THE UNIVERSITY OF BAMENDA

NATIONAL HIGHER POLYTECHNIC INSTITUTE

DEPARTMENT OF COMPUTER ENGINEERING

DESIGN AND IMPLEMENTATION OF AN ECOMMERCE APPLICATION WITH A MACHINE LEARNING-BASED RECOMMENDATION SYSTEM

A Project Submitted to the Department of computer engineering in the National Higher Polytechnic Institute of The University of Bamenda in Partial Fulfilment of the Requirements for the Award of a Bachelor of Engineering Degree in computer engineering

BY:

TEDOM NOUTCHOGOUIN DONALD

REGISTRATION NUMBER: UBA19E0210

SUPERVISORS:

Dr. Fohagui Cyrille
Academic supervisor

Eng. Kunde Godfrey
Academic co-supervisor

ACCADEMY YEAR: 2022-2023

DECLARATION OF ORIGINALITY OF STUDY

I, **TEDOM NOUTCHOGOUIN DONALD**, registration N◦: **UBA19E0210**, in the Department of Computer Engineering, National Higher Polytechnic Institute, The University of Bamenda hereby declare that this work titled ***"Design and Implementation of an Ecommerce Application with a Machine Learning-Based Recommendation System***" is my original work. It has not been presented in any application for a degree or any academic pursuit elsewhere. I have acknowledged all borrowed ideas nationally and internationally through citations.

Date: ___________________ Signature of author ________________

CERTIFICATION OF CORRECTIONS AFTER DEFENSE

This is to certify that this project titled ***Design and Implementation of an Ecommerce Application with a Machine Learning-Based Recommendation System***" is the original work of **TEDOM NOUTCHOGOUIN DONALD**. This work is submitted in partial fulfilment of the requirements for the award of a Bachelor of Engineering Degree in Computer Engineering in the National Higher Polytechnic Institute of The University of Bamenda, Cameroon.

Supervisor: ____

Dr. Fohagui Cyrille

The Head of Department: ____

Dr. NDUKUM PASCALINE LIAKEN DICMU

The Director: ____

Prof. FIDELIS CHO-NGWA

ABSTRACT

In recent years, ecommerce applications in Cameroon have grown significantly, resulting in a wide array of products. However, improving user experience remains a major challenge for existing platforms, particularly with their categorical-based recommendation systems. These systems recommend products based on user purchases within a specific category, which has proven to be inefficient. With the rise of machine learning, alternative solutions offering improved performance have emerged. This report presents the development and evaluation of an ecommerce application with a machine learning-based recommendation system. The project aims to enhance personalized shopping experiences, increase company revenue, and improve customer satisfaction through accurate and relevant product recommendations. The recommendation system employs collaborative filtering, neural networks, content-based, and knowledge-based methods, trained on a dataset of 10,000 records from GroupLens capturing product ratings by hundreds of users in Cameroon. The evaluation indicates promising accuracy (60%), mean reciprocal rank (MRR) of 0.26, mean absolute error (MAE) of 0.214, root mean squared error (RMSE) of 0.370, precision of 0.8, and recall of 0.8. Recently, the ecommerce app was hosted at https://hooyia-market.onrender.com, attracting 13 new user registrations within a week. Where 45 products were added to the cart, with 15 directly influenced by the recommendation system, representing 33.3% of total cart additions. Additionally, the recommendation-driven purchases may lead to a 33.3% increase in sales. The findings demonstrate the hybrid recommendation system's effectiveness in providing accurate and relevant recommendations, significantly enhancing shopping experiences and user engagement in Cameroon. Overall, this project contributes to the field of ecommerce and machine learning, showcasing a successful implementation and evaluation of a hybrid recommendation system tailored for Cameroon. The results serve as a valuable reference for ecommerce companies seeking similar approaches to enhance customer experiences, increase revenue, and foster business growth.

Keywords: Ecommerce, Recommendation System, Hybrid Models, Cameroon, Sales Impact, Customer Satisfaction.

RÉSUMÉ

Ces dernières années, les applications de commerce électronique au Cameroun ont considérablement augmenté, offrant une large gamme de produits. Cependant, améliorer l'expérience utilisateur reste un défi majeur pour les plateformes existantes, notamment en ce qui concerne leurs systèmes de recommandation basés sur des catégories. Ces systèmes recommandent des produits en fonction des achats des utilisateurs dans une catégorie spécifique, ce qui s'est avéré inefficace. Avec l'avènement de l'apprentissage automatique, des solutions alternatives offrant de meilleures performances ont émergé. Ce rapport présente le développement et l'évaluation d'une application de commerce électronique avec un système de recommandation basé sur l'apprentissage automatique. Le projet vise à améliorer l'expérience d'achat personnalisée, augmenter les revenus de l'entreprise et améliorer la satisfaction des clients grâce à des recommandations de produits précises et pertinentes. Le système de recommandation utilise le filtrage collaboratif, les réseaux neuronaux, le contenu et les connaissances, entraînés sur un ensemble de données de 10 000 enregistrements de GroupLens représentant les évaluations de produits par des centaines d'utilisateurs au Cameroun. L'évaluation indique une précision prometteuse (60 %), un rang réciproque moyen (MRR) de 0,26, une erreur moyenne absolue (MAE) de 0,214, une erreur quadratique moyenne (RMSE) de 0,370, une précision de 0,8 et un rappel de 0,8. Récemment, l'application de commerce électronique a été hébergée sur https://hooyia-market.onrender.com, attirant 13 nouvelles inscriptions d'utilisateurs en une semaine. Parmi elles, 45 produits ont été ajoutés au panier, dont 15 ont été directement influencés par le système de recommandation, représentant 33,3 % des ajouts au panier. De plus, les achats générés par les recommandations pourraient entraîner une augmentation de 33,3 % des ventes. Les résultats démontrent l'efficacité du système de recommandation hybride pour fournir des recommandations précises et pertinentes, améliorant considérablement l'expérience d'achat et l'engagement des utilisateurs au Cameroun. Dans l'ensemble, ce projet contribue au domaine du commerce électronique et de l'apprentissage automatique, en présentant la mise en œuvre réussie et l'évaluation d'un système de recommandation hybride adapté au Cameroun. Les résultats servent de référence précieuse pour les entreprises de commerce électronique souhaitant adopter des approches similaires pour améliorer l'expérience client, augmenter les revenus et favoriser la croissance de l'entreprise.

Mots clés : Commerce électronique, Système de recommandation, Modèles hybrides, Cameroun, Impact sur les ventes, Satisfaction des clients.

DEDICATION

I dedicate this work to my beloved families: the KENGNE family and the TEDOM family.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to my supervisors, Dr. Fohagui Cyrille and Eng. Kunde Godfrey, for providing me with the invaluable opportunity to work alongside them. Their guidance and support have been instrumental in my professional development.

I would also like to extend my sincere appreciation to the Head of Department Dr. Ndukum Pascaline, the entire staff of the Computer Department, and the administration of NAHPI. Their assistance and cooperation have been crucial in realizing the successful completion of this project.

I am deeply thankful to my friends and colleagues for their unwavering support throughout the entire project. Their encouragement and collaboration have been truly uplifting.

Furthermore, I would like to express my profound gratitude to my family for their unwavering moral and financial support throughout this journey. Their belief in me and their constant encouragement have been a driving force.

Last but certainly not least, I am immensely grateful to the almighty God for granting me the strength and ability to successfully carry out this project.

TABLE OF CONTENT

LISTE OF TABLE

LISTE OF FIGURES

LIST OF ABREVIATIONS

RS:	Recommender System
CF:	Collaborative Filtering
CBF:	Content-Based Filtering
VSM:	Vector Space Model
TF-IDF:	Term Frequency-Inverse Document Frequency
SVD:	Singular Value Decomposition
ALS:	Alternating Least Squares
NMF:	Non-negative Matrix Factorization
MAE:	Mean Absolute Error
RMSE:	Root Mean Squared Error
MAP:	Mean Average Precision
HR:	Hit Rate
ARHR:	Average Reciprocal Hit Rank
AUC:	Area Under the Curve
ROC:	Receiver Operating Characteristic
IBCF:	Item-Based Collaborative Filtering
UBCF:	User-Based Collaborative Filtering
HRS:	Hybrid Recommender System
KBR:	Knowledge-Based Recommender System
FM:	Factorization Machines
DNN:	Deep Neural Network
CF-NN:	Collaborative Filtering Neural Network
CBF-NN:	Content-Based Filtering Neural Network
RNN:	Recurrent Neural Network
ERA:	Evolutionary Rank Aggregation
MERA:	Multi-objective Evolutionary Rank Aggregation
ReLU:	Rectified Linear Unit
LSTM:	Long Short-Term Memory
AWS:	Amazon Web Service
S3:	Simple Storage Service

CHAPTER 1: INTRODUCTION

1.1 Background of Study

The ecommerce industry in the world and especially in Cameroon has experienced rapid growth in recent years, with an increasing number of businesses (selling products, services, etc) moving towards online platforms. According to Statista, in 2021, retail e-commerce sales amounted to approximately 5.2 trillion U.S. dollars worldwide (Chevalier, 2022).

In the context of e-commerce in Cameroon, one significant challenge that needs to be addressed is the lack of efficient product recommendation systems in most e-commerce apps. This issue hinders the ability of online retailers to provide personalized shopping experiences to their customers.

Effective product recommendation systems play a crucial role in enhancing the overall shopping experience and driving customer engagement. These systems utilize algorithms that analyse customer preferences, purchase history, and browsing behaviour to suggest relevant and tailored product recommendations. By offering personalized recommendations, e-commerce platforms can assist customers in discovering new products, increasing their average order value, and fostering customer loyalty.

However, the current state of product recommendation systems in many e-commerce apps in Cameroon is inadequate. Customers often face difficulties in finding relevant products or discovering new items that align with their interests. This lack of personalized recommendations not only hampers the user experience but also limits the potential for cross-selling and upselling opportunities.
Improving the recommendation systems in e-commerce apps is crucial for enhancing customer satisfaction and driving business growth. By investing in advanced algorithms and data analysis techniques, online retailers can leverage customer data to provide more accurate and relevant product suggestions. This could include personalized recommendations based on previous purchases, browsing history, and customer demographics.
Furthermore, integrating machine learning and artificial intelligence technologies can enable e-commerce platforms to continuously optimize and refine their recommendation systems. By leveraging customer feedback and monitoring user interactions, these systems can adapt

and improve over time, ensuring that the recommendations become increasingly relevant and valuable to customers.
Addressing the deficiency in product recommendation systems is essential for the growth and competitiveness of e-commerce in Cameroon. By offering personalized and relevant product suggestions, online retailers can enhance the overall shopping experience, increase customer satisfaction, and drive higher conversion rates. This, in turn, can contribute to the further development and expansion of the e-commerce sector in the country.

The objective of this project was to design and implement an ecommerce application with a machine learning-based recommendation system which provided a personalized shopping experience for users in Cameroon. The project explored different machine learning models to determine the most effective recommendation system for the ecommerce application. The development of the ecommerce application was done using Python Django, Bootstrap, and jQuery, while Python and TensorFlow were used for the development of the recommender system.

The outcome of this project was a fully functional ecommerce application with a machine learning-based recommendation system that provides personalized product recommendations to customers in Cameroon. The results of this study will be useful for ecommerce companies in Cameroon that are interested in implementing a machine learning-based recommendation system in their applications. By providing a more personalized shopping experience, ecommerce companies can improve customer loyalty and gain a competitive advantage in the industry.

