and `Shipping Phone`.

Linking by Email: We first attempted to link orders to customers using email addresses. Each email in the `df_orders` dataset was matched with the corresponding customer ID in the `df_customers` dataset.

```
customer_id_list = []
for email in df_orders['Email']:
    try:
        customer_id = df_customers[df_customers['Email'] == email]['Customer
        ↪  ID'].values[0]
    except IndexError:
        customer_id = 'no'  # Placeholder for missing customer IDs
    customer_id_list.append(customer_id)

df_orders['customer_id'] = customer_id_list
```

Listing 1: Linking orders to customers using emails

Linking by phone for Unmatched Orders: Orders that could not be linked by email were processed to find matches based on phone numbers. We created a separate DataFrame for these unmatched orders and tried to link them using phone numbers from both `df_orders` and `df_customers`.

```
df_orders_no_email = df_orders[df_orders['customer_id'] == 'no'].copy()

def match_phone_and_add_customer_id(df_orders_no_email, df_customers):
    df_orders_with_phone = pd.DataFrame()
    for idx, order_row in df_orders_no_email.iterrows():
        phones = [order_row['Phone'], order_row['Billing Phone'],
        ↪  order_row['Shipping Phone']]
        matching_customer = df_customers[df_customers['Phone'].isin(phones)]
        if not matching_customer.empty:
            order_row['customer_id'] = matching_customer['Customer ID'].values[0]
            df_orders_with_phone = pd.concat([df_orders_with_phone,
            ↪  pd.DataFrame([order_row])], ignore_index=True)
    return df_orders_with_phone

df_orders_with_phone = match_phone_and_add_customer_id(df_orders_no_email,
↪  df_customers)
```

Listing 2: Linking orders by phone numbers for unmatched orders

Combining Datasets The newly linked orders were combined with the initially matched orders to form a comprehensive dataset with customer IDs.

```
df_combined_orders = pd.concat([df_orders, df_orders_with_phone])
```

Listing 3: Combining matched orders into a single dataset

4.2.5 Removing Confidential and Unnecessary Data

To ensure privacy and reduce data complexity, we removed several columns containing confidential or redundant information. This step helps in maintaining data security and focusing on relevant features for analysis.

Null Value Analysis and Data Cleaning

After linking the orders and customers and removing unnecessary columns, we further analyzed the datasets to identify and remove columns with null values.

Analyzing `df_orders` for Null Values: We first plotted a heatmap to visualize the presence of null values in the `df_orders` dataset.

```
plt.figure(figsize=(12, 8))
sns.heatmap(df_combined_orders.isnull(), cbar=False, cmap='viridis')
plt.show()
```

Listing 4: Heatmap of null values in `df_orders` dataset before cleaning

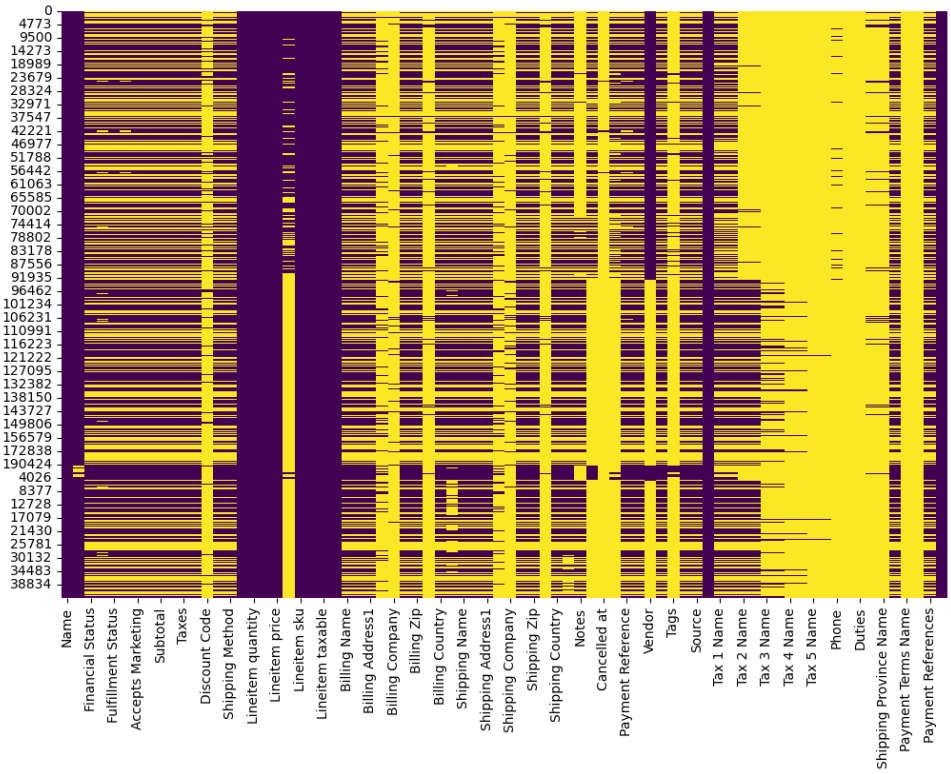


Figure 4.2: Heatmap of null values in `df_orders` dataset before cleaning

Based on the visualization, we identified and dropped columns that contained null values, along with columns that contain confidential data about the customers.

```
df_combined_orders.drop([
    'Email', 'Lineitem compare at price', 'Financial Status', 'Paid at',
    ↪  'Fulfillment Status', 'Fulfilled at',
    'Accepts Marketing', 'Currency', 'Subtotal', 'Shipping', 'Taxes', 'Total',
    ↪  'Discount Code', 'Discount Amount',
    'Shipping Method', 'Billing Name', 'Billing Street', 'Billing Address1',
    ↪  'Billing Address2', 'Billing Company',
    'Billing City', 'Billing Zip', 'Billing Province', 'Billing Country', 'Billing
    ↪  Phone', 'Shipping Name',
    'Shipping Street', 'Shipping Address1', 'Shipping Address2', 'Shipping
    ↪  Company', 'Shipping City', 'Shipping Zip',
    'Shipping Province', 'Shipping Country', 'Shipping Phone', 'Notes', 'Note
    ↪  Attributes', 'Cancelled at',
    'Payment Method', 'Payment Reference', 'Refunded Amount', 'Vendor', 'Id',
    ↪  'Tags', 'Risk Level', 'Source',
    'Tax 1 Name', 'Tax 1 Value', 'Tax 2 Name', 'Tax 2 Value', 'Tax 3 Name', 'Tax 3
    ↪  Value', 'Tax 4 Name',
    'Tax 4 Value', 'Tax 5 Name', 'Tax 5 Value', 'Phone', 'Receipt Number',
    ↪  'Duties', 'Billing Province Name',
    'Shipping Province Name', 'Payment ID', 'Payment Terms Name', 'Next Payment
    ↪  Due At', 'Payment References'
], axis=1, inplace=True)
```

Listing 5: Dropping columns with null values in `df_orders`

We then plotted the heatmap again to confirm the removal of null values:

```
plt.figure(figsize=(12, 8))
sns.heatmap(df_combined_orders.isnull(), cbar=False, cmap='viridis')
plt.show()
```

Listing 6: Heatmap of `df_orders` dataset after cleaning

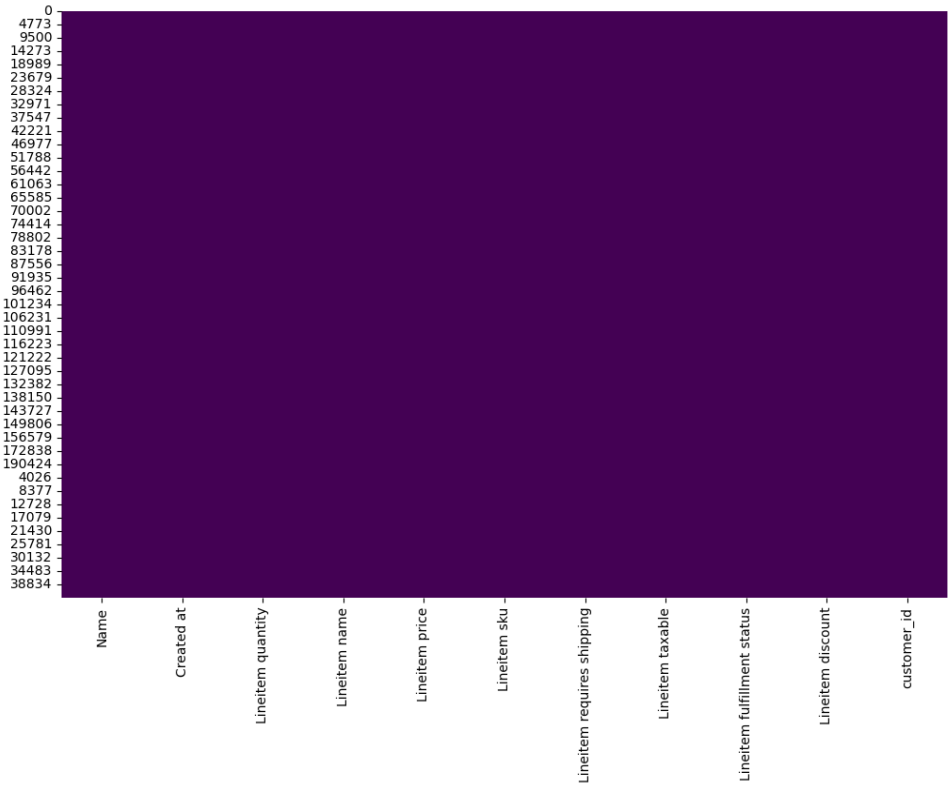


Figure 4.3: Heatmap of null values in `df_orders` dataset after cleaning

Analyzing `df_customers` for Null Values Next, we analyzed the `df_customers` dataset to identify columns with null values.

```
plt.figure(figsize=(12, 8))
sns.heatmap(df_customers.isnull(), cbar=False, cmap='viridis')
plt.show()
```

Listing 7: Heatmap of null values in `df_customers` dataset before cleaning

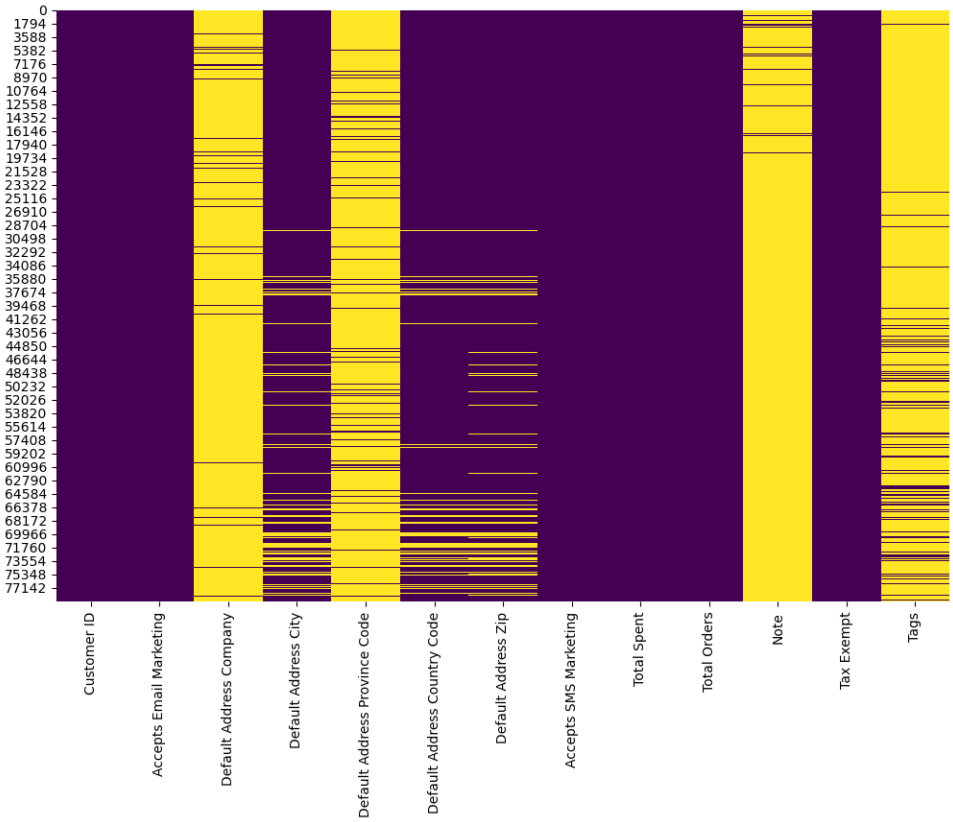


Figure 4.4: Heatmap of null values in `df_customers` dataset before cleaning

We identified and dropped columns with significant null values in the `df_customers` dataset, along with columns that contain confidential data about the customers.

```
df_customers.drop(['Phone', 'Email', 'First Name', 'Last Name', 'Default Address
↪  Address1', 'Default Address Phone', 'Billing Phone', 'Shipping Phone'],
↪  axis=1, inplace=True)
```

Listing 8: Dropping columns with null values in `df_customers`

Analyzing `df_products` for Null Values Next, we analyzed the `df_products` dataset to identify columns with null values.

```
plt.figure(figsize=(12, 8))
sns.heatmap(df_products.isnull(), cbar=False, cmap='viridis')
plt.show()
```

Listing 9: Heatmap of null values in `df_products` dataset before cleaning

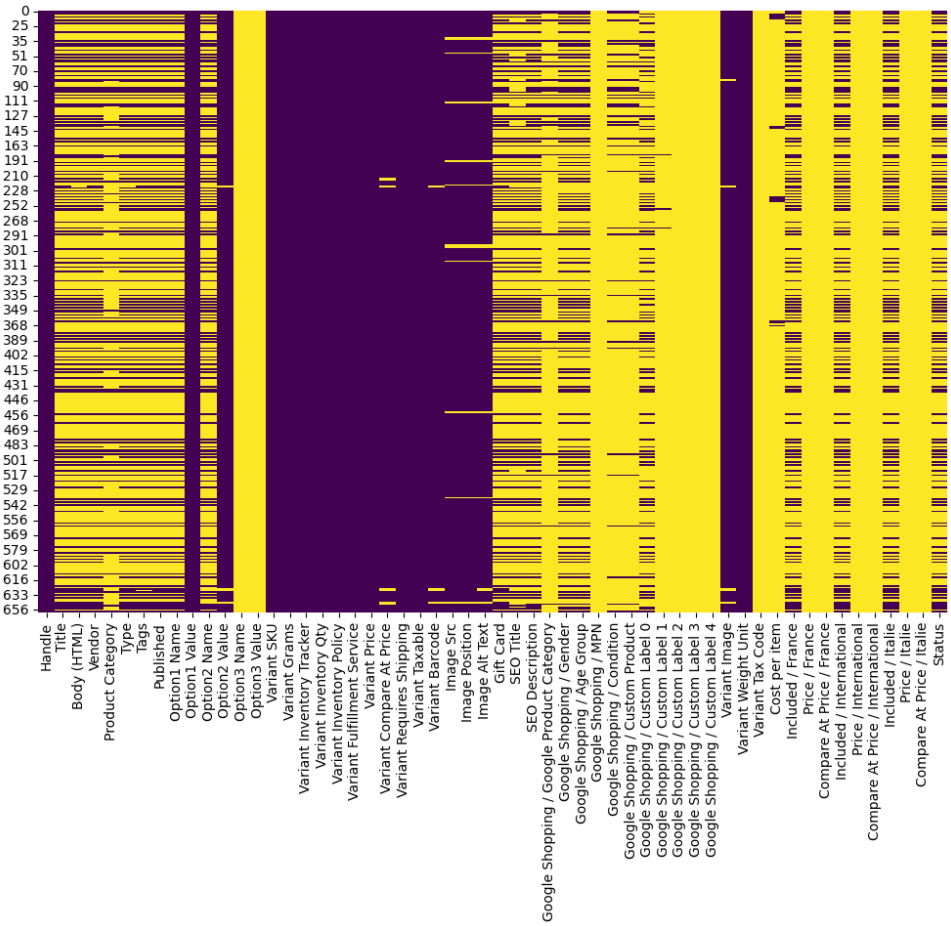


Figure 4.5: Heatmap of null values in `df_products` dataset before cleaning

We identified and dropped columns with significant null values in the `df_products` dataset.

```
df_products.drop(['Vendor', 'Option1 Name', 'Option2 Name', 'Option3 Name',
↪  'Option3 Value', 'Variant Inventory Tracker', 'Variant Inventory Policy',
↪  'Variant Fulfillment Service', 'Variant Requires Shipping', 'Variant Taxable',
↪  'Variant Barcode', 'Image Src', 'Image Position', 'Image Alt Text', 'Gift
↪  Card', 'Google Shopping / MPN', 'Google Shopping / Custom Label 2', 'Google
↪  Shopping / Custom Label 3', 'Google Shopping / Custom Label 4', 'Variant
↪  Image', 'Variant Weight Unit', 'Variant Tax Code', 'Price / France', 'Compare
↪  At Price / France', 'Included / International', 'Price / International',
↪  'Compare At Price / International','Google Shopping / Condition', 'Google
↪  Shopping / Custom Product', 'Google Shopping / Custom Label 1', 'Included /
↪  France', 'Included / Italie', 'Price / Italie', 'Compare At Price / Italie'],
↪  axis=1, inplace=True)
```

Listing 10: Dropping columns with null values in `df_customers`

4.3 Filtered Data Visualization

4.3.1 Visualization of df_customers

In this subsection, we explore the filtered `df_customers` dataset by visualizing both categorical and numerical features. These visualizations provide insight into the distribution

of data, patterns, and potential outliers.

Categorical Features Visualization

We start by visualizing the categorical columns in the `df_customers` dataset.

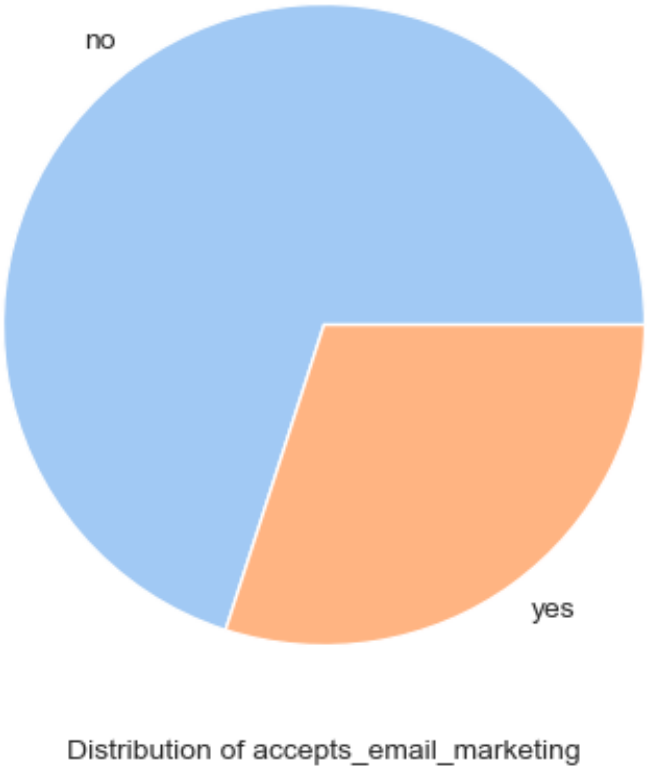


Figure 4.6: Distribution of `accepts_email_marketing`

The pie chart above (Figure 4.6) shows the distribution of customers who accept email marketing. This visualization helps in understanding the proportion of customers who have opted in or out of email marketing.

Figure 4.7: Word Cloud of Unique Values in `default_address_city`

In Figure 4.7, a word cloud is generated to represent the unique values of the `default_address_city`

column. This visual representation helps in identifying the most common cities where customers are located.

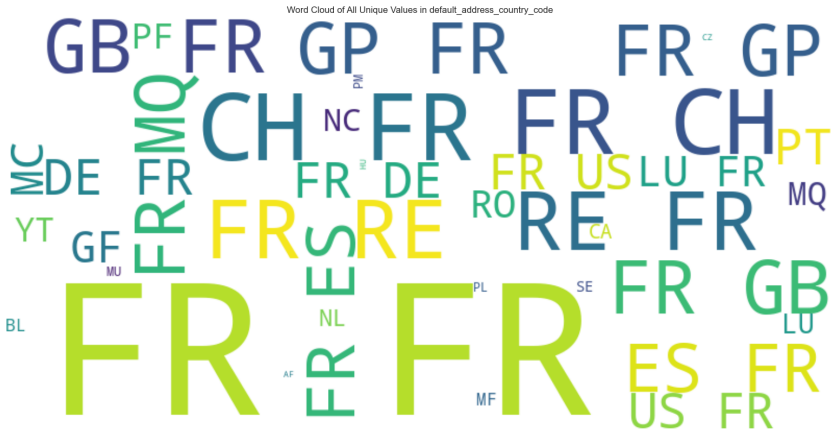


Figure 4.8: Word Cloud of Unique Values in `default_address_country_code`

Similarly, Figure 4.8 displays a word cloud of the unique values in the `default_address_country_code` column, highlighting the most frequent countries of the customers.

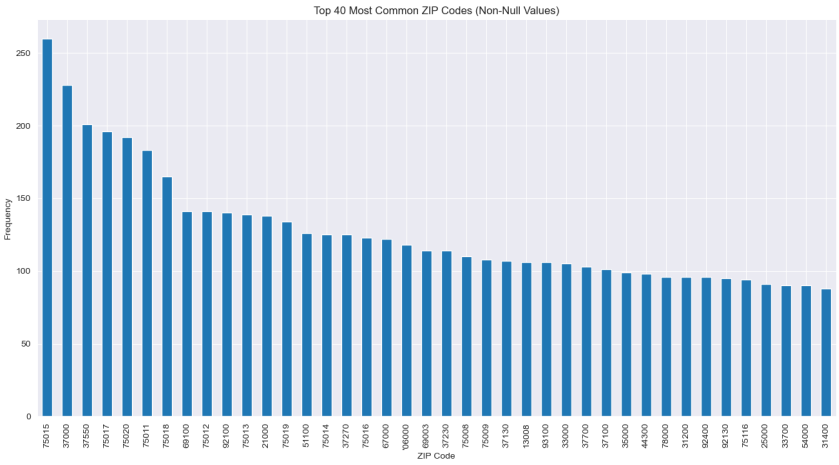


Figure 4.9: Top 40 Most Common ZIP Codes (Non-Null Values)

Figure 4.9 presents a bar chart of the top 40 most common ZIP codes found in the `default_address_zip` column. This visualization is crucial for understanding the geographic distribution of the customer base.

Numerical Features Visualization

Next, we visualize the numerical columns in the `df_customers` dataset.

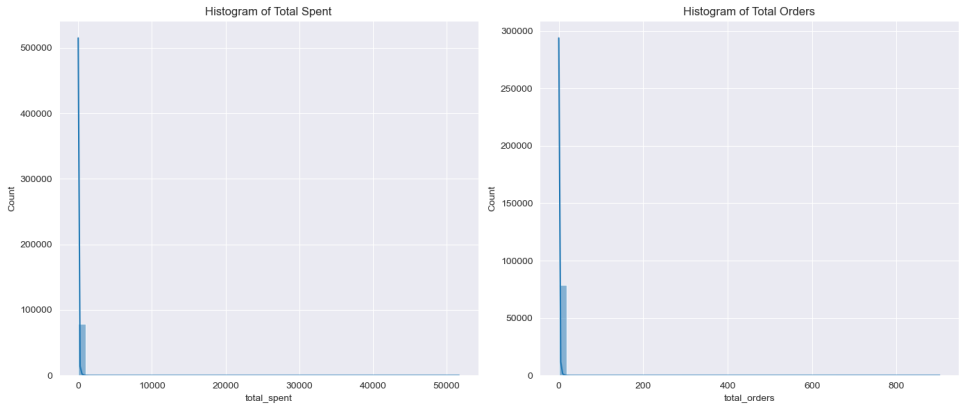


Figure 4.10: Histograms of `total_spent` and `total_orders`

In Figure 4.10, we explore the distribution of the `total_spent` and `total_orders` columns using histograms and boxplots. The distributions give insights into customer spending behavior and the number of orders placed by customers.

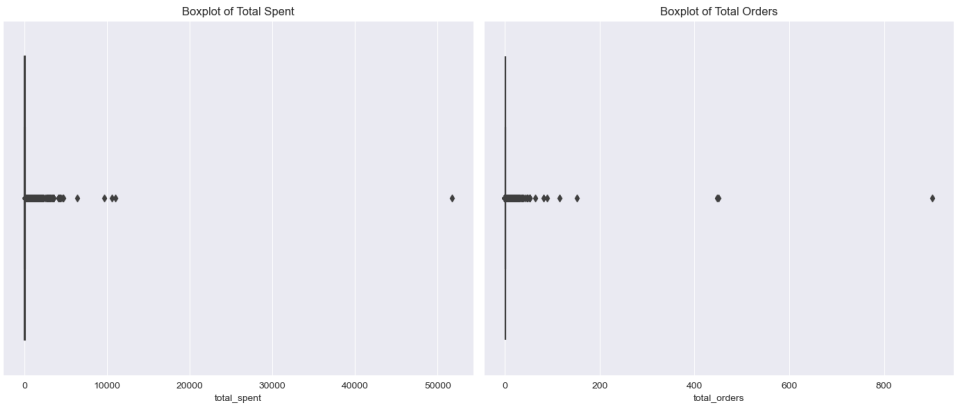


Figure 4.11: Boxplots of `total_spent` and `total_orders`

Figure 4.11 displays boxplots for the `total_spent` and `total_orders` columns. These boxplots are useful for identifying outliers and understanding the central tendency of the data.

4.3.2 Visualization of df_orders

Numerical Features Visualization

In this subsection, we explore the filtered `df_orders` dataset by visualizing the relationships between its numerical features. The visualizations help to identify patterns and correlations within the data.

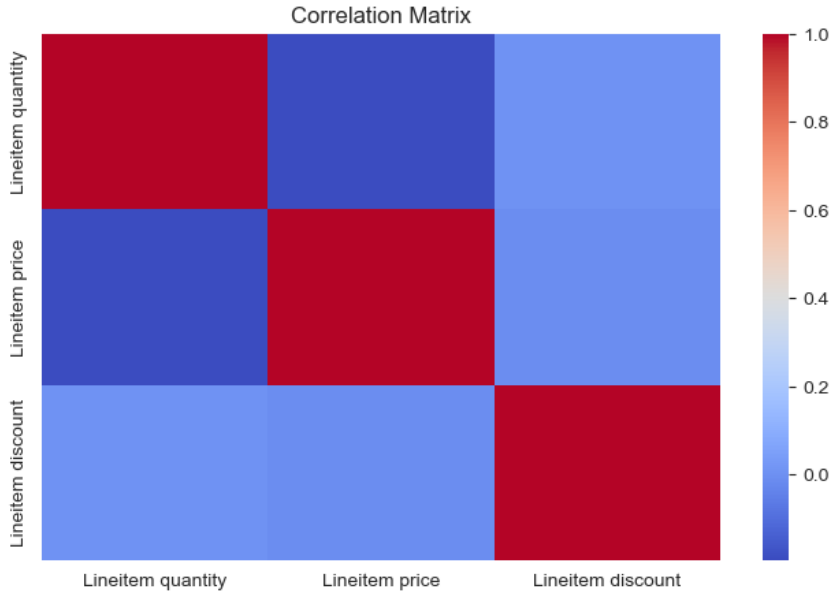


Figure 4.12: Correlation Matrix of Numeric Columns in `df_orders`

Figure 4.12 shows the correlation matrix of the numeric columns in the `df_orders` dataset. The heatmap uses a 'coolwarm' color palette to represent the strength of the correlations. This visualization is crucial for identifying relationships between different numerical features, which could influence further modeling steps.