1.2 Problem statement

Despite the rapid growth of the ecommerce industry in Cameroon, ecommerce companies are struggling to provide personalized shopping experiences for their customers. Customers often find it challenging to search for products on ecommerce websites, which can lead to a poor shopping experience. Furthermore, customers may be presented with products that do not match their preferences, leading to frustration and decreased customer satisfaction.
This lack of personalization in ecommerce applications can be attributed to the inability of traditional ecommerce systems to analyse customer behaviour and provide tailored product recommendations. As a result, customers are left to navigate through a sea of products, with no guidance or personalized suggestions. This situation often results in customers leaving the

ecommerce site without making a purchase, leading to decreased revenue for ecommerce companies.

Therefore, the aim of this project was to develop an ecommerce application with a machine learning-based recommendation system which provided personalized shopping experiences for users. The proposed system utilizes customer data to analyse user behaviour and preferences, and use this information to generate tailored product recommendations. By providing a personalized shopping experience, the ecommerce application helps customers find products that match their preferences, thereby increasing the chances of them making a purchase.

The development of a machine learning-based recommendation system for an ecommerce application presented several challenges. Firstly, the system should be able to analyse large volumes of customer data to generate accurate recommendations. Secondly, the system should be able to adapt to changing customer preferences over time. Finally, the system should be integrated seamlessly into the ecommerce application, ensuring that it does not disrupt the user experience.

The successful development of the proposed system will enable ecommerce companies in Cameroon to provide personalized shopping experiences for their customers, improving customer satisfaction and loyalty. Furthermore, the proposed system can help ecommerce companies increase their revenue by providing tailored product recommendations that match customer preferences. Ultimately, the success of this project will contribute to the growth of the ecommerce industry in Cameroon by providing a solution to the challenges faced by ecommerce companies in the country.

1.3 Rationale

In recent years, the use of recommendation systems in e-commerce applications has become increasingly popular. This is because recommendation systems have been shown to provide several benefits, such as increasing sales, improving customer satisfaction, and providing a personalized shopping experience for customers. Moreover, the use of machine learning techniques in recommendation systems can help ecommerce companies analyse large volumes of customer data and make informed decisions about product recommendations.

The proposed project aims to develop an ecommerce application with a machine learning-based recommendation system to provide personalized shopping experiences for users in

Cameroon. The integration of a recommendation system into an ecommerce application can help customers discover new products that they may not have been aware of, and increase the likelihood of them making a purchase. Additionally, by providing a personalized shopping experience, ecommerce companies can improve customer satisfaction and loyalty, which can lead to increased revenue.

The use of machine learning techniques in recommendation systems can also help ecommerce companies analyse customer behaviour and preferences. This analysis can provide valuable insights into customer preferences, which can be used to make informed decisions about product recommendations. By understanding customer preferences, ecommerce companies can make more accurate product recommendations, resulting in higher sales and increased customer satisfaction.

Overall, the proposed project is relevant because it addresses the challenges faced by ecommerce companies in providing personalized shopping experiences for their customers. The integration of a machine learning-based recommendation system into an ecommerce application can help ecommerce companies in Cameroon increase sales and customer satisfaction, ultimately contributing to the growth of the ecommerce industry in the country.

1.4 Research questions

1.4.1 Main research question

The main research question of this project is: How can the development of an ecommerce application with a machine learning-based recommendation system improve the personalized shopping experience for users in Cameroon, resulting in increased revenue for the ecommerce company, increased satisfaction of clients, and a decrease in the time spent searching for products in the app?

1.4.2 Specific research questions

1. What are the best machine learning techniques for developing a recommendation system for an ecommerce application in Cameroon?
2. How can ecommerce companies in Cameroon implement a machine learning-based recommendation system?
3. What impact can the use of a machine learning-based recommendation system have on sales and customer satisfaction in e-commerce applications in Cameroon?

1.5 Objectives

1.5.1 Overall objective

The overall objective of this project was to develop an ecommerce application with a machine learning-based recommendation system which provided a personalized shopping experience for users in Cameroon, ultimately leading to increased revenue for ecommerce companies by improving customer satisfaction and reducing the time customers spend searching for products on the application.

1.5.2 Specific objectives

1. Develop an effective ecommerce recommendation system:
   - Identify and evaluate various machine learning techniques suitable for the recommendation system.
   - Develop a machine learning-based recommendation system using Python and TensorFlow to analyse customer data and offer personalized product recommendations.
2. Create a user-friendly ecommerce application:
   - Develop the ecommerce application using Python Django, Bootstrap, and jQuery.
   - Integrate the developed recommendation system seamlessly into the ecommerce application for smooth functionality.
3. Ensure optimal performance and evaluate impact:
   - Conduct rigorous testing and debugging of the ecommerce application and recommendation system to ensure reliability and performance.
   - Deploy the ecommerce application with the recommendation system on a suitable server for public access and evaluate its impact on sales and customer satisfaction in Cameroon's ecommerce applications.

1.6 Significance of study

The system has been developed using the Python programming language, the Django web framework, and the PostgreSQL database management system. This system serves as an e-commerce platform that implements a recommendation engine to suggest products to users based on their similarity to other users in the system.

1.6.1 Artefact-1 Web-Application

The developed system is a robust and scalable e-commerce website that utilizes the Python programming language, the Django web framework, and the PostgreSQL database management system. It offers various front-end features, such as user registration, authentication, product search, detailed product descriptions, product categories, subcategories, search bar, user-generated reviews and ratings. The front-end content is dynamically rendered using Django templates that query the database for product information, and only the system administrator can manage and update the database content using Django's built-in administrative interface.

The website includes a review system that enables users to share their feedback about specific products and rate them on a scale of 1 to 5. Additionally, the administrator can add new items for display to other users. The system has been optimized for mobile use with responsive design techniques and includes search engine optimization (SEO) for better performance.

Overall, the developed system offers a comprehensive e-commerce platform that integrates various technical and professional features, including robust back-end functionalities and user-friendly front-end design, to deliver an outstanding user experience.

1.6.2 Artefact-2 Recommender-System

The developed recommender system is a Product Recommendation engine that utilizes the Python programming language and various machine learning algorithms to suggest products to users based on their behaviour and preferences. The system captures user data such as browsing history, search queries, and purchase history, and analyses this data to generate personalized product recommendations.

The recommendation engine utilizes various machine learning techniques such as Collaborative Filtering, Content-Based Filtering, and Hybrid Filtering to generate accurate and relevant product recommendations. The system can also analyse user feedback such as product ratings and reviews to improve the accuracy of the recommendations.

The system has been designed to be scalable and can handle large volumes of user data efficiently. It includes various back-end functionalities such as data pre-processing, model training, and recommendation generation.

The front-end interface provides a user-friendly experience, allowing users to view recommended products and provide feedback.

1.7 Definition of terms

In the context of this project, the following terms are defined as:

E-commerce application: A web-based platform or software system designed for conducting online commercial transactions, including buying and selling of products or services.

Machine learning: A subset of artificial intelligence (AI) that focuses on enabling computer systems to learn from data and improve their performance without being explicitly programmed. In this project, machine learning algorithms are used to develop the recommendation system.

Recommendation system: A software mechanism that suggests relevant items, such as products or content, to users based on their preferences, historical data, and patterns. It utilizes machine learning techniques to generate personalized recommendations and enhance the user experience.

Personalization: The process of tailoring recommendations or content to match individual user preferences and characteristics. It involves utilizing user data, behaviour, and feedback to deliver customized experiences.

Relevance: The degree to which a recommended item aligns with a user's interests, needs, or preferences. The relevance of recommendations is determined based on various factors, including user behaviour, item characteristics, and similarity measures.

Customer satisfaction: The level of contentment or fulfilment experienced by users when interacting with the e-commerce application. It is influenced by factors such as the quality of recommendations, ease of use, user interface, and overall shopping experience.

Overall, the developed Product Recommendation system offers an effective and personalized approach to product recommendations, leveraging machine learning techniques to provide users with accurate and relevant product suggestions. The report is structured as follows: Chapter 2 provides a comprehensive literature review, examining existing solutions in Cameroon's e-commerce landscape and exploring various types of recommendation system models and matrices. Chapter 3 focuses on the materials and methods employed in the design and implementation of the e-commerce application with a machine learning-based recommendation system. It outlines the tools, technologies, and methodologies used in the development process. Chapter 4 presents the results and discussions derived from the implementation of the application. It analyses the performance, effectiveness, and user feedback of the recommendation system. Finally, Chapter 5 concludes the report by summarizing the findings, highlighting the significance of the research, and providing recommendations for future enhancements and areas of exploration within the domain of machine learning-based recommendation systems in e-commerce.

CHAPTER 2: LITERATURE REVIEW

The literature review chapter of this project focuses on exploring existing solutions in Cameroon's e-commerce landscape and different types of recommendation system models and evaluation metrics. It provides an overview of the current state of e-commerce platforms in Cameroon, highlighting the challenges encountered. Additionally, it analyses the effectiveness and applicability of various recommendation system approaches within the context of this project.

An examination of the historical trajectory of product recommendation methods used in the country is crucial. In earlier times, the prevalent approach to product recommendations in Cameroon relied predominantly on "word-of-mouth" referrals. This traditional method involved customers seeking advice and suggestions from their social networks, such as friends, family, or acquaintances, to explore new products or make informed purchasing decisions. As an inherently interpersonal method, it heavily relied on personal connections and, consequently, presented limitations in terms of the breadth and diversity of product recommendations available to consumers.

With the growth of e-commerce platforms and advancements in technology, the approach to product recommendations in Cameroon has shifted towards category-based recommendations we can see that in existing solutions in Cameroon (Durell Market, IZIWAY, Glotello, Prem Market, KiKUU etc.). In this method, products are categorized into specific groups or categories, and customers are provided with recommendations based on the category they are browsing or have shown interest in. For example, if a customer is browsing smartphones, the system would recommend other smartphones within the same category.

While category-based recommendations have improved the browsing experience to some extent, they still have limitations. One major inconvenience of this approach is the lack of personalization. Customers are not provided with tailored recommendations based on their individual preferences, browsing history, or purchase behaviour. This can result in generic recommendations that may not align with the customer's specific needs or interests. As a result, the relevance and effectiveness of these recommendations may be limited, leading to potential frustration or disengagement from customers.

To address these limitations, a more advanced approach utilizing machine learning techniques is desirable. By implementing machine learning algorithms, e-commerce platforms in Cameroon can leverage customer data to provide personalized product recommendations. These algorithms can analyse vast amounts of data, including past purchases, browsing history, product ratings, and demographics, to generate accurate and relevant recommendations for each individual customer.

Although, implementing machine learning-based recommendation systems may come with its own set of challenges. One inconvenience is the need for a substantial amount of high-quality data to train the algorithms effectively. Collecting and processing such data can be time-consuming and resource-intensive. Additionally, ensuring data privacy and security is crucial, as customer information needs to be handled carefully to maintain trust and comply with relevant regulations.

The benefits of machine learning-based recommendations, such as increased personalization, improved relevance, and enhanced customer satisfaction, outweigh the challenges. By leveraging the power of machine learning, e-commerce platforms in Cameroon can provide customers with tailored product recommendations that align with their preferences and enhance their overall shopping experience.

Recommender Systems are computer algorithms that analyse user behaviour and generate personalized recommendations for products, services, or content. These systems are widely used by various online platforms such as YouTube, Amazon, TikTok, etc to enhance user engagement and satisfaction. By leveraging machine learning and deep learning techniques, these systems can provide highly accurate and personalized recommendations to users, leading to increased user engagement and revenue for the platforms.
For example, YouTube uses recommendation algorithms to suggest videos based on a user's viewing history and preferences (Snow, 2021). Similarly, Amazon recommends products based on a user's purchase history and browsing behaviour (Linden, Brent, & Jeremy , 2003). TikTok uses a combination of user engagement metrics and content analysis to generate personalized video feeds (Zhou, 2021).
Recommender Systems have become increasingly popular in recent years due to the explosive growth of online platforms and the vast amounts of user-generated data that they

generate. These systems are beneficial for both users and platforms, as users receive personalized recommendations and platforms can increase user engagement and revenue.
Overall, Recommender Systems play an essential role in enhancing user experience and driving business growth for online platforms.