4.3.3 Visualization of df_products

In this subsection, we explore the filtered `df_products` dataset by visualizing both categorical and numerical features. These visualizations provide insight into the distribution and relationships of the product-related data.

Categorical Features Visualization

We start by visualizing the distribution of various categorical features in the `df_products` dataset. The features include `product_category`, `type`, `tags`, `published`, and several Google Shopping attributes. For each categorical feature, a histogram is plotted to show the frequency of each category.

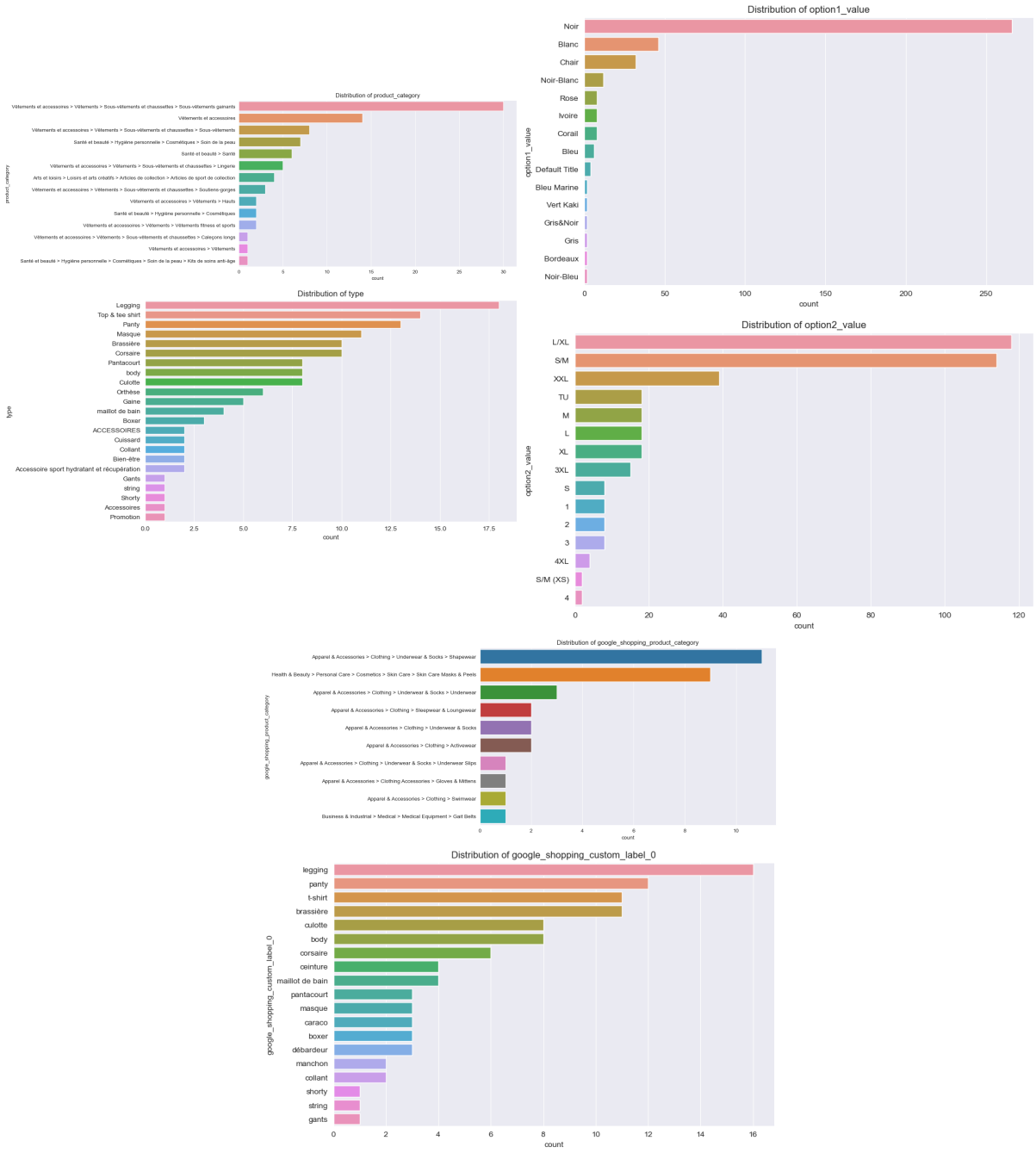


Figure 4.13: Distribution of Categorical Features in `df_products`

Figure 4.13 presents the distribution of selected categorical features. This helps in understanding the variability and common categories in the product data.

Numerical Columns Visualization

Next, we explore the numerical columns of the `df_products` dataset. We begin by plotting the correlation matrix to identify relationships between these numerical features.

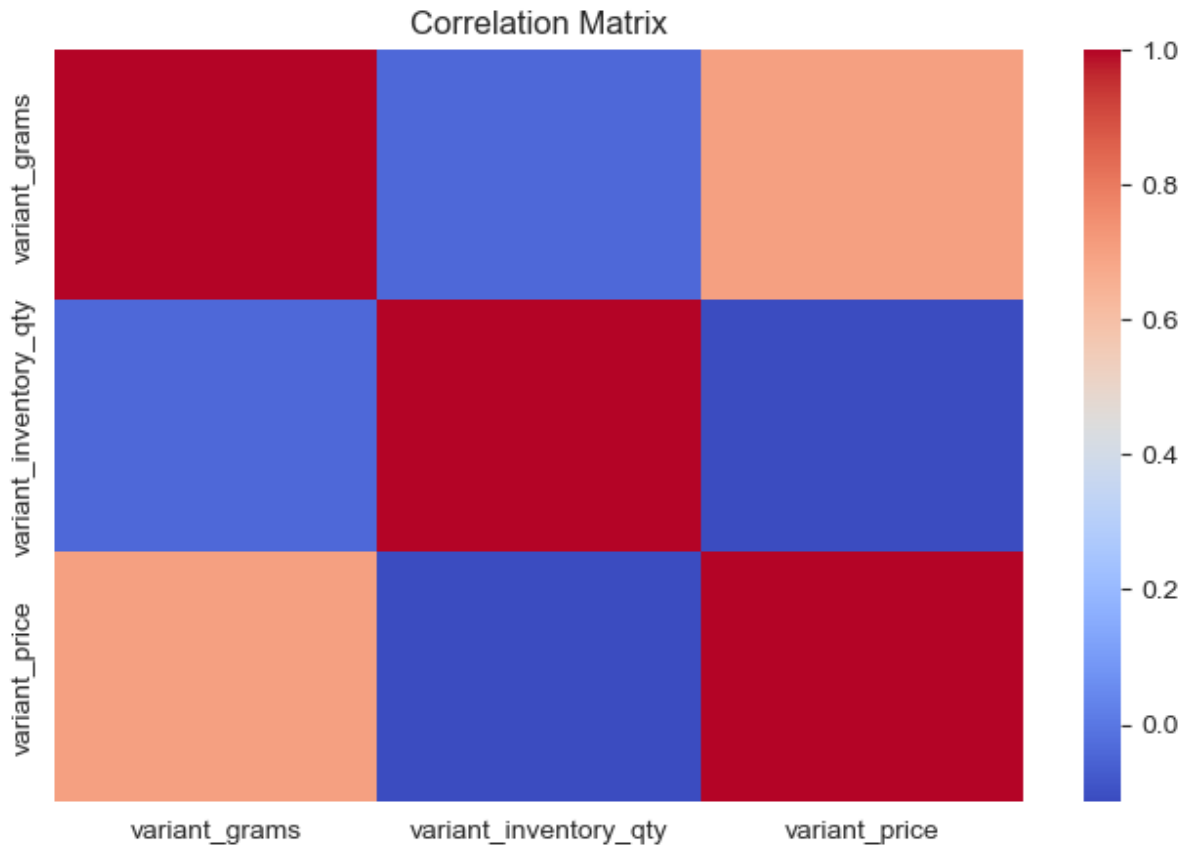


Figure 4.14: Correlation Matrix of Numeric Columns in `df_products`

From the correlation matrix, we identify highly correlated column pairs, specifically those with a correlation greater than 0.8. These pairs are listed in Table

Distribution and Boxplots of Key Numerical Features: To further explore the numerical features, histograms and boxplots are generated for key variables such as `variant_price`, `variant_inventory_qty`, and `variant_grams`. These visualizations help to understand the distribution, central tendency, and spread of these features.

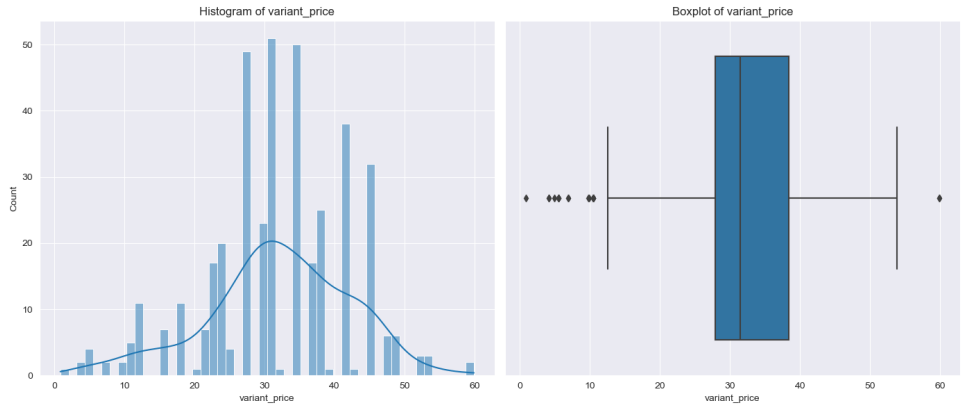


Figure 4.15: Histogram and Boxplot of `variant_price`

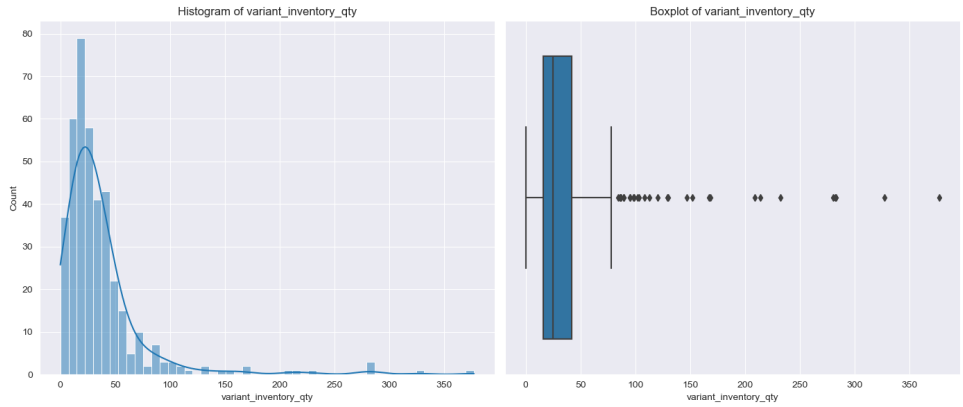


Figure 4.16: Histogram and Boxplot of `variant_inventory_qty`

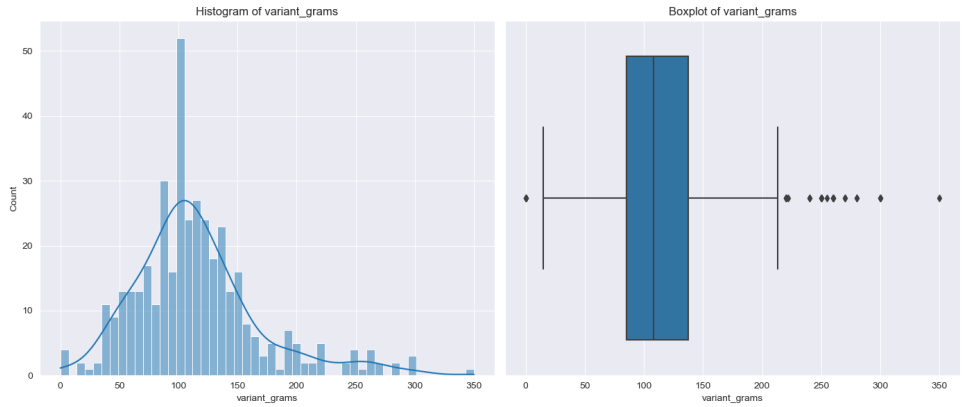


Figure 4.17: Histogram and Boxplot of `variant_grams`

Figures 4.15, 4.16, and 4.17 show the histograms and boxplots for the `variant_price`, `variant_inventory_qty`, and `variant_grams` columns, respectively.

Overall Distribution of Numerical Columns: Lastly, to gain a comprehensive view of the numerical data, histograms, box plots, and density plots for all numerical columns are created.