The primary function of a recommender system is to predict future user preferences based on historical data. These systems are now used in various aspects of our lives, such as purchasing items online, selecting movies, and adding friends on social media. However, predicting user preferences is a complex problem that has been the subject of intensive research for many years. One notable event that sparked increased interest in this area was the Netflix competition, which offered a 1-million-dollar prize to researchers who could substantially improve the quality of recommendations generated (Thompson, 2008).

Two primary approaches to generating recommendations are used in recommender systems. One approach involves predicting the rating a user would give to a specific item in the system, while the other approach predicts a set of items presented as an ordered list that would be recommended to the user. Recommender systems can be divided into personalized and non-personalized systems. Non-personalized systems draw conclusions based on the global behaviour of all users in the system, while personalized systems create a user profile based on their historical activity to generate recommendations.

A recommender system typically consists of a set of users (U) and a set of items (I). The interactions between users and items are recorded in a matrix R, where each element (u,i,rui) represents a user u interacting with an item i and giving it a rating rui.
Several techniques and methods have been proposed for recommender systems, including content-based filtering, collaborative filtering, knowledge-based filtering and hybrid approaches.

2.1 Types of Recommendation Systems

Prior to developing a recommender system, it is crucial to understand the advantages and disadvantages of each type of recommender system, as outlined in this section. There are generally three approaches to developing a recommender system: content-based filtering, collaborative filtering, and hybrid filtering. Each of these approaches has its own limitations and benefits, which are discussed in the latter part of this section.

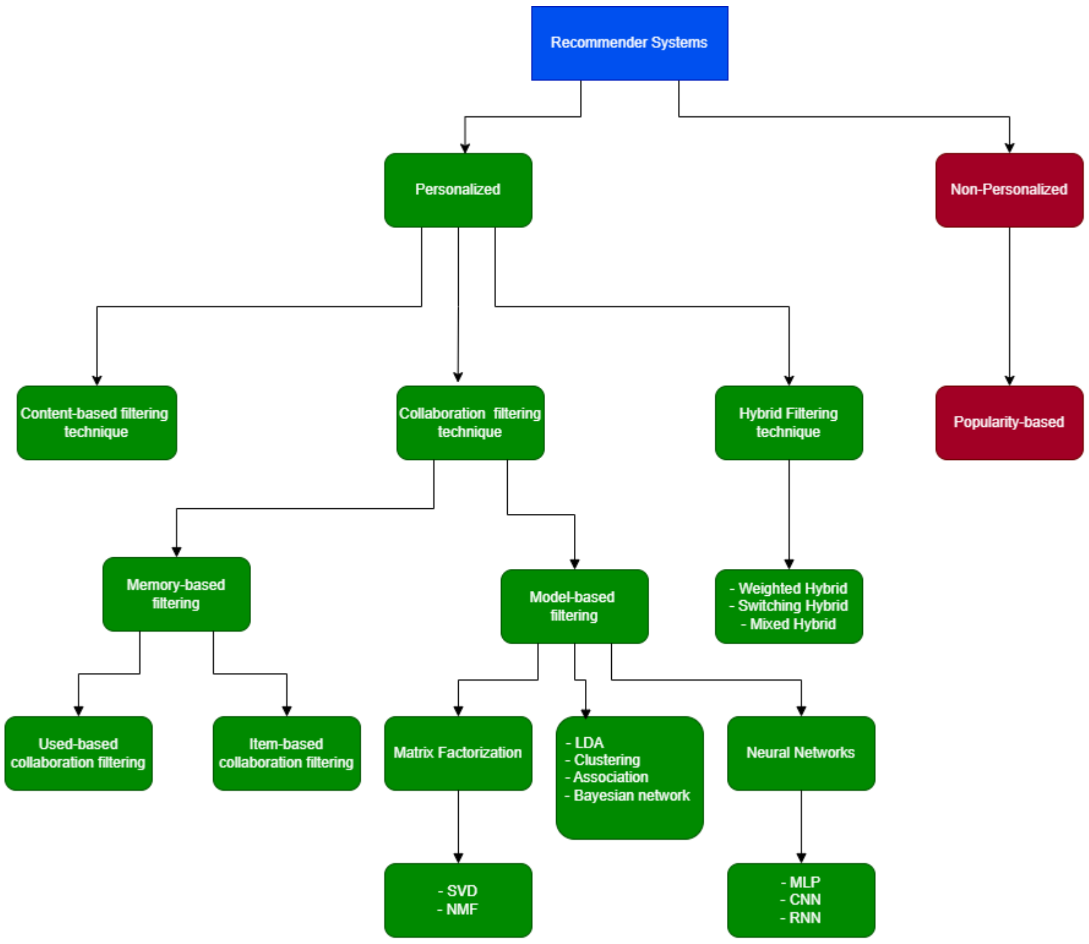


Figure 2. 1 Type of recommendation technique

2.1.1 Content based Filtering

Content-based filtering (CB) is a popular approach for developing recommender systems. CB recommends items to users based on their past preferences for similar items. It works by analysing the content of items that the user has rated or interacted with, such as text, images, or audio, and then recommending items with similar content. For example, if a user has rated several action movies highly, a CB system may recommend other action movies based on their similarity to the movies the user has already enjoyed. CB systems can be particularly useful when there is limited information about users or when users have unique preferences that cannot be easily compared to other users.

2.1.1.1 Keyword-based Vector Space Model

The Keyword-based Vector Space Model (VSM) is a commonly used approach for developing Content-based Filtering (CB) recommender systems. In this approach, items and users' preferences are represented as vectors in a high-dimensional space, where each dimension corresponds to a keyword or feature. The similarity between items and users' preferences is then calculated based on the cosine similarity between their corresponding vectors.

The VSM approach is often used with relatively simple retrieval models, such as keyword matching or basic TF-IDF weighting. TF-IDF is a term-weighting scheme commonly used in VSM that considers the frequency of a term in a document and how often the term occurs in the corpus. The resulting weight vectors are then normalized to prevent longer documents from having a better chance of retrieval.

These assumptions are well exemplified by the TF-IDF function:

$$TF(tk, dj) - IDF \ = TF(tk, dj).log\left(\frac{N}{nk}\right) \qquad 2.1$$

where **N** denotes the number of documents in the corpus, and **nk** denotes the number of documents in the collection in which the term **tk** occurs at least once.

$$TF(tk, dj) = \frac{fk, j}{\max z\ fz, j} \qquad 2.2$$

While the VSM approach is simple and versatile, it may not capture the complexity of users' preferences beyond the content of items, leading to recommendations that lack diversity. Researchers have focused on improving the accuracy and efficiency of content analysis and keyword extraction, as well as incorporating other recommendation techniques (such as collaborative filtering) to enhance the performance of VSM-based systems.

Overall, the VSM approach remains a useful and widely used technique for developing CB recommender systems, and its effectiveness can be enhanced by incorporating other recommendation techniques and contextual information (Salton, 1983).

2.1.2 Collaboration Filtering (CF)

Collaborative Filtering (CF) is a widely used approach in recommender systems that leverages user-item interactions and collective behaviour of similar users to predict users' interests. There are two primary types of CF methods: Memory-based and Model-based.

Memory-based CF methods, including User-based CF, Item-based CF, and Pearson Correlation, rely on similarity measures to generate predictions for new users or items. In contrast, Model-based CF methods, such as Matrix Factorization, Latent Dirichlet Allocation, and Deep Learning, use machine learning algorithms to learn a model from data and generate predictions.

The next topic will cover Memory-based CF methods in more detail, which are simple and easy to implement, but can suffer from scalability issues and sparsity problems. Following that, we will delve into Model-based CF methods, which are more complex and require more data, but can handle large and sparse datasets more efficiently and produce more accurate recommendations.

2.1.2.1 Memory-based Recommendation Systems

Memory-based Recommendation Systems, also known as neighbourhood-based or nearest-neighbour recommendation systems, are a type of Collaborative Filtering technique in which recommendations are made based on the similarity between users or items.

Memory-based Recommendation Systems can be implemented using either User-based or Item-based CF techniques. In User-based CF, the similarity between users is calculated based on their past interactions with items, and recommendations are generated based on the preferences of similar users. In Item-based CF, the similarity between items is calculated based on the users' past interactions with them, and recommendations are generated based on the preferences of similar items (Desrosiers, C., & Karypis, 2011).

In these systems, the past interactions of users or items are used to calculate similarity metrics, such as Jaccard, cosine similarity or Pearson, Spearman, Kendall Tau correlation coefficient, and the top-k most similar users or items are selected as neighbours. Then, the system generates recommendations for a target user or item based on the preferences of its neighbours.

2.1.2.1.1 Jaccard Similarity

Jaccard similarity is a widely used metric for measuring the similarity between two sets. It is defined as the size of their intersection divided by the size of their union. In recommender systems, Jaccard similarity is used to compute item-to-item and user-to-user similarity.

Item-to-Item Jaccard Similarity is calculated based on the interactions of users with items. It is defined as the intersection of items A and B divided by the union of items A and B. This measure is useful for finding similar items to recommend to a user based on their interactions with other items. The advantage of this measure is that it is simple and easy to implement. However, it has some limitations as it does not capture the degree of preference for each item and may not be effective for highly popular items.

Item-to-Item Jaccard Similarity:

$$\mathrm{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|} \qquad 2.3$$

Where $|A|$ and $|B|$ represent the number of users who have interacted with items A and B, respectively.

On the other hand, User-to-User Jaccard Similarity is calculated based on the interactions of users with items. It is defined as the intersection of items that two users have interacted with divided by the union of items that the two users have interacted with. This measure is useful for finding similar users based on their interactions with items and recommending items that one user has interacted with to the other. The advantage of this measure is that it is simple and easy to implement. However, it has some limitations as it does not capture the degree of preference for each item and may be biased towards users who have interacted with a larger number of items.

User-to-User Jaccard Similarity

$$J(U, V) = \frac{|I_U \cap I_V|}{|I_U \cup I_V|} \qquad 2.4$$

Where $|I_U|$ and $|I_V|$ represent the number of items that users U and V have interacted with, respectively.

In conclusion, Jaccard similarity is a useful metric for measuring item-to-item and user-to-user similarity in recommender systems. It has its advantages such as simplicity and ease of implementation. However, it also has limitations such as not capturing the degree of preference for each item.

2.1.2.1.2 Cosine Similarity

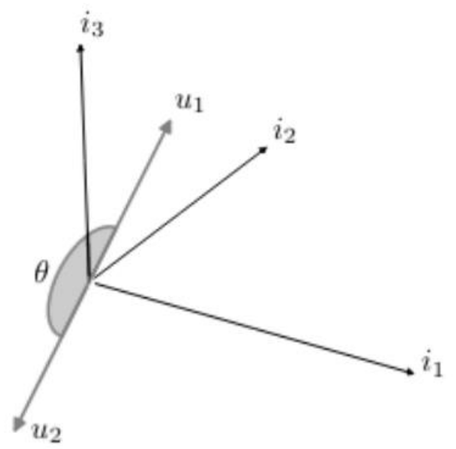
The Jaccard similarity captures our intuition about what items ought to be similar to each other, but is only defined if interactions are represented assets. We would like more nuanced similarity measures for data where feedback is associated with each interaction; for example, we might not regard two users as 'similar' if both had watched the Harry Potter movies, in the event that one of them liked the series and the other disliked it

Cosine similarity is another commonly used metric for measuring similarity between two sets. It is used in recommender systems to calculate item-to-item and user-to-user similarity based on their interactions with items.

Item-to-Item Cosine Similarity is calculated based on the interactions of users with items. It measures the cosine of the angle between two item vectors, which is defined as the dot product of the two vectors divided by the product of their magnitudes. This measure is useful for finding similar items to recommend to a user based on their interactions with other items. The advantage of this measure is that it can capture the degree of preference for each item, making it more effective for highly popular items. However, it requires normalization of item vectors, which can be computationally expensive.

On the other hand, User-to-User Cosine Similarity is calculated based on the interactions of users with items. It measures the cosine of the angle between two user vectors, which is defined as the dot product of the two vectors divided by the product of their magnitudes. This measure is useful for finding similar users based on their interactions with items and recommending items that one user has interacted with to the other. The advantage of this measure is that it can capture the degree of preference for each item and is less biased towards users who have interacted with a larger number of items.

The Cosine Similarity between two users u1 and u2 is defined in terms of the angle between the vectors u1 and u2. Recall that the angle between two vectors a and b is defined as:



$$\text{Cosine similarity } (u, v) = \frac{Ru \cdot Rv}{|Ru| \cdot |Rv|} \qquad 2.5$$

Figure 2. 2: Angle between two users

In conclusion, cosine similarity is a useful metric for measuring item-to-item and user-to-user similarity in recommender systems. It has its advantages such as capturing the degree of preference for each item and being less biased towards users who have interacted with a larger number of items. However, it also has limitations such as the requirement for normalization of item vectors, which can be computationally expensive.

2.1.2.1.3 Pearson Correlation

Pearson Correlation Coefficient is a commonly used metric for measuring similarity between two sets in recommender systems. It is used to calculate item-to-item and user-to-user similarity based on their interactions with items.

Item-to-Item Pearson Correlation Coefficient is calculated based on the interactions of users with items. It measures the linear correlation between the ratings of two items by all users who have rated both items.

On the other hand, User-to-User Pearson Correlation Coefficient is calculated based on the interactions of users with items. It measures the linear correlation between the ratings of two users for all items that they have both rated.

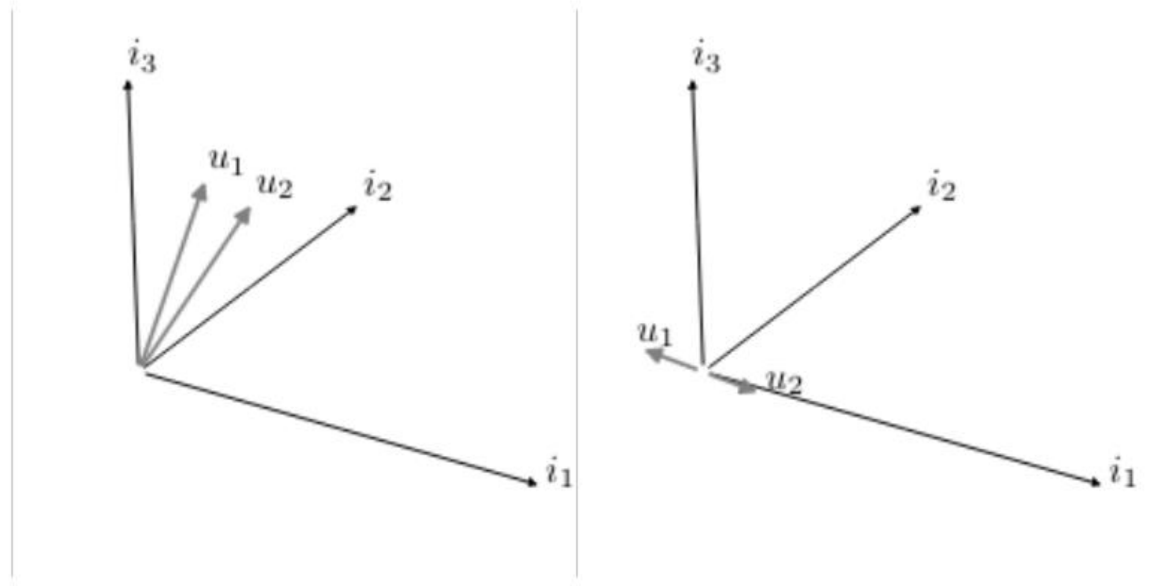


Figure 2. 3: Pearson Similarity

Two users have rating vectors that point roughly in the same direction (left); after subtracting the average from each, they point in opposite directions (right)

in Figure 5 (left) as an example. Here two users have rated the same items (i2 and i3); user u1 rated them 3 and 5 stars (respectively); user u2 rated them 5 and 3.

According to the Jaccard similarity (based only on interactions), we would consider the two users to be identical (Jaccard similarity of 1); according to the cosine similarity, we would regard them as being very similar, since the angle between the two vectors is small. However, one could argue that these users are polar opposites of each other: if we consider 5 stars to be a positive rating and 3 stars to be a negative rating, then these two users indeed have opposite opinion polarity. Our definition of the cosine similarity does not account for this interpretation, essentially because it depends on the interactions already having explicitly positive or negative polarity. To correct this, we might appropriately normalize our ratings: if we subtract the average for each user (4 stars for both u1 and u2), we end that their ratings are each 1 star above or below their personal average. The above is essentially the idea captured by the Pearson Similarity. The Pearson Correlation Coefficient is a classical measurement for assessing the relationship between two variables, i.e., whether they trend in the same direction, regardless of scale and constant differences between them. The Pearson Correlation between two vectors x and y is defined as:

$$\text{Pearson Correlation}(x, y) = \frac{\sum_{i=0}^{|x|}(\mathrm{xi} - \bar{x})(\mathrm{yi} - \bar{y})}{\sqrt{\sum_{i=1}^{|x|}(\mathrm{xi} - \bar{x})^2}\ \sqrt{\sum_{i=2}^{|y|}(\mathrm{yi} - \bar{y})^2}} \qquad 2.6$$

we might define the similarity between two users u and v (or similarly, items) only in terms of items they have both interacted with:

$$\text{Pearson Correlation}(x, y) = \frac{\sum_{i\,€\,Iv\,\cap Iu}^{|x|}\left(\mathrm{Ru,i} - \overline{Ru}\right)\left(\mathrm{Rv,i} - \overline{Rv}\right)}{\sqrt{\sum_{i\,€\,Iv\,\cap\,Iu}^{|x|}\left(\mathrm{Ru,i} - \overline{Ru}\right)^2}\ \sqrt{\sum_{i\,€\,Iv\,\cap\,Iu}^{|y|}\left(\mathrm{Rv,i} - \overline{Rv}\right)^2}} \qquad 2.7$$

However, 2.7 also has limitations. It requires users to have rated both items or users to have rated the same items, which can result in a sparse data matrix. It also assumes that the ratings follow a normal distribution, which may not be true for all datasets.

There have been several studies that have evaluated the performance of Pearson correlation coefficient for item-to-item and user-to-user similarity in recommender systems. For instance, (Su & Khoshgoftaar, 2009) conducted a study on similarity metrics for collaborative filtering and found that Pearson correlation coefficient outperformed other similarity measures such as cosine similarity and Jaccard similarity.

In conclusion, Pearson correlation coefficient is a useful metric for measuring item-to-item and user-to-user similarity in recommender systems. It has its advantages such as capturing the linear relationship between ratings and being less affected by outliers. However, it also has limitations such as the requirement for users to have rated both items or users to have rated the same items, which can result in a sparse data matrix. Researchers should consider these limitations when using Pearson correlation coefficient for recommender systems.

2.1.2.1.4 Adjusted Cosine Similarity

Adjusted Cosine Similarity is a variation of cosine similarity that is used in recommender systems to calculate item-to-item and user-to-user similarity. It considers the differences in rating scales between users, by subtracting the user's average rating from their ratings.
Item-to-Item Adjusted Cosine Similarity is calculated based on the ratings of users for items. It is defined as the cosine similarity between the ratings of two items after adjusting for user biases. On the other hand, User-to-User Adjusted Cosine Similarity is calculated based on the

ratings of items by users. It is defined as the cosine similarity between the ratings of two users after adjusting for item biases.
So, the adjusted cosine similarity is calculated by using the following formula:

$$\mathrm{sim}(i,j) = \frac{\sum_{i \in Iuv} (r\_ui - \overline{r_u})(r\_uj - \overline{r_u})}{\sqrt{\sum_{i \in Iu} (r_{ui} - \overline{r_u})^2 \sum_{i \in Iv} (r_{uj} - \overline{r_u})^2}} \qquad 2.8$$

where r_ui is the rating of user u for item i, ̄ (r_u) is the average rating of user u, and the summation is over all users who have rated both items i and j.

$$\mathrm{sim}(u,v) = \frac{\sum_{i \in Iuv} (r\_ui - \overline{r_i})(r\_vi - \overline{r_i})}{\sqrt{\sum_{i \in Iu} (r_{ui} - \overline{r_i})^2 \sum_{i \in Iv} (r_{vi} - \overline{r_i})^2}} \qquad 2.9$$

where r_ui is the rating of user u for item i, $\overline{r\_i}$ is the average rating of item i, and the summation is over all items that have been rated by both users u and v.
In conclusion, Adjusted Cosine Similarity is a useful metric for measuring item-to-item and user-to-user similarity in recommender systems. It considers the differences in rating scales between users, making it more effective in capturing user preferences.

2.1.2.1.5 Kendall Tau Correlation

Kendall Tau Correlation, also known as Kendall's rank correlation coefficient, is a metric used to measure the similarity between two sets of rankings. In recommender systems, it is often used to calculate the similarity between two users based on their rankings of items.

The Kendall Tau Correlation is calculated by comparing the number of concordant pairs and discordant pairs in the two rankings. A concordant pair is defined as a pair of items that are ranked in the same order in both rankings. A discordant pair is defined as a pair of items that are ranked in opposite orders in the two rankings. The formula for calculating Kendall Tau Correlation is:

$$\tau = \frac{(\text{number of concordant pairs} - \text{number of discordant pairs})}{\left(\frac{n(n-1)}{2}\right)} \qquad 2.10$$

where n is the number of items being ranked.

One advantage of Kendall Tau Correlation is that it is less sensitive to outliers than other correlation coefficients such as Pearson correlation. This makes it useful for datasets with extreme values or non-normal distributions. Additionally, Kendall Tau Correlation is a non-parametric measure, meaning it does not assume any specific distribution of the data.

However, one limitation of Kendall Tau Correlation is that it does not consider the magnitude of the difference in rankings between items. This means that it may not capture subtle differences in user preferences. Additionally, it requires complete rankings from users, which may not always be available in practical applications (Zhang, Cheng, & Sun, 2011).

In conclusion, Kendall Tau Correlation is a useful metric for measuring similarity between two sets of rankings in recommender systems. It has advantages such as being less sensitive to outliers and not assuming any specific distribution of the data. However, it also has limitations such as not considering the magnitude of the difference in rankings and requiring complete rankings from users.

2.1.2.1.6 Spearman Correlation

Spearman Correlation is a non-parametric measure of correlation that evaluates the relationship between two variables. In the context of recommender systems, Spearman Correlation is often used as a similarity measure to calculate the similarity between two users or two items based on their ratings.

To calculate Spearman Correlation, the ratings are first ranked and then the correlation is computed based on the ranks rather than the actual values. This makes Spearman Correlation more robust to outliers and non-linear relationships compared to Pearson Correlation.

The Spearman Correlation coefficient ranges from -1 to 1, where a value of 1 indicates a perfect positive correlation, a value of -1 indicates a perfect negative correlation, and a value of 0 indicates no correlation.