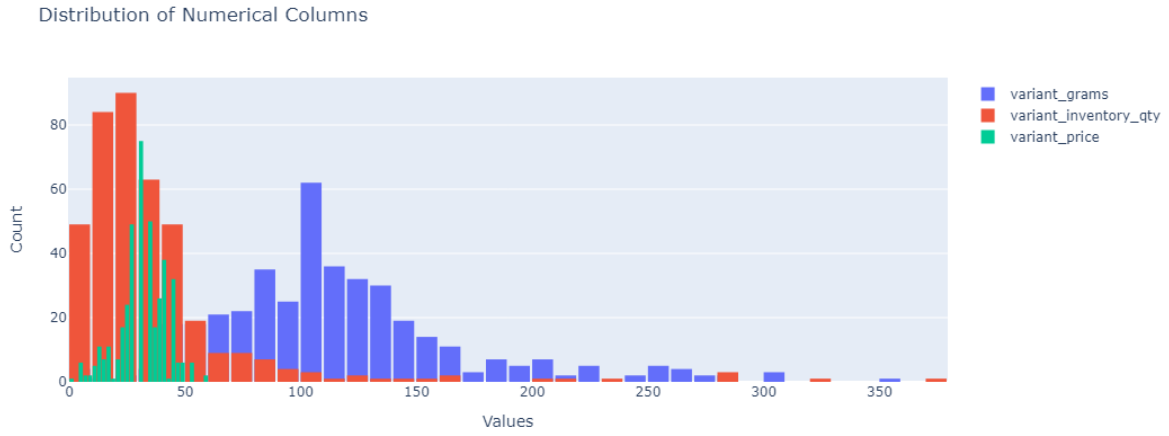


Figure 4.18: Distribution of All Numerical Columns in `df_products`

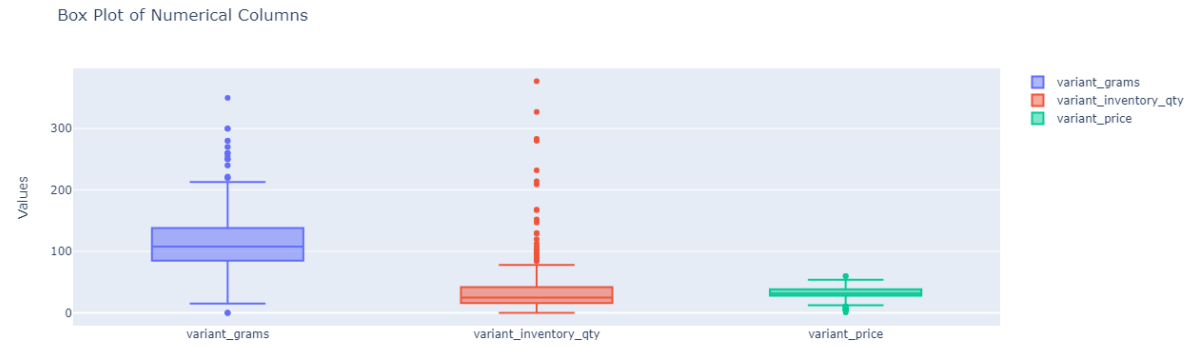


Figure 4.19: Box Plot of All Numerical Columns in `df_products`

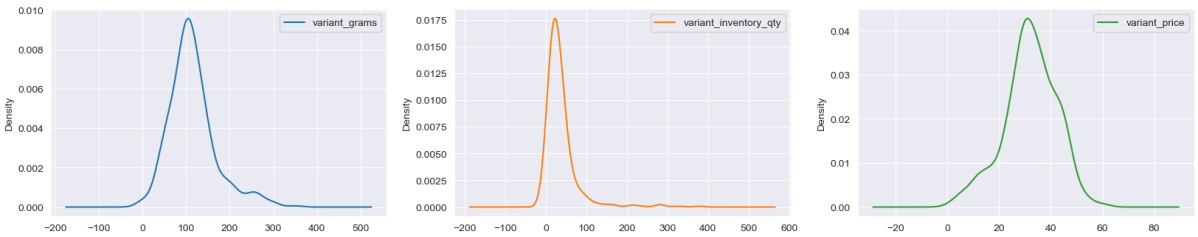


Figure 4.20: Density Plots of All Numerical Columns in `df_products`

Figures 4.18, 4.19, and 4.20 display the overall distribution, box plots, and density plots of all numerical columns in the dataset, respectively. These plots provide a holistic view of the data's distribution and variability, which is essential for understanding the characteristics of the `df_products` dataset.

5 Data Preprocessing

5.1 Context and Objectives of Preprocessing

The data preprocessing stage plays a crucial role in our project. Its objective is to prepare and format the raw data to make it usable in our machine learning models. This phase is essential for optimizing data quality, thereby ensuring accurate and meaningful results during the subsequent analysis and modeling stages.

5.2 Feature Engineering

One of the first steps in preprocessing is feature engineering. This involves identifying and processing the essential features for our analysis. To achieve this, we used various techniques such as normalization, scaling, and encoding of categorical variables.

5.2.1 Feature Engineering of cutomers' dataset

Encoding Binary Categorical Variables:

Binary categorical variables were converted into numerical formats suitable for machine learning algorithms. For instance, the 'accepts_email_marketing' feature, which originally had 'yes' and 'no' values, was mapped to 1 and 0, respectively. Similarly, the 'default_address_company_status' feature was converted into a binary format, where a non-null value was encoded as 1 and a null value as 0.

```
# Convert 'accepts_email_marketing' to binary
df_customers['accepts_email_marketing'] =
↪  df_customers['accepts_email_marketing'].map({'yes': 1, 'no': 0})
# Convert 'default_address_company_status' to binary
df_customers['default_address_company_status'] =
↪  np.where(df_customers['default_address_company_status'].notnull(), 1, 0)
```

Listing 11: Encoding Binary Categorical Variables in df_customers

Encoding Multi-Categorical Features:

For features with multiple categories, different strategies were employed:

- **Top N Encoding:** The 'default_address_city' feature was encoded by retaining the top 50 cities based on frequency and grouping the rest under the category 'other'.
- **One-Hot Encoding:** The 'default_address_city', 'default_address_country_code', and 'zip_region' features were one-hot encoded to convert categorical variables into

a numerical format. For ‘zip_region‘, the first digit of the ‘default_address_zip‘ was extracted and used for encoding.

```
# Handle 'default_address_city': Encode top N cities, group others as 'other'
top_cities = df_customers['default_address_city'].value_counts().index[:50]
df_customers['default_address_city'] =
↪  df_customers['default_address_city'].apply(lambda x: x if x in top_cities else
↪  'other')

# One-hot encode 'default_address_city'
df_customers = pd.get_dummies(df_customers, columns=['default_address_city'],
↪  prefix='city')
# One-hot encode 'default_address_country_code'
df_customers = pd.get_dummies(df_customers,
↪  columns=['default_address_country_code'], prefix='country')
# Handle 'default_address_zip': Group by the first digit
df_customers['zip_region'] = df_customers['default_address_zip'].apply(lambda x:
↪  str(x)[0] if pd.notna(x) else x)

# One-hot encode 'zip_region'
df_customers = pd.get_dummies(df_customers, columns=['zip_region'], prefix='zip')

# Drop the original 'default_address_zip' column as it's now represented by
↪  'zip_region'
df_customers.drop(columns=['default_address_zip'], inplace=True)
```

Listing 12: Encoding Multi-Categorical Features

Through these transformations, categorical variables in the ‘df_customers‘ dataset were converted into a format suitable for machine learning models, which helped improve the performance and accuracy of our predictive models.

Numerical Feature Engineering

The numerical features within the ‘df_customers‘ dataset were analyzed and transformed to ensure they were in a suitable format for modeling. This process involved understanding the distribution of each feature and applying various transformations to reduce skewness and kurtosis, which are common issues in raw data.

Skewness and Kurtosis Analysis: To determine the nature of the distribution for each numerical feature, skewness and kurtosis were calculated for the ‘total_spent‘, ‘total_orders‘, and ‘avg_spent_per_order‘ features. High skewness and kurtosis values indicated that the distributions were heavily skewed and leptokurtic, which could potentially affect model performance.

```
print("Skewness of total_spent:", skew(df_customers['total_spent']))
print("Kurtosis of total_spent:", kurtosis(df_customers['total_spent']))

print("Skewness of total_orders:", skew(df_customers['total_orders']))
print("Kurtosis of total_orders:", kurtosis(df_customers['total_orders']))

print("Skewness of avg_spent_per_order:",
↪  skew(df_customers['avg_spent_per_order']))
print("Kurtosis of avg_spent_per_order:",
↪  kurtosis(df_customers['avg_spent_per_order']))
```

Listing 13: Skewness and Kurtosis of Numerical Features

Transformations Applied:

Given the high skewness and kurtosis, several transformations were tested to normalize these distributions:

- **Log Transformation:** Applied to all features to reduce skewness.
- **Square Root Transformation:** Applied to all features to moderate skewness and stabilize variance.
- **Box-Cox Transformation:** Used for strictly positive data to achieve a more normal-like distribution.
- **Yeo-Johnson Transformation:** Applicable to both positive and negative data, this transformation was used to normalize data without the constraint of strictly positive values.
- **Min-Max Normalization:** Scaled the data to a range of $[0, 1]$.
- **Standardization:** Standardized the data to have a mean of 0 and a standard deviation of 1.

```
# Log Transformation
df_customers['log_total_spent'] = np.log1p(df_customers['total_spent'])
df_customers['log_total_orders'] = np.log1p(df_customers['total_orders'])
df_customers['log_avg_spent_per_order'] =
↪  np.log1p(df_customers['avg_spent_per_order'])

# Square Root Transformation
df_customers['sqrt_total_spent'] = np.sqrt(df_customers['total_spent'])
df_customers['sqrt_total_orders'] = np.sqrt(df_customers['total_orders'])
df_customers['sqrt_avg_spent_per_order'] =
↪  np.sqrt(df_customers['avg_spent_per_order'])

# Box-Cox Transformation (only for strictly positive values)
df_customers['boxcox_total_spent'], _ = stats.boxcox(df_customers['total_spent'] +
↪  1)
df_customers['boxcox_total_orders'], _ = stats.boxcox(df_customers['total_orders']
↪  + 1)
df_customers['boxcox_avg_spent_per_order'], _ =
↪  stats.boxcox(df_customers['avg_spent_per_order'] + 1)

# Yeo-Johnson Transformation (can handle zero and negative values)
df_customers['yeojohnson_total_spent'], _ =
↪  stats.yeojohnson(df_customers['total_spent'])
df_customers['yeojohnson_total_orders'], _ =
↪  stats.yeojohnson(df_customers['total_orders'])
df_customers['yeojohnson_avg_spent_per_order'], _ =
↪  stats.yeojohnson(df_customers['avg_spent_per_order'])

# Min-Max Normalization
scaler = MinMaxScaler()
df_customers['minmax_total_spent'] =
↪  scaler.fit_transform(df_customers[['total_spent']])
df_customers['minmax_total_orders'] =
↪  scaler.fit_transform(df_customers[['total_orders']])
df_customers['minmax_avg_spent_per_order'] =
↪  scaler.fit_transform(df_customers[['avg_spent_per_order']])

# Standardization
scaler = StandardScaler()
df_customers['standard_total_spent'] =
↪  scaler.fit_transform(df_customers[['total_spent']])
df_customers['standard_total_orders'] =
↪  scaler.fit_transform(df_customers[['total_orders']])
df_customers['standard_avg_spent_per_order'] =
↪  scaler.fit_transform(df_customers[['avg_spent_per_order']])
```

Listing 14: Transformations Applied to Numerical Features

Evaluation and Selection of Best Transformation:

To determine the most effective transformation for each numerical feature, the distributions after each transformation were visualized and compared. The selected best transformations for each feature were:

- **total_spent:** Log Transformation.
- **total_orders:** Box-Cox Transformation.

- **avg_spent_per_order:** Box-Cox Transformation.

```
def plot_distributions(df, columns, transformations):
    fig, axes = plt.subplots(len(columns), len(transformations) + 1, figsize=(20,
    ↪  15))
    for i, col in enumerate(columns):
        sns.histplot(df[col], ax=axes[i, 0], kde=True)
        axes[i, 0].set_title(f'Original {col}')
        for j, (trans_name, trans_col) in enumerate(transformations.items()):
            sns.histplot(df[trans_col[i]], ax=axes[i, j + 1], kde=True)
            axes[i, j + 1].set_title(f'{trans_name} {col}')
    plt.tight_layout()
    plt.show()

# Columns to transform
columns = ['total_spent', 'total_orders', 'avg_spent_per_order']
transformations = {
    'Log': ['log_total_spent', 'log_total_orders', 'log_avg_spent_per_order'],
    'Sqrt': ['sqrt_total_spent', 'sqrt_total_orders', 'sqrt_avg_spent_per_order'],
    'BoxCox': ['boxcox_total_spent', 'boxcox_total_orders',
    ↪  'boxcox_avg_spent_per_order'],
    'YeoJohnson': ['yeojohnson_total_spent', 'yeojohnson_total_orders',
    ↪  'yeojohnson_avg_spent_per_order'],
    'MinMax': ['minmax_total_spent', 'minmax_total_orders',
    ↪  'minmax_avg_spent_per_order'],
    'Standard': ['standard_total_spent', 'standard_total_orders',
    ↪  'standard_avg_spent_per_order']
}

# Plot the distributions
plot_distributions(df_customers, columns, transformations)
```

Listing 15: Comparison of Transformations on Numerical Features

The selected transformations were then applied, and the original features and less effective transformations were dropped from the dataset to streamline the modeling process.

```
# Drop original and non-selected transformed columns
df_customers.drop(['total_spent', 'total_orders', 'avg_spent_per_order',
                   'log_total_orders', 'log_avg_spent_per_order',
                   'sqrt_total_spent', 'sqrt_total_orders',
                   ↪  'sqrt_avg_spent_per_order',
                   'boxcox_total_spent', 'yeojohnson_total_spent',
                   'yeojohnson_total_orders', 'yeojohnson_avg_spent_per_order',
                   'minmax_total_spent', 'minmax_total_orders',
                   ↪  'minmax_avg_spent_per_order',
                   'standard_total_spent', 'standard_total_orders',
                   ↪  'standard_avg_spent_per_order'],
                  axis=1, inplace=True)
```

Listing 16: Dropping Non-Selected Columns in `df_customers`

This process of feature transformation ensured that the numerical data was in a more

normalized and less skewed format, improving the performance and interpretability of the machine learning models.

5.2.2 Feature Engineering of products' dataset

Categorical Feature Engineering in `df_products`

In the `df_products` dataset, several categorical columns were processed to prepare the data for modeling. The steps involved filling missing values, combining text columns, creating binary category columns, and applying one-hot encoding.

Handling Missing Values and Text Processing: First, categorical columns with textual data, such as `handle`, `title`, `body_html`, `product_category`, `type`, `tags`, `published`, `seo_title`, `google_shopping_product_category`, `google_shopping_gender`, `google_shopping_age_gr` and `google_shopping_custom_label_0`, were filled with empty strings to handle any missing values. These columns were then combined into a single text column, `combined_text`, for further processing. The combined text was converted to uppercase to ensure uniformity in text matching.

```
columns_to_check = [
    'handle', 'title', 'body_html', 'product_category', 'type', 'tags',
    'published', 'seo_title', 'google_shopping_product_category',
    ↪  'google_shopping_gender',
    'google_shopping_age_group', 'google_shopping_custom_label_0'
]
df_products[columns_to_check] = df_products[columns_to_check].fillna('')
df_products['combined_text'] = df_products[columns_to_check].apply(lambda x: '
↪  '.join(x.astype(str)), axis=1)
df_products['combined_text'] = df_products['combined_text'].str.upper()
```

Listing 17: Handling Missing Values and Combining Text Columns in `df_products`

Category and Subcategory Extraction: A set of categories and subcategories was defined based on the product types and attributes present in the data. For each category, binary columns were created to indicate the presence of specific keywords within the `combined_text` column.

```
categories = {
    'LEGGING': ['LEGGING'],
    'CULOTTE': ['CULOTTE'],
    'GAINE': ['GAINE'],
    'BODY': ['BODY'],
    'MAILLOT DE BAIN': ['MAILLOT DE BAIN'],
    'PANTACOURT': ['PANTACOURT'],
    'CORSAIRE': ['CORSAIRE'],
    'PANTY': ['PANTY'],
    'SHORTY': ['SHORTY'],
    'COLLANT': ['COLLANT'],
    'STRING': ['STRING'],
    'BRASSIÈRE': ['BRASSIERE'],
    'CARACO': ['CARACO'],
    'BOXER': ['BOXER'],
    'DEBARDEUR': ['DEBARDEUR'],
    'STRING': ['STRING'],
    'GANT': ['GANT'],
    'TOP & TEE SHIRT': ['TOP', 'TEE SHIRT', 'T-SHIRT'],
    'CEINTURE': ['CEINTURE'],
    'MASQUE': ['MASQUE'],
    'ACCESSOIRES': ['ACCESSOIRES'],
    'SOUS-VETEMENTS': ['SOUS', 'UNDERWEAR'],
    'SOCKS': ['SOCKS'],
    'VETEMENT': ['VETEMENTS', 'VÊTEMENTS'],
    'SOIN': ['SOIN'],
    'SANTE': ['SANTE'],
    'HAUT': ['HAUT', 'TOP'],
    'LINGERIE': ['LINGERIE'],
    'MINCEUR NUIT': [
        'LEGGING AMINCISSANT NUIT', 'PANTACOURT AMINCISSANT NUIT',
        'CORSAIRE AMINCISSANT NUIT', 'PANTY AMINCISSANT NUIT',
        'ENSEMBLE AMINCISSANT NUIT'
    ],
    'MINCEUR JOUR': [
        'LEGGING AMINCISSANT', 'PANTACOURT AMINCISSANT', 'CORSAIRE AMINCISSANT',
        'PANTY AMINCISSANT', 'BODY AMINCISSANT', 'COLLANT AMINCISSANT',
        'TOP & TEE SHIRT AMINCISSANT', 'ENSEMBLE AMINCISSANT'
    ],
    'LINGERIE GAINANTE': [
        'GAINE AMINCISSANTE', 'CULOTTE GAINANTE', 'BODY GAINANT',
        'LEGGING GAINANT', 'COLLANT GAINANT', 'STRING GAINANT',
        'MAILLOT DE BAIN GAINANT', 'PANTY PUSH UP GAINANT', 'TEE-SHIRT GAINANT'
    ],
    'ANTI-CELLULITE': [
        'LEGGING ANTICELLULITE', 'PANTACOURT ANTICELLULITE', 'CORSAIRE
        ↪  ANTICELLULITE',
        'PANTY ANTI CELLULITE', 'ACCESSOIRES ANTICELLULITE'
    ],
    'RAFFERMISSANT BUSTE': [
        'BRASSIÈRE RAFFERMISSANTE', 'TOP ET CARACO RAFFERMISSANT'
    ],
    'SPORT': [
        'LEGGING ET BAS SPORT GAINANT', 'TOP ET BRASSIÈRE SPORT RAFFERMISSANT',
        'ACCESSOIRE SPORT HYDRATANT ET RÉCUPÉRATION'
    ],
    'GRANDES TAILLES': ['GRANDES TAILLES'],
    'MATERNITE': ['MATERNITE'],
    'HOMME': ['HOMME', 'MALE'],
    'FEMME': ['FEMME', 'FEMALE'],
    'UNISEX': ['UNISEX'],
    'ENFANT': ['ENFANT'],
    'ADULT': ['ADULT'],
}
```

```
# Create binary columns for categories and subcategories
for category, keywords in categories.items():
    regex_pattern = '|'.join([re.escape(keyword) for keyword in keywords])
    df_products[f'{category.replace(" ", "_")}'] =
    ↪  df_products['combined_text'].str.contains(regex_pattern)

# Ensure that if a subcategory is checked, the category is also checked
for category, subcategories in categories.items():
    if len(subcategories) > 1:  # Only for categories with subcategories
        category_col = f'{category.replace(" ", "_")}'
        for subcategory in subcategories:
            subcategory_col = f'{subcategory.replace(" ", "_")}'
            if subcategory_col in df_products.columns:
                df_products[category_col] = df_products[category_col] |
                ↪  df_products[subcategory_col]
```

Listing 19: Binary Category Columns Creation in `df_products`

Final Preprocessing Steps: After extracting the categories, the `combined_text` column and other unnecessary text columns were dropped. The `option1_value` column was renamed to `Color`, and the `option2_value` column was filled with empty strings and renamed to `Size`. Finally, one-hot encoding was applied to the `Color` and `Size` columns to convert them into numerical features.

```
# Drop the combined text column and unnecessary columns
df_products.drop(columns=['combined_text', 'handle', 'title', 'body_html',
↪  'product_category', 'type', 'tags', 'seo_title',
↪  'google_shopping_product_category', 'google_shopping_gender',
↪  'google_shopping_age_group', 'google_shopping_custom_label_0'], inplace=True)

# Rename and process options
df_products.rename(columns={'option1_value': 'Color'}, inplace=True)
df_products['Size'] = df_products['option2_value'].fillna('')
df_products.drop(columns=['option2_value'], inplace=True)

# Apply one-hot encoding
df_products = pd.get_dummies(df_products, columns=['Color', 'Size'])
```

Listing 20: Final Preprocessing Steps in `df_products`

Through this process, the categorical features in the `df_products` dataset were effectively transformed into a format suitable for modeling, ensuring that the data could be efficiently utilized in subsequent machine learning tasks.

Numerical Features in `df_products`

The ‘df_products‘ dataset contains several numerical features that required transformation to ensure better modeling performance and interpretability. In this subsection, we focus on three key numerical columns: `variant_price`, `variant_inventory_qty`, and `variant_grams`. The skewness and kurtosis values of these features were computed to assess their distribution characteristics.

- `variant_price`:
  - Skewness: -0.376
  - Kurtosis: 0.249
- `variant_inventory_qty`:
  - Skewness: 4.161
  - Kurtosis: 22.315
- `variant_grams`:
  - Skewness: 1.123
  - Kurtosis: 2.100

These values indicate that while `variant_price` is relatively symmetrical with mild kurtosis, both `variant_inventory_qty` and `variant_grams` exhibit significant skewness and kurtosis, indicating the presence of outliers and asymmetry in the data distribution.

To address these issues, several transformations were applied to each column, including logarithmic (Log), square root (Sqrt), Box-Cox, Yeo-Johnson, Min-Max normalization, and Standardization. The distributions of the original and transformed features were visualized to identify the most effective transformation for each column. Figure 5.1 provides a comprehensive view of these distributions.

Figure 5.1: Distribution of Numerical Features in `df_products` Before and After Transformation

Based on the visual inspection and analysis of the transformed distributions, the following transformations were selected as the best fit for each column:

- `variant_price`: Yeo-Johnson Transformation
- `variant_inventory_qty`: Box-Cox Transformation
- `variant_grams`: Box-Cox Transformation

Following the selection of the optimal transformations, the original columns, along with other non-optimal transformed columns, were removed from the dataset. This step ensured that only the most effective representations of the numerical features were retained for subsequent modeling processes.

5.2.3 Feature Engineering of orders' dataset

Categorical Features in `df_orders`

In the `df_orders` dataset, several categorical features were derived and processed to enhance the dataset's utility for modeling. This subsection outlines the steps taken to extract relevant temporal features from the order creation timestamps and the encoding of categorical columns.

Extraction of Temporal Features The `created_at` column in the dataset, which records the timestamp of each order's creation, was first converted to a datetime object, taking into account the timezone offset. From this datetime object, various temporal components were extracted to create new features that could capture time-based patterns in the data:

- `created_year`: Extracts the year in which the order was created.
- `created_month`: Extracts the month of the year (from 1 to 12) in which the order was created.
- `created_day_of_week`: Extracts the day of the week (from 0 for Monday to 6 for Sunday) on which the order was created.
- `created_week_of_year`: Extracts the week of the year (from 1 to 52/53) in which the order was created.
- `created_quarter`: Extracts the quarter of the year (from 1 to 4) in which the order was created.

```
# Convert the 'created_at' column to a datetime object (handling the timezone
↪  offset)
df_orders['created_at'] = pd.to_datetime(df_orders['created_at'], utc=True)

# Extracting different components from the datetime object
df_orders['created_year'] = df_orders['created_at'].dt.year
df_orders['created_month'] = df_orders['created_at'].dt.month
df_orders['created_day_of_week'] = df_orders['created_at'].dt.dayofweek # from 0
↪  (monday) to 6
df_orders['created_week_of_year'] = df_orders['created_at'].dt.isocalendar().week
df_orders['created_quarter'] = df_orders['created_at'].dt.quarter

df_orders.describe(include='all')
```

Listing 21: Extraction of Temporal Features from `df_orders`

After these features were successfully extracted, the original `created_at` column was no longer needed and was therefore removed from the dataset.

One-Hot Encoding of Categorical Columns The `lineitem_fulfillment_status` column, which indicates the fulfillment status of the line items in an order, was one-hot encoded. This encoding transformed the categorical data into a set of binary columns, each representing a different fulfillment status. The resulting columns were prefixed with `fulfillment_` to clearly denote their origin.

```
# One-hot encode 'lineitem_fulfillment_status'
df_orders = pd.get_dummies(df_orders, columns=['lineitem_fulfillment_status'],
↪  prefix='fulfillment')
```

Listing 22: One-hot encode 'lineitem_fulfillment_status' from `df_orders`

This preprocessing step was crucial for converting the categorical information into a numerical format suitable for machine learning algorithms. The one-hot encoding process effectively expanded the feature space, enabling the model to capture distinctions between different fulfillment statuses.

Numerical Features in `df_orders`

In this section, we focus on the numerical features present in the `df_orders` dataset. We performed various operations to transform and analyze these features, including the calculation of skewness and kurtosis, applying different transformations, and selecting the most appropriate transformation for each feature.

Total Spend Per Line Item: The total spend per line item was calculated by multiplying the `lineitem_quantity` by the `lineitem_price`. This value was stored in a new column called `total_spend_per_order`.

```
# Calculate the total spend per line item
df_orders['total_spend_per_order'] = df_orders['lineitem_quantity'] *
↪  df_orders['lineitem_price']
```

Listing 23: Calculation of Total Spend Per Line Item

Analysis of Skewness and Kurtosis: We analyzed the skewness and kurtosis for three key numerical features in the dataset: `lineitem_price`, `lineitem_discount`, and `total_spend_per_order`. The results were as follows:

- **lineitem_price**: Skewness = 0.5801, Kurtosis = 0.0160
- **lineitem_discount**: Skewness = 27.0087, Kurtosis = 1152.9615
- **total_spend_per_order**: Skewness = 8.0798, Kurtosis = 230.9065

Data Transformation and Visualization: To handle the skewness and kurtosis observed in the features, we applied several transformations, including Log, Square Root, Box-Cox, Yeo-Johnson, Min-Max Normalization, and Standardization. The following code was used to visualize the distribution of the original and transformed data:

```python
def plot_distributions(df, columns, transformations):
    fig, axes = plt.subplots(len(columns), len(transformations) + 1, figsize=(20,
    ↪  15))
    for i, col in enumerate(columns):
        sns.histplot(df[col], ax=axes[i, 0], kde=True)
        axes[i, 0].set_title(f'Original {col}')
        for j, (trans_name, trans_col) in enumerate(transformations.items()):
            sns.histplot(df[trans_col[i]], ax=axes[i, j + 1], kde=True)
            axes[i, j + 1].set_title(f'{trans_name} {col}')
    plt.tight_layout()
    plt.show()

# Columns to transform
columns = ['lineitem_price', 'lineitem_discount', 'total_spend_per_order']

transformations = {
    'Log': ['log_lineitem_price', 'log_lineitem_discount',
            'log_total_spend_per_order'],
    'Sqrt': ['sqrt_lineitem_price', 'sqrt_lineitem_discount',
             'sqrt_total_spend_per_order'],
    'BoxCox': ['boxcox_lineitem_price', 'boxcox_lineitem_discount',
               'boxcox_total_spend_per_order'],
    'YeoJohnson': ['yeojohnson_lineitem_price', 'yeojohnson_lineitem_discount',
                   'yeojohnson_total_spend_per_order'],
    'MinMax': ['minmax_lineitem_price', 'minmax_lineitem_discount',
               'minmax_total_spend_per_order'],
    'Standard': ['standard_lineitem_price', 'standard_lineitem_discount',
                 'standard_total_spend_per_order']
}

# Plot the distributions
plot_distributions(df_orders, columns, transformations)
```