One advantage of using Spearman Correlation as a similarity measure is its robustness to outliers and non-linear relationships. It can also handle missing data and can be used with both numeric and categorical data. However, it may not be suitable for large datasets as it requires sorting and ranking of the data, which can be computationally expensive.

Several studies have compared the performance of Spearman Correlation with other similarity measures in recommender systems, such as Pearson Correlation, Cosine Similarity, and Jaccard Similarity. In a study by (Lee, 2014) , Spearman Correlation was found to outperform Pearson Correlation and Cosine Similarity in terms of recommendation accuracy for movie recommendation datasets. Another study by (Pizzato & Chung, 2019) found that Spearman Correlation performed better than Jaccard Similarity and Cosine Similarity for implicit feedback datasets.

Memory-based recommendation systems have been widely used in the industry for several years due to their simplicity and effectiveness. They assume that users who have rated items similarly in the past are likely to have similar preferences in the future.

One of the primary drawbacks of memory-based recommendation systems is that they suffer from the cold start problem, which refers to the inability to provide recommendations for new users or items with little or no rating history. Additionally, they are sensitive to sparsity and noise in the data, which can negatively affect the quality of recommendations.

To overcome these limitations, model-based recommendation systems have been developed. Model-based approaches use statistical models to learn patterns and relationships in the data and make recommendations based on these models. They are more scalable and flexible than memory-based approaches and can handle cold-start and data sparsity problems more effectively.

In conclusion, while memory-based recommendation systems have proven to be effective and simple to implement, they are limited in their ability to handle cold-start and data sparsity issues. Model-based approaches, on the other hand, offer more scalable and flexible solutions to these challenges, making them an attractive alternative for building robust recommendation systems.

2.1.2.2 Model-based Recommendation Systems

Model-based recommendation is a type of recommendation system that uses statistical models to make predictions about users' preferences for items. The models are trained on historical data, such as ratings or interactions between users and items, to capture patterns and relationships in the data. These models are then used to make recommendations for new or unseen items to users. We have two ways to train our model-based recommender system:

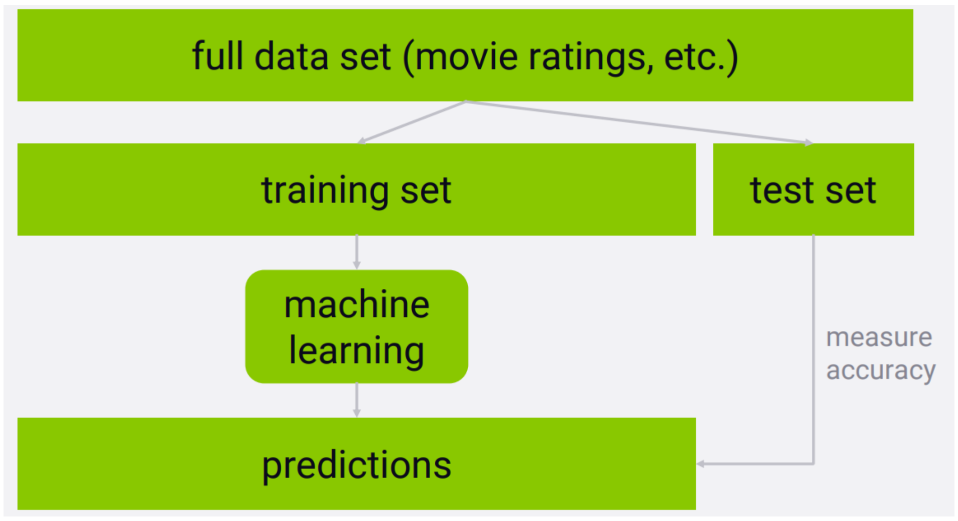


Figure 2. 4: Train/Test Process (Frank, 2018)

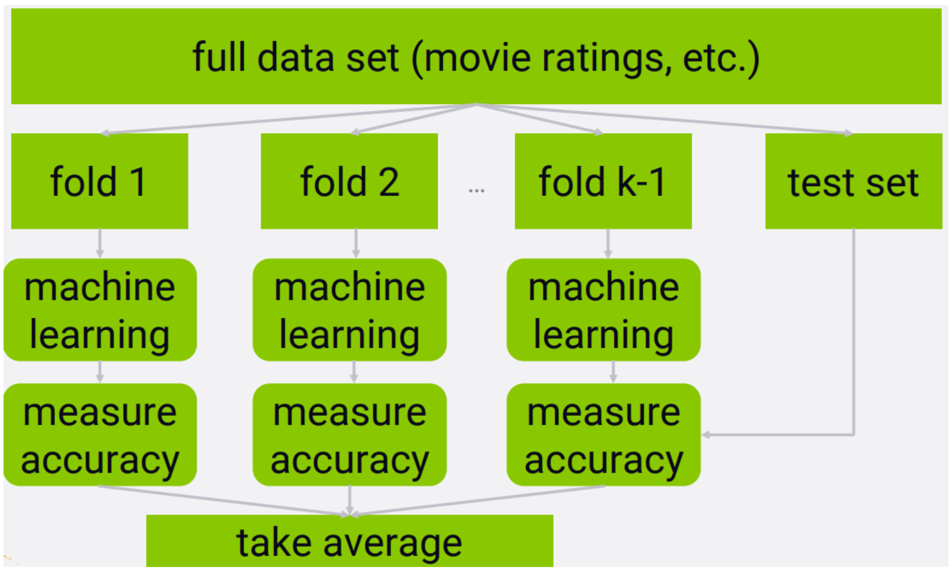


Figure 2. 5: K-folk cross-Validation (Frank, 2018)

The main advantage of model-based recommendation is that it can handle the cold-start problem, which occurs when a recommendation system has no or limited data on new users or items. Model-based approaches are also more scalable and flexible than memory-based approaches, which rely on similarity measures between items or users.

There are several types of model-based recommendation techniques, including matrix factorization, neural networks, and Bayesian networks.

However, model-based recommendation systems also have some limitations, such as the need for large amounts of data to train the models and the risk of overfitting to the training data. Nonetheless, with the development of more advanced machine learning techniques and the increasing availability of large-scale data, model-based recommendation systems are becoming increasingly popular in the industry.
Overall, model-based recommendation is a powerful and flexible approach to building recommendation systems that can handle the cold-start problem and provide accurate and personalized recommendations to users.

2.1.2.2.1 Matrix Factorization

Matrix factorization is a popular technique used in model-based recommendation systems to reduce the dimensionality of the user-item interaction matrix by decomposing it into two lower-dimensional matrices. The two matrices capture the underlying features of users and items and can be used to make recommendations to users. Matrix factorization has been shown to provide accurate and scalable recommendations in large-scale datasets (Koren, Bell, & Volinsky, 2009) .One of the most popular techniques for generating recommendations is matrix factorization, which involves transforming the matrix R into two smaller matrices (P and Q) using the formula:

$$R \approx PQ^T \qquad 2.11$$

In this decomposition is a matrix representing user features of |U| * k, and Q is a matrix representing item features of |I| * k. Then, in order to determine the user 's preference u for an item i, it is required to:

$$s(i|u) = \overline{pu}.\overline{pi} \qquad 2.12$$

Where $\overline{pu}$ is the feature vector for user u, and $\overline{pi}$ is the feature vector for item i. This technique
allows users and items to be represented by a small number of latent features, and it has become very popular due to **Simon Funk** who used it in the Netflix competition. An example of such factorization is presented below in Figure 2. 6.

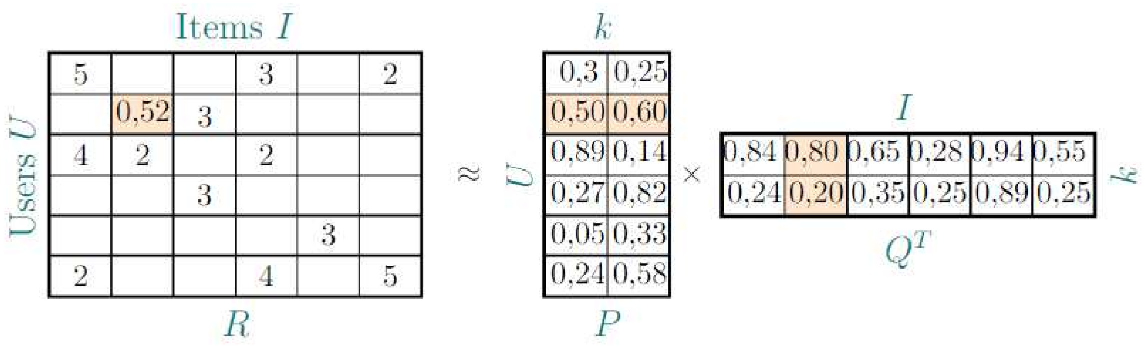


Figure 2. 6: Example of user-item matrix factorization (Usrzula & Micheal, 2023)

There are several matrix factorization techniques used in recommendation systems, including Singular Value Decomposition (SVD), SVD+, and Non-negative Matrix Factorization (NMF).

2.1.2.2.1.1 Singular Value Decomposition (SVD)

SVD is a classical matrix factorization technique that has been widely used in recommendation systems (Koren, Bell, & Volinsky, 2009). It decomposes the user-item interaction matrix into two matrices, one for users and one for items, where the columns of each matrix represent the latent factors that capture the underlying features of users and items. The latent factors are learned from the historical data, such as user-item interactions, and can be used to make predictions for new or unseen items. SVD is known for its accuracy and efficiency in providing recommendations.

The Singular Value Decomposition, a matrix M is decomposed as:

$$M = U\Sigma V \qquad 2.13$$

where U and V are left and right singular values of M (eigenvectors of MM^T and M^TM), and Σ is a diagonal matrix of eigenvectors of MM^T or M^TM. Critically, the best possible rank K approximation of M (in terms of the MSE)

To train an SVD-based recommendation system, the user-item interaction matrix is decomposed into two lower-dimensional matrices using the SVD algorithm. The latent factors are then learned by minimizing the difference between the predicted and actual ratings of users for items. This process is done iteratively until the error is minimized.

2.1.2.2.1.2 SVD+

SVD+ is an extension of SVD that incorporates explicit user and item features, such as demographic data or item content, into the matrix factorization process (Koren, Bell, & Volinsky, 2009). The explicit features are added as additional columns to the user and item matrices, allowing for a more accurate representation of users and items. SVD+ has been shown to improve the accuracy of recommendations in datasets with explicit features.

To train an SVD+-based recommendation system, the user-item interaction matrix is augmented with explicit user and item features. The augmented matrix is then decomposed into two lower-dimensional matrices using the SVD algorithm. The latent factors are learned by minimizing the difference between the predicted and actual ratings of users for items, as well as the difference between the predicted and actual values of the explicit features.

2.1.2.2.1.3 Non-negative Matrix Factorization (NMF)

NMF is a matrix factorization technique that is non-negative and typically used for non-negative data such as text and images (Lee & Seung, 2001). In the context of recommendation systems, NMF decomposes the user-item interaction matrix into two non-negative matrices, one for users and one for items, where the columns of each matrix represent the latent factors. NMF has been used in recommendation systems as an alternative to SVD and has shown promising results.

To train an NMF-based recommendation system, the user-item interaction matrix is decomposed into two non-negative matrices using the NMF algorithm. The latent factors are learned by minimizing the difference between the predicted and actual ratings of users for items.

Matrix factorization techniques are widely used in the industry for their accuracy and scalability in making recommendations to users. However, they also have limitations such as the need for large amounts of data to train the models and the risk of overfitting to the training data. Nonetheless, with the development of more advanced machine learning techniques, matrix factorization techniques are becoming increasingly popular in the industry.

2.1.2.2.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a probabilistic topic modelling technique used to discover hidden topics in a large set of documents. It assumes that each document is a mixture of topics, and each topic is a probability distribution over words. The goal of LDA is to learn the underlying topic distribution for each document and the word distribution for each topic.