Listing 24: Distribution of Original and Transformed Numerical Features in `df_orders`

The figure (figure reference here) shows the distributions of the original and transformed features. After reviewing the transformed distributions, the best transformation for each column was selected:

- **lineitem_price**: Square Root Transformation
- **lineitem_discount**: Box-Cox Transformation
- **total_spend_per_order**: Log Transformation

Dropping Unnecessary Columns: After selecting the best transformations, we dropped the original and unnecessary transformed columns from the dataset to streamline the data.

```
# Drop multiple columns
df_orders.drop(['lineitem_price', 'lineitem_discount', 'total_spend_per_order',
                'log_lineitem_price', 'log_lineitem_discount',
                'sqrt_lineitem_discount', 'sqrt_total_spend_per_order',
                'boxcox_lineitem_price', 'boxcox_total_spend_per_order',
                'yeojohnson_lineitem_price', 'yeojohnson_lineitem_discount',
                'yeojohnson_total_spend_per_order', 'minmax_lineitem_price',
                'minmax_lineitem_discount', 'minmax_total_spend_per_order',
                'standard_lineitem_price', 'standard_lineitem_discount',
                'standard_total_spend_per_order'], axis=1, inplace=True)
```

Listing 25: Dropping Unnecessary Columns from `df_orders`

Conclusion

After performing feature engineering on both categorical and numerical features across the three datasets, we have prepared the data for further modeling and analysis. The cleaned and transformed datasets were saved in the "data/prepared_data/" directory as CSV files. This will allow for efficient and accurate analysis in subsequent stages of the project.

6 Modeling and Evaluation

6.1 Content-Based Recommendation System

Content-Based Recommendation Systems focus on analyzing the properties of items themselves rather than the interaction between users and items. In this project, we use four similarity methods to compute the similarity between user profiles and product features. These methods are Cosine Similarity, Euclidean Distance, Pearson Correlation, and Mahalanobis Distance.

6.1.1 Cosine Similarity Method

Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space. It is particularly useful in high-dimensional spaces where the magnitude of vectors can vary widely. The formula for cosine similarity is:

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \cdot ||\mathbf{B}||}$$

Where $\mathbf{A}$ and $\mathbf{B}$ are vectors of user and product features.

```
def cosine_similarity_method(user_feature_df, product_features_df):
    return cosine_similarity(user_feature_df, product_features_df)
```

Listing 26: Cosine Similarity Method

Cosine similarity captures the orientation of the vectors, making it robust to different magnitudes. Ideal when the scale of the features varies greatly but the relative importance (direction) remains the same.

6.1.2 Euclidean Similarity Method

Euclidean distance is the straight-line distance between two points in a multi-dimensional space. The similarity is inversely related to the Euclidean distance:

$$\text{Euclidean Similarity} = \frac{1}{1 + \text{Euclidean Distance}}$$

This method captures the absolute differences in feature values.

```
def euclidean_similarity_method(user_feature_df, product_features_df):
    euclidean_dist = euclidean_distances(user_feature_df, product_features_df)
    return 1 / (1 + euclidean_dist)
```

Listing 27: Euclidean Similarity Method

Euclidean similarity provides a measure based on actual distances, making it sensitive to the magnitude of the differences. Useful when absolute differences in feature values are important.

6.1.3 Pearson Similarity Method

Pearson correlation measures the linear correlation between two sets of data. It is defined as the covariance of the variables divided by the product of their standard deviations:

$$\text{Pearson Similarity} = \frac{\text{Cov}(\mathbf{A}, \mathbf{B})}{\sigma_A \sigma_B}$$

This method captures the degree to which two variables change together.

```
def pearson_similarity_method(user_features, product_features):
    user_features_centered = user_features - np.mean(user_features, axis=1,
    ↪  keepdims=True)
    product_features_centered = product_features - np.mean(product_features,
    ↪  axis=1, keepdims=True)

    numerator = np.dot(user_features_centered, product_features_centered.T)
    denominator = np.sqrt(np.sum(user_features_centered**2, axis=1,
    ↪  keepdims=True)) @ np.sqrt(np.sum(product_features_centered**2, axis=1,
    ↪  keepdims=True)).T
    similarity = np.divide(numerator, denominator, out=np.zeros_like(numerator),
    ↪  where=denominator != 0)
    return similarity
```

Listing 28: Pearson Similarity Method

Pearson similarity captures linear relationships between features, focusing on how two variables change together. Ideal for scenarios where correlation between features is significant.

6.1.4 Mahalanobis Similarity Method

Mahalanobis distance is a measure of the distance between a point and a distribution. Unlike Euclidean distance, it accounts for correlations between variables and is scale-invariant:

$$\text{Mahalanobis Distance} = \sqrt{(\mathbf{A} - \mathbf{B})^T \mathbf{C}^{-1} (\mathbf{A} - \mathbf{B})}$$

Where $\mathbf{C}$ is the covariance matrix of the distribution.

```python
def mahalanobis_similarity_method(user_features, product_features):
    try:
        covariance_matrix = np.cov(product_features.T)
        inverse_covariance_matrix = np.linalg.pinv(covariance_matrix)
    except LinAlgError as e:
        print(f"Error inverting covariance matrix: {e}")
        inverse_covariance_matrix = np.eye(product_features.shape[1])

    similarity = np.zeros((user_features.shape[0], product_features.shape[0]))
    for i in range(user_features.shape[0]):
        for j in range(product_features.shape[0]):
            try:
                similarity[i, j] = 1 / (1 + mahalanobis(user_features[i],
                ↪  product_features[j], inverse_covariance_matrix))
            except ValueError:
                similarity[i, j] = 0
    return similarity
```

Listing 29: Mahalanobis Similarity Method

Mahalanobis similarity takes into account the correlation between variables, providing a more informed distance metric. Suitable for datasets where features are correlated.

6.1.5 Evaluation and Comparison of Methods

To evaluate the performance of the different similarity methods, we used precision, recall, and F1 score as metrics. Precision measures the accuracy of the recommendations, recall measures the completeness, and F1 score balances both precision and recall.

```python
def evaluate_recommendations(recommendations, test_users, test_data, k=5):
    precisions, recalls, f1_scores = [], [], []

    for user in test_users:
        if user in test_data.index:
            actual_products = set(test_data[user])
            recommended_products = set(recommendations.get(user, [])[:k])

            true_positives =
            ↪  len(actual_products.intersection(recommended_products))
            false_positives = len(recommended_products) - true_positives
            false_negatives = len(actual_products) - true_positives

            precision = true_positives / (true_positives + false_positives) if
            ↪  true_positives + false_positives > 0 else 0
            recall = true_positives / (true_positives + false_negatives) if
            ↪  true_positives + false_negatives > 0 else 0
            f1 = 2 * precision * recall / (precision + recall) if precision +
            ↪  recall > 0 else 0

            precisions.append(precision)
            recalls.append(recall)
            f1_scores.append(f1)

    return np.mean(precisions), np.mean(recalls), np.mean(f1_scores)
```

Listing 30: Evaluation of Recommendation Methods

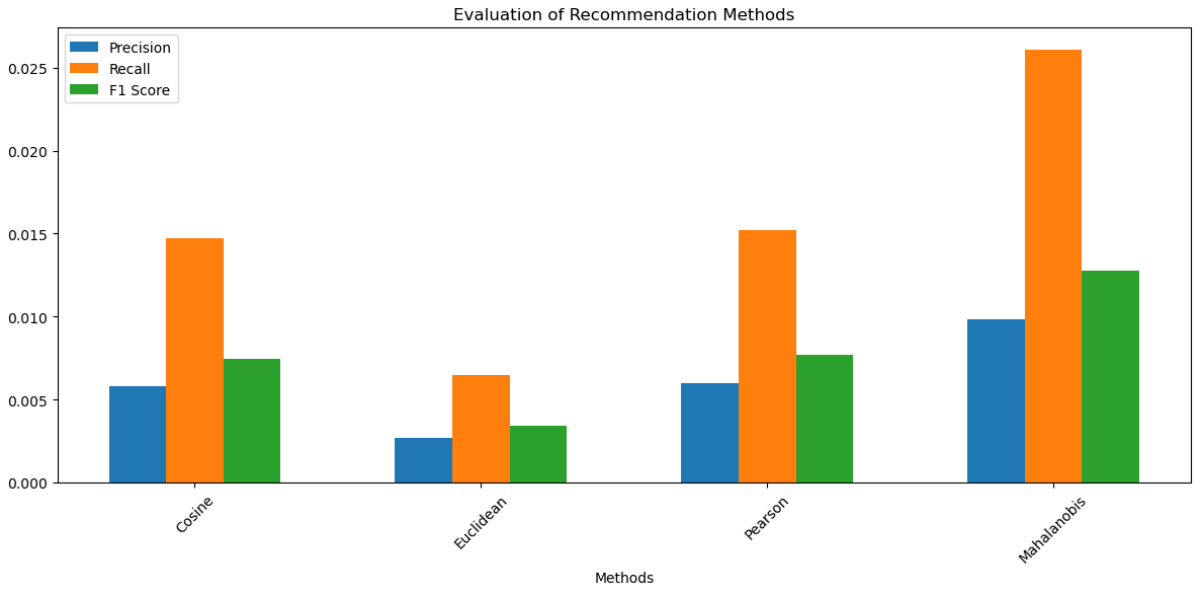


Figure 6.1: Evaluation of Recommendation Methods

6.1.6 Conclusion

Each of these methods offers unique advantages, and their performance was evaluated using standard metrics to determine the most effective approach for our recommendation system.

The evaluation of the four similarity methods—Cosine, Euclidean, Pearson, and Ma-

halanobis—reveals that none of the methods achieved high performance in terms of precision, recall, or F1 score. However, among the methods tested, the Mahalanobis similarity method outperformed the others, with the highest precision (0.0098), recall (0.0261), and F1 score (0.0128).

This suggests that while Mahalanobis distance might be slightly better suited for capturing the relationships between user profiles and product features in this dataset, the overall effectiveness of content-based recommendations using these similarity measures is limited. The low performance metrics indicate that the system may struggle to provide highly relevant recommendations based solely on the product features and the methods tested. Further optimization or consideration of alternative approaches may be necessary to improve the recommendation quality.

6.2 Collaborative Filtering Recommendation System

Collaborative Filtering is a widely-used recommendation technique that relies on the interactions between users and items. The primary assumption behind collaborative filtering is that users who have agreed in the past will continue to agree in the future. This section focuses on User-Based Collaborative Filtering, which suggests items to users based on the preferences of similar users.

6.2.1 User-Based Collaborative Filtering

User-Based Collaborative Filtering identifies similar users and recommends items that those similar users have interacted with. The similarity between users can be calculated using various methods. In this project, we explored three different similarity measures: Cosine Similarity, Pearson Correlation, and k-Nearest Neighbors (KNN).

Cosine Similarity

Cosine Similarity measures the cosine of the angle between two non-zero vectors, representing the interaction between users and items. A similarity score close to 1 indicates that the two users have similar preferences, while a score close to 0 indicates dissimilar preferences.

To implement Cosine Similarity, a user-item interaction matrix was created where rows represent users and columns represent items. Each cell in the matrix contains the number of times a user interacted with a particular item. The Cosine Similarity between users was computed as follows:

```
from sklearn.metrics.pairwise import cosine_similarity
user_similarity_cosine = cosine_similarity(user_item_matrix)
```

Listing 31: cosine_similarity

The user-item interaction matrix was then used to identify similar users and recommend items. The evaluation of the Cosine Similarity method resulted in the following

metrics:

- **Precision**: 0.1933
- **Recall**: 0.9953
- **F1-Score**: 0.3001

Pearson Correlation

Pearson Correlation measures the linear correlation between two users' interaction vectors. A Pearson correlation coefficient close to 1 indicates that two users have similar preferences, while a coefficient close to -1 indicates dissimilar preferences.

To calculate Pearson Correlation, a sample of the user-item interaction matrix was transposed, and the correlation was computed as follows:

```
user_sample = user_item_matrix.sample(n=10000)
user_similarity_pearson_sample = user_sample.T.corr()
```

Listing 32: Pearson Correlation

Similar to Cosine Similarity, the user-item interaction matrix and the Pearson similarity scores were used to generate recommendations. The evaluation of the Pearson Correlation method produced the following metrics:

- **Precision**: 0.1918
- **Recall**: 0.9956
- **F1-Score**: 0.2988

k-Nearest Neighbors (KNN)

The k-Nearest Neighbors (KNN) approach identifies the top k users who are most similar to a given user based on their interaction patterns. In this project, we used the Cosine Similarity metric to find the nearest neighbors, and the number of neighbors was set to 10.

The implementation involved fitting a KNN model to a sample of the user-item interaction matrix:

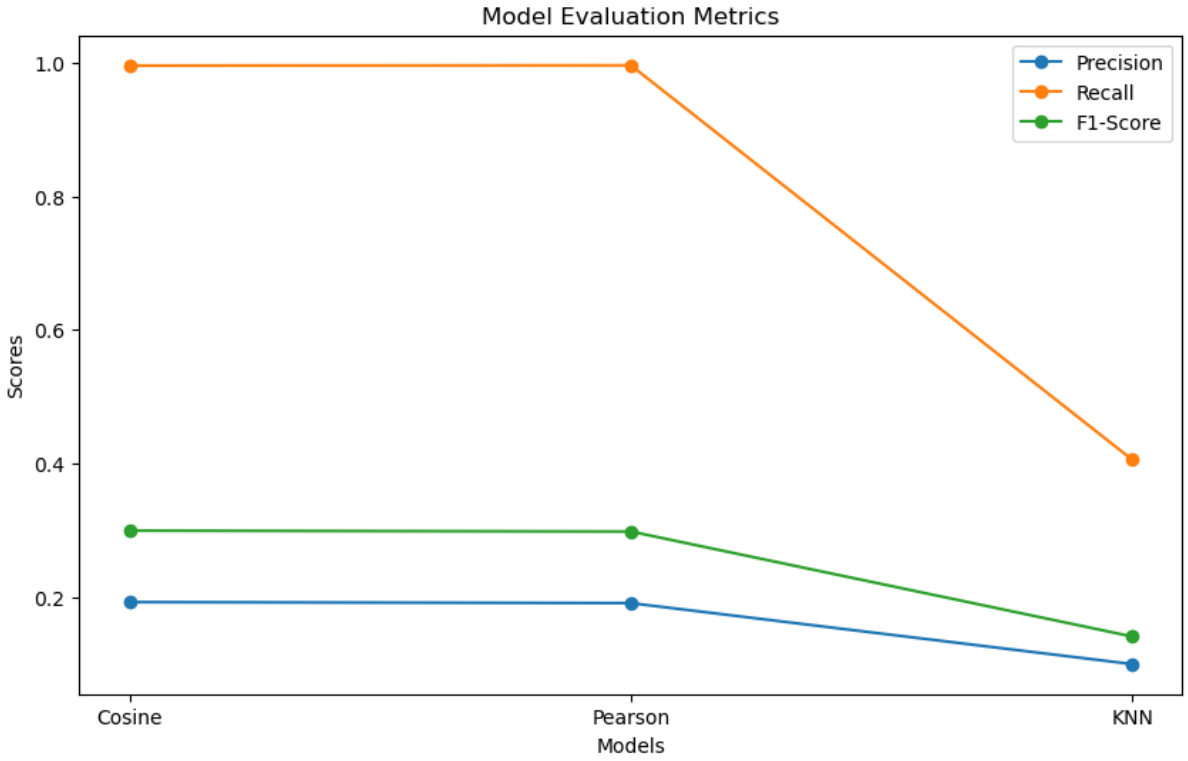
```
from sklearn.neighbors import NearestNeighbors

knn = NearestNeighbors(metric='cosine', algorithm='brute')
user_item_matrix_sample = user_item_matrix.sample(n=5000)
knn.fit(user_item_matrix_sample)
distances, indices = knn.kneighbors(user_item_matrix_sample, n_neighbors=10)
```

Listing 33: KNN

For each user, the KNN model identified the most similar users, and items were recommended based on the items interacted with by those similar users. The evaluation of the KNN method resulted in the following metrics:

- **Precision**: 0.10
- **Recall**: 0.4061
- **F1-Score**: 0.1418

Conclusion for Evaluation and Comparison of Methods



Evaluation of Recommendation Methods

The figure above illustrates the Precision, Recall, and F1-Score for the three methods evaluated: Cosine Similarity, Pearson Correlation, and KNN.
After evaluating the various methods used in the User-Based Collaborative Filtering approach, the results reveal distinct performances across the different algorithms:

1. **Cosine Similarity** produced the highest recall among all methods. However, its precision and F1-Score indicate that while it retrieves a large number of relevant items, many of these recommendations may not be accurate.
2. **Pearson Correlation** performed similarly to Cosine Similarity, achieving high recall but lower precision and F1-Score. This suggests that while Pearson Correlation is effective in identifying items that users are likely to interact with, the recommendations may lack precision.
3. **K-Nearest Neighbors (KNN)**, using Cosine distance, although yielding lower recall compared to the other two methods, achieved a more balanced performance

in terms of precision and F1-Score. This indicates that KNN may provide more accurate recommendations, albeit fewer in number.

In conclusion, **none of the methods stand out as ideal** for this specific dataset and problem context. While the high recall from Cosine and Pearson methods may be beneficial in certain scenarios, the trade-off with precision suggests that these models may generate a significant amount of noise in the recommendations. KNN, with its more balanced metrics, appears to be the most reliable method in terms of accuracy, though it may miss some potential recommendations. Therefore, the choice of method should depend on the specific goals of the recommendation system—whether the focus is on maximizing the number of relevant items retrieved (recall) or on ensuring that the items recommended are highly relevant to the user (precision).

6.2.2 Item-Based Collaborative Filtering

Implementation of Item-Based Collaborative Filtering

In this subsection, we implement an Item-Based Collaborative Filtering recommendation system using two similarity measures: **Cosine Similarity** and **Pearson Correlation**. The following sections detail the process and results of these implementations.

Matrix Construction: To begin, we constructed an item-user interaction matrix by transposing the user-item matrix derived from the `df_orders` dataset. Given the potential size of the data, a sample of this matrix was optionally used to facilitate computation.

```
# Create an item-user interaction matrix (transpose of the user-item matrix)
item_user_matrix = user_item_matrix.T
item_sample = item_user_matrix.sample(n=10000, axis=1)  # Adjust sample size
```

Listing 34: Item-User Interaction Matrix Construction

Cosine Similarity

Next, we computed the Cosine Similarity between items. This measure captures the cosine of the angle between two non-zero vectors in the item-user matrix, effectively identifying items with similar user interaction patterns.

```
from sklearn.metrics.pairwise import cosine_similarity
item_similarity_cosine = cosine_similarity(item_user_matrix)
```

Listing 35: Cosine Similarity Calculation

Pearson Correlation

We also computed the Pearson Correlation between items using the sampled item-user interaction matrix. Pearson Correlation measures the linear relationship between items, offering a statistical perspective on item similarity.

```
item_similarity_pearson = item_sample.corr()
```

Listing 36: Pearson Correlation Calculation

Evaluation Metrics

To evaluate the effectiveness of the Item-Based Collaborative Filtering models, we employed three key metrics: Precision, Recall, and F1-Score. These metrics were calculated by assessing the top N recommendations for each item.

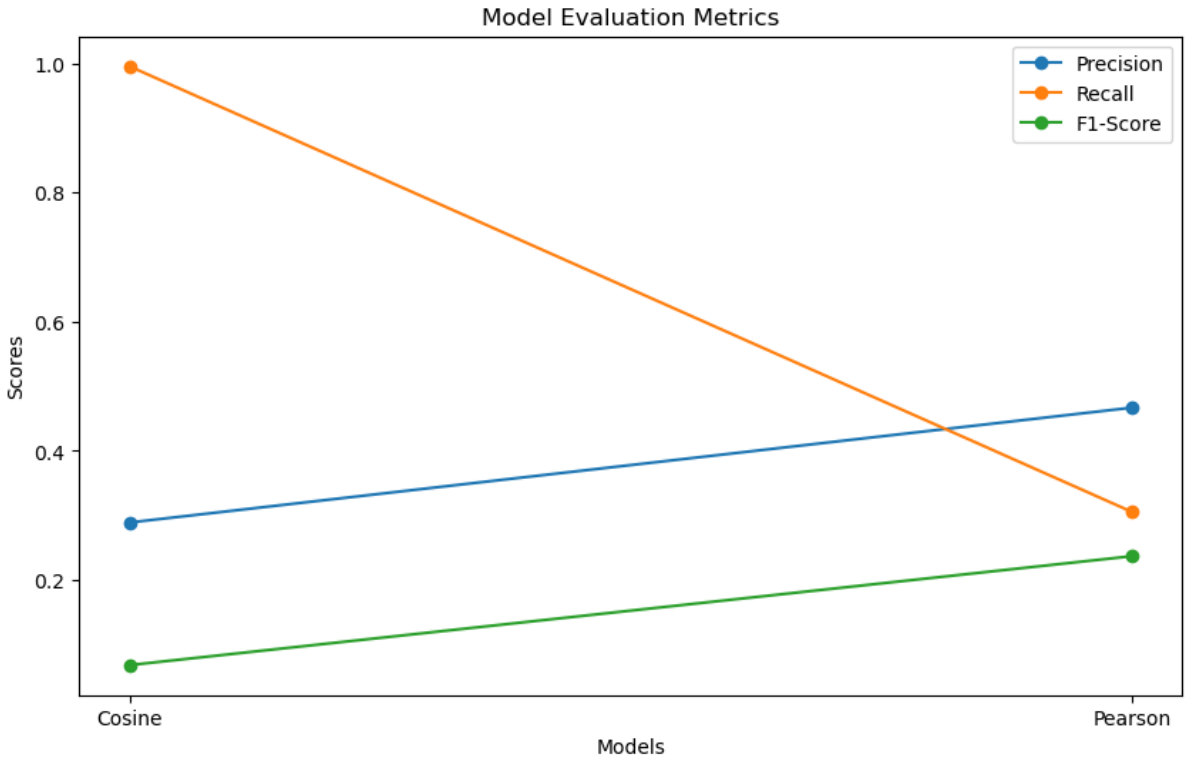
```
precision_cosine_item, recall_cosine_item, f1_cosine_item =
↪  evaluate_item_based_model(item_similarity_cosine, item_user_matrix, top_n=10)
precision_pearson_item, recall_pearson_item, f1_pearson_item =
↪  evaluate_item_based_model(item_similarity_pearson.values, item_sample.values,
↪  top_n=10)
```

Listing 37: Evaluation of Item-Based Collaborative Filtering

Results The evaluation results for the Cosine and Pearson methods are summarized as follows:

- **Cosine Similarity:**
  - Precision: 0.2889
  - Recall: 0.0700
  - F1-Score: 0.0682
- **Pearson Correlation:**
  - Precision: 0.4668
  - Recall: 0.3052
  - F1-Score: 0.2369

Conclusion for Evaluation and Comparison of Methods



Evaluation of Recommendation Methods

The figure above illustrates the Precision, Recall, and F1-Score for the two methods evaluated: Cosine Similarity and Pearson Correlation.

Based on the results, the **Pearson Correlation** method demonstrates superior performance compared to **Cosine Similarity**. Specifically, Pearson Correlation achieved a Precision of 0.4668, Recall of 0.3052, and an F1-Score of 0.2369, indicating it is the more effective method for Item-Based Collaborative Filtering in this context.

However, despite Pearson's better performance, the relatively low Recall and F1-Score suggest that further optimization or alternative approaches may be necessary to improve the recommendation quality. Both methods could benefit from additional tuning or the integration of complementary algorithms to enhance their effectiveness.

6.2.3 Model-Based Collaborative Filtering

Implementation of Model-Based Collaborative Filtering

In this subsection, we implement a Model-Based Collaborative Filtering recommendation system using three techniques: **Singular Value Decomposition (SVD)**, **Alternating Least Squares (ALS)**, and **Factorization Machines (FM)**. These techniques model the latent factors that influence user-item interactions, enabling the prediction of user preferences.

Matrix Construction: We first constructed the user-item interaction matrix using the `df_orders` dataset. This matrix forms the basis for both SVD and ALS model training.

```
# Create the user-item interaction matrix
user_item_matrix = df_orders.pivot_table(index='customer_id',
↪  columns='lineitem_sku', aggfunc='size', fill_value=0)
```

Listing 38: User-Item Interaction Matrix Construction

Singular Value Decomposition (SVD)

We trained an SVD model using the `Surprise` library, which is commonly used in recommendation system research. SVD decomposes the user-item interaction matrix into latent factors, enabling the prediction of user preferences.

```
from surprise import Dataset, Reader, SVD, accuracy
from surprise.model_selection import train_test_split

# Prepare the data for Surprise
reader = Reader(rating_scale=(1, user_item_matrix.max().max()))
data = Dataset.load_from_df(df_orders[['customer_id', 'lineitem_sku',
↪  'lineitem_quantity']], reader)
trainset, testset = train_test_split(data, test_size=0.2)

# Train an SVD model
model_svd = SVD()
model_svd.fit(trainset)

# Evaluate the model
predictions = model_svd.test(testset)
precision_svd, recall_svd, f1_svd = evaluate_predictions(predictions)
print(f"SVD - Precision: {precision_svd:.4f}, Recall: {recall_svd:.4f}, F1-Score:
↪  {f1_svd:.4f}")
```

Listing 39: SVD Model Training and Evaluation

Alternating Least Squares (ALS)

Next, we implemented the ALS algorithm using the `implicit` library. ALS is particularly well-suited for collaborative filtering tasks involving implicit feedback data, as it alternates between optimizing user and item factors.

```
import scipy.sparse as sp
import implicit
from sklearn.decomposition import TruncatedSVD

# Convert user-item matrix to a sparse matrix
user_item_matrix_sparse = sp.csr_matrix(user_item_matrix.values)
user_ids = user_item_matrix.index
item_ids = user_item_matrix.columns

# Apply dimensionality reduction using SVD
svd = TruncatedSVD(n_components=50)  # Adjust n_components as needed
user_item_matrix_reduced = svd.fit_transform(user_item_matrix_sparse)

# Initialize and train the ALS model with increased regularization
model_als = implicit.als.AlternatingLeastSquares(factors=50, regularization=0.5,
↪  iterations=20)
model_als.fit(sp.csr_matrix(user_item_matrix_reduced))

# Evaluate the ALS model
precision_als, recall_als, f1_als = evaluate_als_model(model_als,
↪  user_item_matrix, sp.csr_matrix(user_item_matrix_reduced), top_n=10)
print(f"ALS - Precision: {precision_als:.4f}, Recall: {recall_als:.4f}, F1-Score:
↪  {f1_als:.4f}")
```

Listing 40: ALS Model Training and Evaluation

Evaluation Metrics

To evaluate the effectiveness of the Model-Based Collaborative Filtering models, we employed three key metrics: Precision, Recall, and F1-Score. These metrics were calculated by assessing the top N recommendations for each item.

```python
from surprise import Dataset, Reader, SVD, accuracy
from surprise.model_selection import train_test_split

# Prepare the data for Surprise
reader = Reader(rating_scale=(1, user_item_matrix.max().max()))
data = Dataset.load_from_df(df_orders[['customer_id', 'lineitem_sku',
↪  'lineitem_quantity']], reader)
trainset, testset = train_test_split(data, test_size=0.2)

# Train an SVD model
model_svd = SVD()
model_svd.fit(trainset)

# Evaluate the model
predictions = model_svd.test(testset)
precision_svd, recall_svd, f1_svd = evaluate_predictions(predictions)
print(f"SVD - Precision: {precision_svd:.4f}, Recall: {recall_svd:.4f}, F1-Score:
↪  {f1_svd:.4f}")

# Evaluate the ALS model
precision_als, recall_als, f1_als = evaluate_als_model(model_als,
↪  user_item_matrix, sp.csr_matrix(user_item_matrix_reduced), top_n=10)
print(f"ALS - Precision: {precision_als:.4f}, Recall: {recall_als:.4f}, F1-Score:
↪  {f1_als:.4f}")
```

Listing 41: Evaluation of Model-Based Collaborative Filtering

Results The evaluation results for the Singular Value Decomposition (SVD) and Alternating Least Squares (ALS) algorithms are summarized as follows:

- **Singular Value Decomposition (SVD):**
  - Precision: 0.6143
  - Recall: 0.4378
  - F1-Score: 0.5112
- **Alternating Least Squares (ALS):**
  - Precision: 0.0036
  - Recall: 0.0171
  - F1-Score: 0.0054

Conclusion for Evaluation and Comparison of Methods

Based on the results, the **Singular Value Decomposition (SVD):** method demonstrates the best performance so far.