In the context of recommendation systems, LDA can be used to model the latent preferences and features of users and items. It can be used to identify the hidden factors that are driving the preferences of users, which can be used to make personalized recommendations.

The LDA model can be trained using variational inference or Gibbs sampling. Variational inference involves approximating the posterior distribution of the latent variables using a simpler distribution, such as a Gaussian distribution. Gibbs sampling involves sampling the latent variables from their conditional distributions given the observed data and the current values of the other variables.

LDA has been used in a variety of recommendation systems, such as movie recommendation systems and news article recommendation systems. However, it is less commonly used than matrix factorization-based techniques. One disadvantage of LDA is that it is computationally expensive and can be slow to train on large datasets.

2.1.2.2.3 Bayesian Network

Bayesian Networks are a probabilistic graphical model that represents a set of random variables and their conditional dependencies through a directed acyclic graph (DAG). It is based on Bayes' theorem and allows for reasoning about uncertain knowledge and making predictions about new data.

The key advantage of Bayesian Networks is their ability to handle complex dependencies between variables and incorporate prior knowledge or beliefs about the relationships between variables. This makes them particularly useful in situations where there is incomplete data or a high level of uncertainty.

The joint probability distribution of the network can be factorized using the chain rule of probability, as follows:

$$\mathrm{P}(\mathrm{X}_1, \mathrm{X}_2, \dots, \mathrm{X}_\mathrm{n}) = \prod P(\mathrm{X}_\mathrm{i} \mid \mathrm{Parents}(\mathrm{X}_\mathrm{i})) \qquad 2.14$$

where $X_1, X_2, ..., X_n$ are the random variables in the network, and Parents (X_i) are the parents of node X_i in the DAG (Seiya Imoto & Hiroshi , 2006). The conditional probability distributions P (X_i | Parents (X_i)) can be estimated from the data using maximum likelihood or Bayesian methods.

To make predictions with the Bayesian Network, we can use the concept of Bayesian inference. Given some evidence about a subset of the variables in the network, we can calculate the posterior probability distribution over the remaining variables. This allows us to make predictions about the values of the unknown variables.

However, one disadvantage of Bayesian Networks is that they can be computationally expensive to train and infer, especially for large and complex networks. In addition, the quality of the predictions is highly dependent on the structure of the network and the accuracy of the conditional probability distributions.

2.1.2.2.4 Clustering-based Collaborative Filtering

Clustering-based Collaborative Filtering (CF) is a model-based approach to recommendation that groups users or items based on similarity and then makes recommendations based on the clusters. This approach is often used when the data is sparse or when there are too many users and items to make direct comparisons.

The clustering process involves grouping users or items into clusters based on their similarity. There are several clustering algorithms that can be used, such as k-means, hierarchical clustering, or density-based clustering. Once the clusters are formed, recommendations can be made to a user based on the preferences of the other users or items within the same cluster.

The key advantage of clustering-based CF is its ability to handle large datasets and sparsity, which can be a challenge for memory-based CF. It also allows for a more scalable approach to recommendation, since the clustering can be performed offline and the recommendations can be made in real-time.

The basic formula for clustering-based CF is:

$$\hat{r}\_ui = \mu + \sum_{i \in v}^{|v|} (r\_ui - \mu\_v) \qquad 2.15$$

where r̂_ui is the predicted rating of user u for item i, μ is the global mean rating, C_u is the set of users similar to user u, r_ui is the rating of user u for item i, and μ_v is the average rating of user v (Su & Khoshgoftaar, 2009).

The main disadvantage of clustering-based CF is that it requires a good clustering algorithm to be effective. The quality of the recommendations is highly dependent on the quality of the clusters, and choosing the right algorithm and parameters can be difficult. In addition, clustering-based CF does not handle dynamic changes in the data very well, since the clusters need to be updated each time new data is added.

Some popular clustering-based CF algorithms include K-means clustering, hierarchical clustering, and density-based clustering.

1.2.2.5 Neural Networks

Neural networks have become increasingly popular in the field of recommender systems due to their ability to handle non-linear relationships in data and extract meaningful features from high-dimensional data. They can be used for both content-based and collaborative filtering approaches (He, et al., 2017).

2.1.2.2.5.1 Multi-Layer Perceptron (MLP)

Multi-layer perceptron (MLP) is a type of neural network that consists of multiple layers of interconnected perceptron (neurons). MLP can be used for both content-based and collaborative filtering approaches (Cheng, et al.).

Advantages

- MLP can handle non-linear relationships in data and extract meaningful features from high-dimensional data.
- MLP can be used for both content-based and collaborative filtering approaches.

Drawbacks:

- MLP can be computationally expensive to train, especially on large datasets.
- Overfitting can be a problem if the network is too complex or the training data is limited.

2.1.2.2.5.2 Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) are commonly used in image and text data analysis but have also been applied to recommender systems. CNN can be used for content-based approaches (Pietro, Apr 16, 2022).

Advantages

- CNN can extract meaningful features from high-dimensional data.
- CNN can handle both image and text data, which makes it a versatile choice for content-based recommender systems.

Drawbacks

- Training a CNN can be computationally expensive, especially on large datasets.
- The performance of a CNN may depend on the quality of the input data.

2.1.2.2.5.3 Recurrent Neural Networks (RNN)

Recurrent neural networks (RNN) are commonly used in natural language processing and have also been applied to recommender systems. RNN can be used for both content-based and collaborative filtering approaches.

The training of RNNs involves optimizing the weights of the connections between the nodes to minimize a loss function, similar to MLPs. However, RNNs also have a feedback loop that allows them to learn from previous predictions and update their internal state accordingly (Cheng, et al.).

Advantages

- RNN can handle sequential data, such as text and user behaviour data.
- RNN can capture temporal dependencies in data, which is important for modelling user preferences over time.

Drawbacks

- Training an RNN can be computationally expensive, especially on large datasets.
- RNNs can suffer from the vanishing gradient problem, which can make it difficult to train the network effectively.

When to use: Neural networks can be used when the data is high-dimensional and contains non-linear relationships, and when more complex models are needed to capture user preferences. However, neural networks can be computationally expensive to train, especially on large datasets. Therefore, they may not be suitable for small-scale recommender systems or systems with limited computing resources.

2.1.3 Hybrid Recommendation (HR)

Hybrid recommendation systems (HRS) combine two or more recommendation techniques to provide more accurate and personalized recommendations. Over the past few years, a number of algorithms have been proposed that are designed to generate recommendations in the form of a ranking (called "TopN recommendation algorithms") Despite years of research, no universal algorithm has been proposed to generate high-quality recommendations in all cases. In addition, when we compare the generated recommendations in the context of a particular user, these algorithms do not generate identical recommendations. For this reason, researchers suggest the use of aggregation methods, the task of whose is to aggregate the rankings, generated by individual recommendation algorithms, in order to create a new, "better" recommendation (Usrzula & Micheal, 2023).

The rank aggregation problem is a computationally expensive problem. Therefore, an interesting direction of research is the use of metaheuristic algorithms that allow finding an approximate solution in an acceptable time. For example, in the publication (Usrzula & Micheal, 2023), the authors proposed a hybridization technique that combines recommendations generated by different recommendation algorithms, using an evolutionary algorithm for multi-criteria optimization. The publication (Usrzula & Micheal, 2023) suggested an Evolutionary Rank Aggregation (ERA) algorithm that used genetic programming to directly optimize the MAP measure. The authors tested the suggested solution on four datasets, and the results clearly indicate that the technique improved the quality of the generated recommendations. The authors proposed the Multi-objective Evolutionary Rank Aggregation (MERA) algorithm, which was an algorithm for multi-criteria optimization. The publication (Usrzula & Micheal, 2023) suggested using the Differential Evolution algorithm, to directly optimize the AP(Average Precision) measure for individual users in the system. This approach made it possible to find a vector, determining the preference of a given user over individual rankings. However, the main disadvantage of techniques based on metaheuristic algorithms is that, they are often difficult to implement correctly and require appropriate tuning.

Rank aggregation which involves combining multiple rankings of a set of elements to create a new, better ranking. The problem is framed in terms of a set of elements I, a set of rankings T generated by algorithms in set A, and a function Ψ that aggregates the rankings to create a new ranking τ∗. The function Ψ can be score-based or permutation-based, and can use supervised or unsupervised learning algorithms.

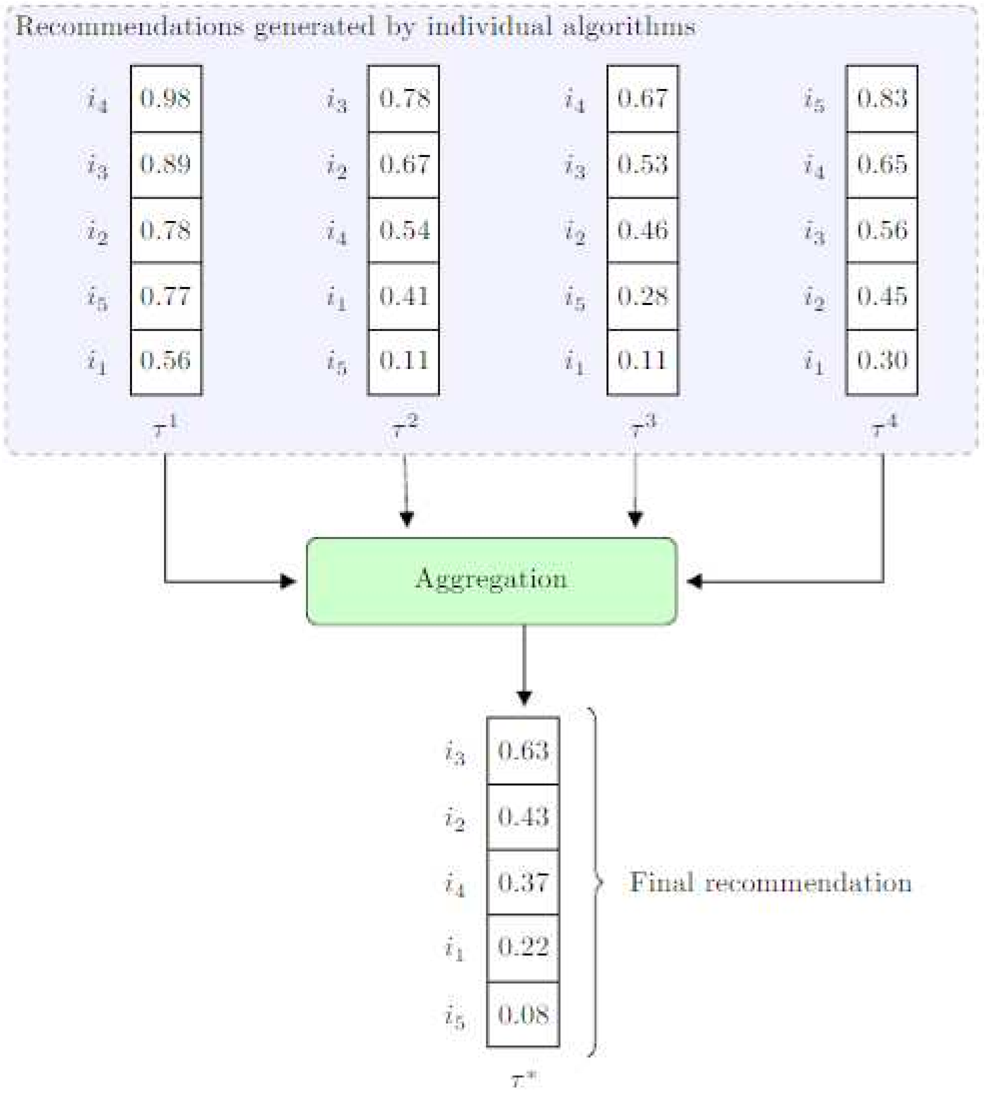


Figure 2. 7: An aggregation method

HRS can be classified into three main types: weighted hybrid, switching hybrid, and mixed hybrid (Burke & Ramezani, 2011).