Factorization Machines (FM)

To further explore the effectiveness of matrix factorization techniques, we implemented a Factorization Machines (FM) model using the `SGDRegressor` from `scikit-learn`. We tested two types of ratings: the actual line item quantities as ratings and binary ratings.

Data Preparation: The datasets were merged and preprocessed to be suitable for training the FM model. This included encoding categorical variables, imputing missing values, and standardizing numerical features.

```
# Merge df_orders with df_customers on 'customer_id'
df_merged = pd.merge(df_orders, df_customers, on='customer_id', how='left')

# Merge df_merged with df_products on 'lineitem_sku'
df_merged = pd.merge(df_merged, df_products, on='lineitem_sku', how='left')

# Encode categorical variables with Label Encoding
le = LabelEncoder()
for col in df_merged.select_dtypes(include=['object']).columns:
    df_merged[col] = le.fit_transform(df_merged[col].astype(str))

# Impute missing values with the mean of each column
imputer = SimpleImputer(strategy='mean')
df_merged = pd.DataFrame(imputer.fit_transform(df_merged),
↪  columns=df_merged.columns)

# Standardize numerical features
scaler = StandardScaler()
numerical_columns = df_merged.select_dtypes(include=['float64', 'int64']).columns
df_merged[numerical_columns] = scaler.fit_transform(df_merged[numerical_columns])

# Data preparation for lineitem_quantity as the rating
df_ratings_quantity = df_merged.copy()
df_ratings_quantity['rating'] = df_ratings_quantity['lineitem_quantity']

# Data preparation for binary rating
df_ratings_binary = df_merged.copy()
df_ratings_binary['rating'] = 1

# Drop irrelevant or redundant columns
df_ratings_quantity = df_ratings_quantity.drop(columns=['lineitem_quantity'])
df_ratings_binary = df_ratings_binary.drop(columns=['lineitem_quantity'])

# Train-test split for both datasets
train_data_quantity, test_data_quantity = train_test_split(df_ratings_quantity,
↪  test_size=0.2, random_state=42)
train_data_binary, test_data_binary = train_test_split(df_ratings_binary,
↪  test_size=0.2, random_state=42)

# Prepare datasets for FM model
X_train_quantity = train_data_quantity.drop(columns=['rating'])
y_train_quantity = train_data_quantity['rating']
X_test_quantity = test_data_quantity.drop(columns=['rating'])
y_test_quantity = test_data_quantity['rating']

X_train_binary = train_data_binary.drop(columns=['rating'])
y_train_binary = train_data_binary['rating']
X_test_binary = test_data_binary.drop(columns=['rating'])
y_test_binary = test_data_binary['rating']
```

Listing 42: Data Preparation for Factorization Machines

Model Training and Evaluation: We trained and evaluated the FM model using both rating types.

```
def train_evaluate_model(X_train, y_train, X_test, y_test, model_name):
    # Initialize the model
    model = SGDRegressor(loss='squared_error', max_iter=1000, tol=1e-3)

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)

    # Evaluate the model
    mse_train = mean_squared_error(y_train, y_pred_train)
    mse_test = mean_squared_error(y_test, y_pred_test)
    mae_test = mean_absolute_error(y_test, y_pred_test)
    r2_test = r2_score(y_test, y_pred_test)

    # Print evaluation metrics
    print(f"{model_name} - Training MSE: {mse_train}")
    print(f"{model_name} - Testing MSE: {mse_test}")
    print(f"{model_name} - Testing MAE: {mae_test}")
    print(f"{model_name} - Testing R^2 Score: {r2_test}")

    return model, y_pred_test

# Train and evaluate for both rating types
model_quantity, y_pred_test_quantity = train_evaluate_model(X_train_quantity,
↪  y_train_quantity, X_test_quantity, y_test_quantity, "FM with Quantity Rating")
model_binary, y_pred_test_binary = train_evaluate_model(X_train_binary,
↪  y_train_binary, X_test_binary, y_test_binary, "FM with Binary Rating")
```

Listing 43: FM Model Training and Evaluation

Evaluation Results: The following results were obtained from the FM model:

- **FM with Quantity Rating:**
  - Training MSE: 3.37e+22
  - Testing MSE: 2.81e+22
  - Testing MAE: 7789047901.07
  - Testing R^2 Score: -2.21e+22
- **FM with Binary Rating:**
  - Training MSE: 2.61e+22
  - Testing MSE: 1.33e+22
  - Testing MAE: 7074762773.44
  - Testing R^2 Score: 0.0

Discussion: The results from the FM model indicate that the implementation of matrix factorization for the given dataset presents challenges, particularly in achieving a high level of accuracy in predictions. The FM with binary rating performed better than the one with quantity ratings, but the overall effectiveness of this approach was limited in this context.

Based on the results, the Singular Value Decomposition (SVD) of Model-Based Collaborative Filtering method demonstrates superior performance compared to all other used methods. Specifically, SVD achieved a Precision of 0.6143, Recall of Recall: 0.4378, and an F1-Score of 0.5112, indicating it is the more effective method for this project. However, despite SVD's better performance, the relatively low Recall indicating that the model is missing some relevant items that it should recommendand, suggest that further optimization approaches may be necessary to improve the recommendation quality. Additionally, F1-Score which balances precision and recall, is 49.02%. This is a good metric to consider as it provides a single measure of the model's accuracy considering both false positives and false negatives.

6.3 Deployment

6.3.1 Deployment Objectives

The deployment phase aims to transition the recommendation system from a development environment into a fully operational web application. This phase ensures that the system meets the business objectives, integrates seamlessly with the existing e-commerce platform, and delivers personalized product recommendations to end-users. The primary objectives are to ensure system reliability, scalability, and performance while maintaining data security and user privacy.

6.3.2 Functional and Non-Functional Requirements

Functional Requirements

The functional requirements for the deployment of the recommendation system include:

- **User Authentication and Authorization:** Secure login and access control mechanisms to ensure that only authorized users can interact with the system.
- **Real-time Recommendation Generation:** The system must provide real-time product recommendations based on user interactions and historical data.
- **Data Synchronization:** Continuous synchronization of customer, order, and product data from the e-commerce platform to ensure that the recommendation engine has up-to-date information.
- **Recommendation Display:** Seamless integration of product recommendations into the user interface of the e-commerce website.

Non-Functional Requirements

The non-functional requirements address the system's performance, reliability, and security, including:

- **Scalability:** The system should handle increasing amounts of data and user requests without degradation in performance.
- **Performance:** The recommendation generation process should be optimized for low latency to ensure a smooth user experience.
- **Security:** The system must comply with data protection regulations, ensuring the confidentiality and integrity of user data.
- **Availability:** The system should be available with minimal downtime, ensuring continuous access to the recommendation engine.
- **Maintainability:** The system should be designed for easy maintenance and updates, allowing for future enhancements and bug fixes.

6.3.3 System Architecture

The proposed system architecture consists of several interconnected components that work together to deliver personalized recommendations:

- **Data Layer:** This layer includes the databases storing customer, product, and order information. It also contains the preprocessed datasets used for training the recommendation models.
- **Recommendation Engine:** This core component utilizes various machine learning models, including content-based filtering, collaborative filtering, and matrix factorization, to generate personalized recommendations.
- **API Layer:** The API layer provides an interface for the web application to interact with the recommendation engine. It handles incoming requests, processes data, and returns recommended products to the front-end.
- **Web Application Layer:** The front-end of the e-commerce platform, where recommendations are displayed to users. This layer also manages user interactions and sends data to the API layer for processing.
- **Monitoring and Logging:** Tools and services for monitoring system performance, tracking user interactions, and logging errors for troubleshooting and analysis.

6.3.4 Web Application Integration

The final phase involves the integration of the recommendation system into the existing e-commerce web application. This integration requires collaboration between the development team and the web application team to ensure seamless functionality and user experience. The key steps include:

- **Frontend Integration:** Embedding the recommendation widget into product pages, home pages, and other relevant sections of the e-commerce site.
- **Backend Integration:** Setting up API endpoints that allow the front-end to request and display recommendations based on user behavior and interactions.
- **User Testing and Feedback:** Conducting user acceptance testing (UAT) to gather feedback and make necessary adjustments before full deployment.
- **Deployment and Monitoring:** Rolling out the system to production, followed by continuous monitoring to ensure optimal performance and quick resolution of any issues.

7 General Conclusion

This report was prepared as part of our end-of-study internship for obtaining a Licence's degree in Computer Science, specializing in Software Engi-neering and Information Systems at ISI Mahdia,. The internship was carried out within the Web Development agency PerpetualCode.

The objective of this project was to develop a robust and effective recommendation system tailored to an e-commerce platform. The project encompassed various stages, including data collection, preprocessing, model selection, and evaluation, culminating in the deployment of the system within a real-world web application.

Throughout this process, different recommendation techniques were explored, including content-based filtering, collaborative filtering, and matrix factorization. Each method was rigorously tested and evaluated based on precision, recall, F1 score, and other relevant metrics. While some models demonstrated higher accuracy and relevance in recommendations, others provided valuable insights into user behavior and preferences.

This project represents a significant step forward in leveraging machine learning and data-driven approaches to drive business growth in e-commerce. By offering personalized product recommendations, the system not only improves customer satisfaction but also contributes to increased sales and customer retention.

In conclusion, the recommendation system developed in this project is a versatile and powerful tool that can be further expanded and refined. Future work could include exploring advanced algorithms, incorporating additional data sources, and continuously optimizing the system to adapt to changing user behaviors and market trends. The successful implementation of this project lays a solid foundation for future innovations in the intersection of artificial intelligence and e-commerce.

In conclusion, this project has led to the creation of a valuable tool for E-Commerce Web developpers. It constitutesis a versatile and powerful tool that can be further expanded and refined. Future work could include exploring advanced algorithms, incorporating additional data sources, and continuously optimizing the system to adapt to changing user behaviors and market trends. The successful implementation of this project lays a solid foundation for future innovations in the intersection of artificial intelligence and e-commerce.. This report provides a comprehensive synthesis of the various steps undertaken, highlighting the results obtained and the avenues for future improvements..

8 Webography

1: 18:05:2024: Retrieved from netapp: https://www.netapp.com/fr/artificialintelligence/what-is-artificial-intelligence/

2: 18:05:2024: (s.d.). Retrieved from talend: https://www.talend.com/fr/resources/what-is-machinelearning/(s.d.). Récupéré sur datascientest: https://datascientest.com/deep-learningdefinition

3: 18:05:2024: Retrieved from datascientest: https://datascientest.com/deep-learningdefinition

4: 20:05:2024: Retrieved from Nvidea: https://blogs.nvidia.com/blog/whats-a-recommender-system/

5:22:05:2024: Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/kdd-process-in-data-mining/

6: 22:05:2024: Retrieved from ibm: https://www.ibm.com/docs/fr/spss-modeler/saas?topic=dm-crisphelp-overview

Summary

This report details the development of a recommendation system tailored for an e-commerce platform. Utilizing a structured approach grounded in Artificial Intelligence (AI) principles and the CRISP-DM methodology, we meticulously analyzed and prepared datasets from Shopify, encompassing products, customers, and orders.

The project employed a variety of recommendation techniques, including content-based filtering, collaborative filtering, and factorization machines. The aim was to refine customer understanding, enhance predictive accuracy for future purchases, and drive business growth through personalized recommendations. The findings offer actionable insights specific to the operational context of the e-commerce site.

Abstract

This report provides a thorough investigation into creating an effective recommendation system for an e-commerce website. By applying Artificial Intelligence (AI) principles and following the CRISP-DM framework, we prepared and analyzed data sourced from Shopify, focusing on product, customer, and order information.

We implemented several recommendation approaches, such as content-based and collaborative filtering methods, as well as advanced factorization techniques. The project's goal was to gain insights into customer preferences and predict future buying behaviors, ultimately aiding in business expansion. The results deliver valuable, context-specific recommendations aimed at improving the user experience and operational efficiency of the e-commerce platform.