2.1.3.1 Weighted Hybrid

Weighted hybrid systems combine recommendations from different methods by assigning a weight to each method, and then summing up the weighted recommendations to generate the final recommendations. The weights can be predefined or learned from the data. Weighted hybrid systems have been used successfully in many applications, such as movie and book recommendations (complexity, 2021) .

2.1.3.2 Switching Hybrid

Switching hybrid systems select the best method for each user or item based on their characteristics or the context of the recommendation. For example, a switching hybrid system might use collaborative filtering for recommending movies to a new user, and content-based filtering for recommending news articles to a regular user. Switching hybrid systems can provide better recommendations than individual methods, especially when the characteristics of the users or items vary widely (ZHANG, WalesLINA YAO , & WalesAIXIN SUN, 2017).

2.1.3.3 Mixed Hybrid

Mixed hybrid systems combine the output of different methods using techniques such as clustering or neural networks. For example, a mixed hybrid system might use collaborative filtering to generate a set of candidate items, and then use content-based filtering to rank these items based on their attributes. Mixed hybrid systems can overcome the limitations of individual methods and provide more accurate and diverse recommendations (Adomavicius & Tuzhilin, 2005).

Nonetheless, building an effective HRS requires careful consideration of the strengths and limitations of each method, as well as the appropriate combination and weighting of the methods. One of the advantages of HRS is that it can overcome the limitations of individual methods and provide more accurate and diverse recommendations. However, a significant drawback of HRS is that they can be complex and computationally expensive, which makes them less scalable than memory-based or model-based systems (Burke & Ramezani, 2011).

Overall, HRS can be a powerful tool for building recommendation systems that provide accurate and personalized recommendations. It is essential to consider the strengths and weaknesses of each method, as well as the appropriate combination and weighting of the methods, to build an effective HR.

2.1.4 Knowledge based recommendation

KBF is a type of recommendation system that makes recommendations based on a user's past interactions with a product or service, as well as their preferences and characteristics. This approach uses explicit knowledge about the user, such as their demographic information, preferences, and previous purchases, to provide personalized recommendations.

In a knowledge-based recommendation system, the system builds a model of the user's preferences and generates recommendations based on that model. This model can be created using various techniques such as collaborative filtering, content-based filtering, or hybrid approaches.

One advantage of knowledge-based recommendation systems is that they can provide highly personalized recommendations even when there is limited data available. However, they are limited by the amount of explicit knowledge about the user that is available and may not be as effective as other types of recommendation systems in cases where there is little data available about the user.

2.2 Evaluation Metrics for Recommendation Systems

In order to evaluate the effectiveness of a recommendation system, it is important to use appropriate metrics to measure its performance. Evaluation metrics for recommendation systems are used to assess the accuracy, diversity, novelty, coverage, and other aspects of the recommendations provided to the users. By using these metrics, we can determine whether the recommendations are relevant and useful to the users, and make improvements to the recommendation system if necessary. This section will introduce some common evaluation metrics used in recommendation systems and explain how they are calculated and interpreted.

2.1 Accuracy Metrics:

Accuracy metrics are used to evaluate the accuracy of the recommendation system by comparing the predicted ratings with the actual ratings provided by the users. Two common accuracy metrics are Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

2.2.1.1 Mean Absolute Error (MAE):

Mean Absolute Error (MAE) is a measure of the average absolute difference between the predicted and actual ratings. It is calculated as follows (Frank, 2018):

$$MAE = \left(\frac{1}{N}\right)\sum_{i=1}^{N}(\ predicted\ rating - actual\ rating) \qquad 2.16$$

where N is the total number of ratings.

The smaller the MAE value, the more accurate the recommendation system is.

2.2.1.2 Root Mean Squared Error (RMSE):

Root Mean Squared Error (RMSE) is a measure of the difference between the predicted and actual ratings, but it is more sensitive to large errors than MAE. It is calculated as follows (Frank, 2018):

$$RMSE = \sqrt{\left(\frac{1}{N}\right)\sum_{i=1}^{N}(\ predicted\ rating - actual\ rating)^2} \qquad 2.17$$

where N is the total number of ratings.

The smaller the RMSE value, the more accurate the recommendation system is.

2.2.2 Diversity Metrics:

Diversity metrics are used to evaluate the diversity of the recommended items. A recommendation system with high diversity can provide more varied recommendations to the users, which can lead to better user satisfaction and engagement. Several diversity metrics are commonly used, including Novelty, Coverage, Hit rate, Average reciprocal hit rate (ARHR), Cumulative hit rate (cHR), Rating hit rate (rHR), Diversity, Churn, a/b tests, and Serendipity.

2.2.2.1 Novelty:

Novelty measures the degree to which the recommended items are new or unexpected to the users. It is calculated as follows (Frank, 2018):

$$Novelty = 1 - \left(\frac{1}{N}\right)\sum_{i=1}^{N}\log_2 rank(i) \qquad 2.18$$

where N is the set of recommended items and rank(i) is the rank of item i in the list of recommended items.

The higher the novelty value, the more novel the recommended items are.

2.2.2.2 Coverage:

Coverage measures the proportion of items in the entire item set that are recommended to at least one user. It is calculated as follows:

$$Coverage = \frac{|recommended\ items|}{|total\ items|} \qquad 2.19$$

where |recommended items| is the number of items recommended to at least one user, and |total items| is the total number of items in the item set.

The higher the coverage value, the more diverse the recommended items are.

2.2.2.3 Hit rate:

Hit rate measures the proportion of recommended items that are selected by the users. It is calculated as follows:

$$Hit\ rate = \frac{|recommended\ item\ selected\ by\ users|}{|\ recommended\ items|} \qquad 2.20$$

The higher the hit rate value, the more relevant the recommended items are to the users.

2.2.2.4 Average reciprocal hit rate (ARHR):

Average reciprocal hit rate (ARHR) is a modified version of hit rate that considers the rank of the selected items. It is calculated as follows (Frank, 2018):

$$ARHR = \left(\frac{1}{|U|}\right) \qquad 2.21$$

where U is the set of users, and rank(i) is the rank of the selected item i in the list of recommended items.

The higher the ARHR value, the more relevant and diverse the recommended items are.

2.2.2.5 Cumulative hit rate (cHR):

Cumulative hit rate (cHR) measures the proportion of recommended items that are selected by the users within a certain number of top items. It is calculated as follows:

$$cHR = \frac{|\ recommended\ items\ selected\ by\ users\ within\ topk|}{k} \qquad 2.22$$

where k is the number of top recommended items.

The higher the cHR value, the more relevant the top recommended items are to the users.

2.2.2.6 Rating hit rate (rHR):

Rating hit rate (rHR) measures the proportion of recommended items that are selected by the users with a high rating. It is calculated as follows:

$$rHR = \frac{|\ recommended\ items\ selected\ by\ users\ with\ a\ high\ rating|}{recommeded\ items} \qquad 2.23$$

Where a high rating is defined as a rating above a certain threshold.

The higher the rHR value, the more relevant the recommended items are to the users.

2.2.2.7 Diversity:

Diversity measures the degree to which the recommended items are different from each other. It is calculated as follows (Frank, 2018):

$$Diversity = \frac{1}{|N|^2} \sum_{i=0}^{N} \sum_{j=0}^{N} sim(i,j) \qquad 2.24$$

where N is the set of recommended items, and sim(i,j) is the similarity between item i and item j.

The higher the diversity value, the more varied the recommended items are.

2.2.2.8 Churn:

Churn measures the degree to which the recommended items change over time. A recommendation system with high churn provides different recommendations to the users over time, which can lead to better user satisfaction and engagement. It is calculated as follows:

$$Churn = 1 - \frac{1}{|U|} \sum sim(t,i) * sim(t+1,j) \qquad 2.25$$

where U is the set of users, and sim(t,i) and sim(t+1,j) are the similarity between item i at time t and item j at time t+1.

The higher the churn value, the more varied the recommended items are over time.

2.2.2.9 A/B tests:

A/B tests are a statistical method used to evaluate the performance of different recommendation algorithms. In an A/B test, users are randomly assigned to either the control group or the test group. The control group receives recommendations from the current algorithm, while the test group receives recommendations from a new algorithm. The performance of the two algorithms can then be compared based on metrics such as click-through rate, conversion rate, and revenue.

2.2.2.10 Serendipity:

Serendipity measures the degree to which the recommended items are surprising or unexpected to the users. It is calculated as follows:

$$Diversity = \frac{1}{|U|}\sum(1 - sim(i, p(i))) \qquad 2.26$$

where U is the set of users, p(i) is the probability that item i would have been recommended by chance, and sim(i,p(i)) is the similarity between item i and the item that would have been recommended by chance.

The higher the serendipity value, the more surprising or unexpected the recommended items are.

2.2.2.11 Accuracy:

In the context of a recommendation system, accuracy refers to the measure of how effectively the system identifies the correct recommendations. It is calculated by determining the ratio of true positives and true negatives to the total number of recommendations.

$$accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TN} + \mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \qquad 2.27$$

2.2.2.12 Precision:

Precision in the context of a recommendation system represents the exactness of the system's predictions. A higher precision indicates that there are fewer false positives, meaning that the recommendations provided are more likely to be relevant and aligned with the user's preferences. On the other hand, a lower precision suggests a higher rate of false negatives, where some relevant recommendations might be missed.

$$precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad 2.28$$

2.2.2.13 Recall:

Recall, also known as sensitivity, assesses the completeness of the recommendations provided by the system. A higher recall indicates that the system is capturing a larger proportion of relevant recommendations, resulting in fewer false negatives.

$$recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad 2.29$$

Where:

- TP (True positive): Recommendations that are correctly identified as relevant by the system.
- FP (False positive): Recommendations that are incorrectly identified as relevant by the system.

- TN (True negative): Non-relevant recommendations that are correctly identified as such by the system.
- FN (False negative): Relevant recommendations that are incorrectly identified as non-relevant by the system.

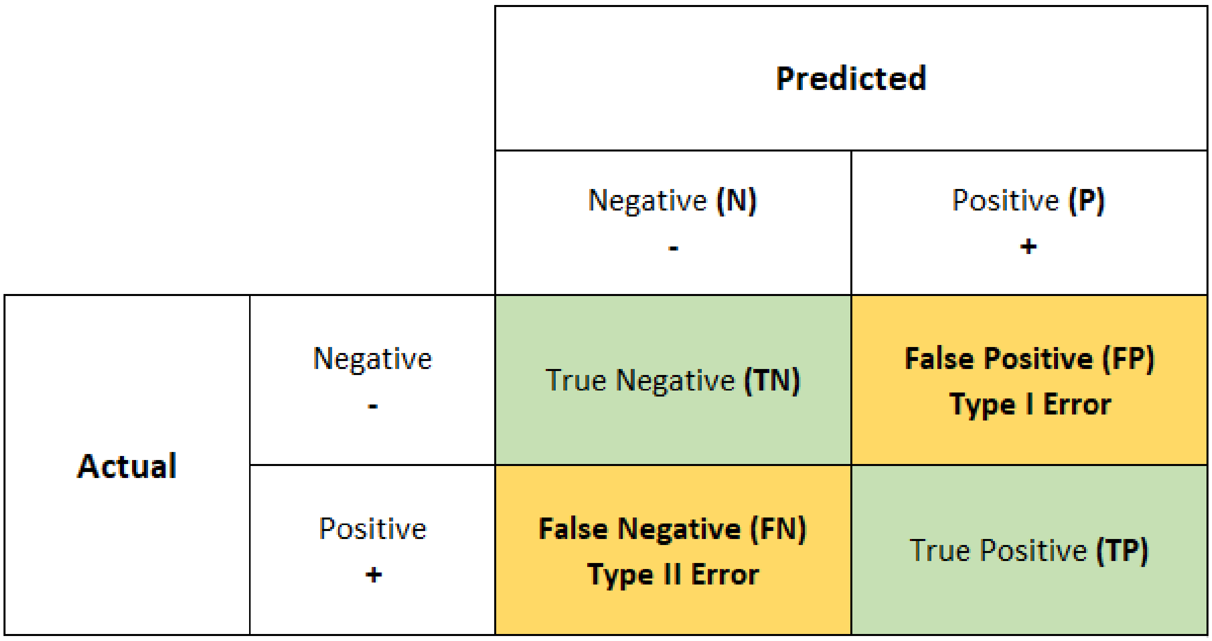
2.2.2.14 F1-score:

The F1-score is a combined measure of precision and recall, providing an overall evaluation of the recommendation system's accuracy. It is calculated as the harmonic mean of precision and recall. A higher F1-score indicates that the system is more accurate in delivering relevant recommendations.

$$\text{F1} - \text{score} = \frac{2 * (\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \qquad 2.30$$

2.2.2.15 Confusion Matrix:

A confusion matrix is a table that helps assess the performance of a classification algorithm. It provides a summary and visualization of how well the algorithm has classified different instances. The matrix consists of four outcomes: True Negative (TN), True Positive (TP), False Negative (FN), and False Positive (FP). These outcomes serve as the basis for calculating key performance metrics such as accuracy, precision, recall, and F1 score for classification models.



		Predicted	
		Negative **(N)** **-**	Positive **(P)** **+**
Actual	Negative -	True Negative **(TN)**	**False Positive (FP)** **Type I Error**
	Positive **+**	**False Negative (FN)** **Type II Error**	True Positive **(TP)**

Figure 2. 8: Example of confusion matrix

2.6 Conclusion

In this study, we have discussed various types of recommendation systems and evaluation metrics. Collaborative filtering, content-based filtering, and hybrid filtering are the most commonly used types of recommendation systems. Collaborative filtering is useful when there is a large amount of user-item interaction data available, while content-based filtering is more suitable for recommending items with specific features. Hybrid filtering combines the strengths of collaborative filtering and content-based filtering to provide more accurate and diverse recommendations.

Accuracy metrics such as MAE and RMSE are commonly used to evaluate the accuracy of recommendation systems. Diversity metrics such as Novelty, Coverage, and Hit rate are used to evaluate the diversity of recommended items, which is important for user satisfaction and engagement.

It is important to note that there is no one-size-fits-all recommendation system or evaluation metric. The choice of recommendation system and evaluation metric depends on various factors such as the type of data available, the nature of the items being recommended, and the specific requirements of the application.

CHAPTER 3: MATERIALS AND METHODS

Materials and methods utilized in the development of the eCommerce app with a recommendation system provides a detailed overview of the resources, tools, and datasets employed, as well as the approaches and techniques used to gather requirements and implement the system.

3.1 Materials

The materials used in the development of the eCommerce app with a recommendation system encompassed hardware, software, and datasets. These materials were essential for the successful implementation and functioning of the app.

3.1.1 Hardware

The hardware utilized for the development and testing of the app included a Thinkpad laptop with an Intel Core i5 processor, 8GB of RAM, and a 500GB hard disk drive. This hardware configuration provided sufficient computing power and storage capacity to support the app's development environment and ensure smooth operation during testing and deployment.

3.1.2 Software

The software components played a crucial role in the development and deployment of the app. The primary Integrated Development Environment (IDE) used was **PyCharm Professional**, which provided a comprehensive set of tools and features for Python development. Additionally, Spyder was used for specific tasks related to the development and training of the hybrid model.

The programming languages utilized in the app's development were ***Python*** and ***JavaScript***. Python served as the primary language for backend development and implementing the recommendation system, while JavaScript was used for frontend development to enhance interactivity and user experience.

The app was built using the ***Django*** framework, which provided a robust and scalable foundation for developing web applications. Django facilitated rapid development, streamlined database management, and offered built-in security features.

Django REST Framework was utilized as a key software tool. Django REST Framework is a powerful and flexible framework that enables the creation of robust and scalable APIs (Application Programming Interfaces) for web applications. It provides a comprehensive set of tools and functionalities for building RESTful APIs, allowing seamless communication between the front-end and back-end components of the application.

By incorporating Django REST Framework into the development process, it was possible to design and implement RESTful endpoints that facilitate data exchange and interaction with the eCommerce app. This framework offered built-in support for handling common tasks such as serialization, authentication, and request/response management, simplifying the development process and promoting code reusability.

To enhance the user interface and overall design aesthetics, libraries such as jQuery and **Bootstrap** were employed. ***jQuery*** simplified DOM manipulation and event handling, while Bootstrap provided a responsive and visually appealing framework for building user interfaces. ***HTML*** and ***CSS*** were utilized to structure and style the app's webpages.

UML (Unified Modelling Language) was used to create a diagram representing the structure and relationships of my system. To create the UML diagram, we employed the **Draw.io** tool, which provided a user-friendly interface for designing and visualizing the components of my system.

Using UML, we were able to present a clear and concise representation of the system's architecture, including its various components, classes, and their relationships. The diagram helped to illustrate the organization and interactions among different modules, allowing for a better understanding of the system's overall design.

To develop the model for recommendation system of this project, several libraries were used. These libraries provided essential functionality for data manipulation, visualization, machine learning algorithms, and deep learning architectures. The following libraries were utilized:

Pandas: Pandas is a powerful library for data manipulation and analysis. It provides data structures and functions to efficiently handle structured data, such as CSV files, in a tabular format.
NumPy: NumPy is a fundamental library for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

Matplotlib and Seaborn: Matplotlib and Seaborn are plotting libraries used for data visualization. They offer a wide range of options to create various types of plots, including line plots, scatter plots, histograms, and more.

Scikit-learn: Scikit-learn is a popular machine learning library that provides a diverse set of algorithms and tools for tasks such as classification, regression, clustering, and model evaluation. It includes implementations of various machine learning algorithms, data pre-processing techniques, and evaluation metrics.

TensorFlow and Keras: TensorFlow is an open-source library for numerical computation and large-scale machine learning. Keras is a high-level neural networks API that runs on top of TensorFlow. Together, they offer a flexible framework for building and training deep learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more.

MinMaxScaler: MinMaxScaler is a utility class in scikit-learn that provides a simple method to scale features to a specified range. It was used for feature scaling in the data pre-processing step.

In summary, these libraries played a crucial role in data pre-processing, exploratory data analysis, model development, and model evaluation, enabling the implementation of machine learning and deep learning models for accurate and effective recommendations in this project.

PostgreSQL was employed as the database management system for the eCommerce app with a recommendation system. PostgreSQL is a powerful and robust open-source relational database that offers advanced features and ensures data integrity and scalability. It provided a reliable and efficient storage solution for handling various aspects of the app, including user profiles, product catalogue management, order information, and recommendation data. PostgreSQL's support for complex queries and its ability to handle large datasets made it an ideal choice for managing the app's data and ensuring smooth and reliable operations.

For Payment on the eCommerce app we used **Flutterwave** is a payment platform that allows us to accept various payment methods (mobile money, credit card, etc).

For the software infrastructure, the eCommerce app was hosted on the **Render cloud platform**, which provided a reliable and scalable hosting environment. Render is a cloud-based platform that offers managed hosting services, allowing developers to deploy and manage their applications with ease.

GitHub played a significant role in the development process of the eCommerce app. It served as a version control system. By utilizing GitHub, I created branches, tracked changes, and merged code seamlessly. This version control system provided a reliable and organized environment for managing code revisions, making it easier to identify and resolve issues, track progress, and maintain a stable and up-to-date codebase.

In addition to utilizing GitHub as a version control system, we leveraged **AWS S3** for storing product images in our Django-based eCommerce app. AWS S3 provided a scalable and reliable solution for managing our image storage needs. Storing images in the Django application's file system becomes inefficient as the image count grows. With AWS S3, we offloaded storage to a dedicated and highly available cloud service. AWS S3 offers virtually unlimited storage capacity, high durability, and availability. Integration with Django was straightforward using the Django Storages library, allowing easy upload, retrieval, and serving of images. AWS S3 also provided robust access control and permissions management, ensuring image security. Leveraging AWS **CloudFront**, we distributed images globally, enhancing user experience with reduced load times and latency. Overall, AWS S3 in our Django app provided a scalable, reliable, and secure infrastructure for managing and delivering product images, allowing us to focus on other critical development tasks.

Additionally, the app utilized a **Redis server** as a key component of its architecture. Redis is an in-memory data structure store that enables fast data retrieval and caching. It was employed in the eCommerce app to handle background tasks, such as session management, caching of frequently accessed data, prediction computation and improving overall system performance.

3.1.3 Datasets

A dataset is a collection of data that is typically related to a specific subject. It comprises rows of data and can encompass various types such as categorical, numerical, text, or image data.

Datasets play a crucial role in training machine learning models. They are typically divided into training and testing sets. The training set is utilized to train the model, while the testing set is employed to assess the model's performance and prevent overfitting. By providing a larger and more diverse dataset, the model's accuracy and generalization capability can be improved.

For our project, we utilized a dataset sourced from GroupLens (https://grouplens.org/datasets/movielens/latest/). GroupLens is a human–computer interaction research lab in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities specializing in recommender systems and online communities.

3.2 Methods

3.2.1 Requirement Gathering

The requirement gathering phase of the project involved gathering the necessary information and insights to guide the development of the eCommerce app with a recommendation system. Various techniques and approaches were employed to ensure a comprehensive understanding of the project requirements.

Stakeholder identification was the initial step in the requirement gathering process. The stakeholders, including potential users, managers, and domain experts, were identified, and their inputs were sought to gain a holistic perspective of the project. Through meetings and discussions, their expectations, preferences, and specific requirements were gathered to shape the app's functionalities and features.

To further gather requirements, observation, research, and studying existing eCommerce companies (Amazon, Alibaba, Udemy) were chosen as fact-finding techniques. By observing similar systems and studying the practices of established eCommerce platforms, valuable insights were obtained regarding industry standards, user expectations, and common features implemented in successful eCommerce apps. This helped identify the essential elements that needed to be incorporated into the developed app.

Moreover, thorough research was conducted to explore the latest trends and advancements in the Ecommerce domain, including emerging technologies and best practices. This research provided a broader context and helped identify potential areas for innovation and improvement in the app's design and functionality.

The requirement gathering process also involved analysing existing eCommerce companies to gain a deeper understanding of their approaches to user experience, product recommendation, and other key aspects. This analysis served as a benchmark for identifying the desired functionalities and features for the developed app.

It's important to note that interviews and questionnaires were not chosen as the primary fact-finding techniques in this project. Instead, observation, research, and studying existing eCommerce companies proved to be more effective in gathering the necessary requirements. By observing similar systems and studying competitors, valuable insights were obtained regarding user expectations, industry standards, and successful implementation strategies.

Overall, the requirement gathering phase was crucial in determining the key functionalities and features of the eCommerce app with a recommendation system. The inputs gathered from stakeholders, observation, research, and studying existing eCommerce companies laid the foundation for the subsequent phases of the project, guiding the design and development process.

3.2.2 User Requirements

The user requirements phase focused on gathering and documenting the specific needs and expectations of the users for the eCommerce app with a recommendation system. This phase aimed to understand the functionality, usability, and overall user experience that the app should deliver.

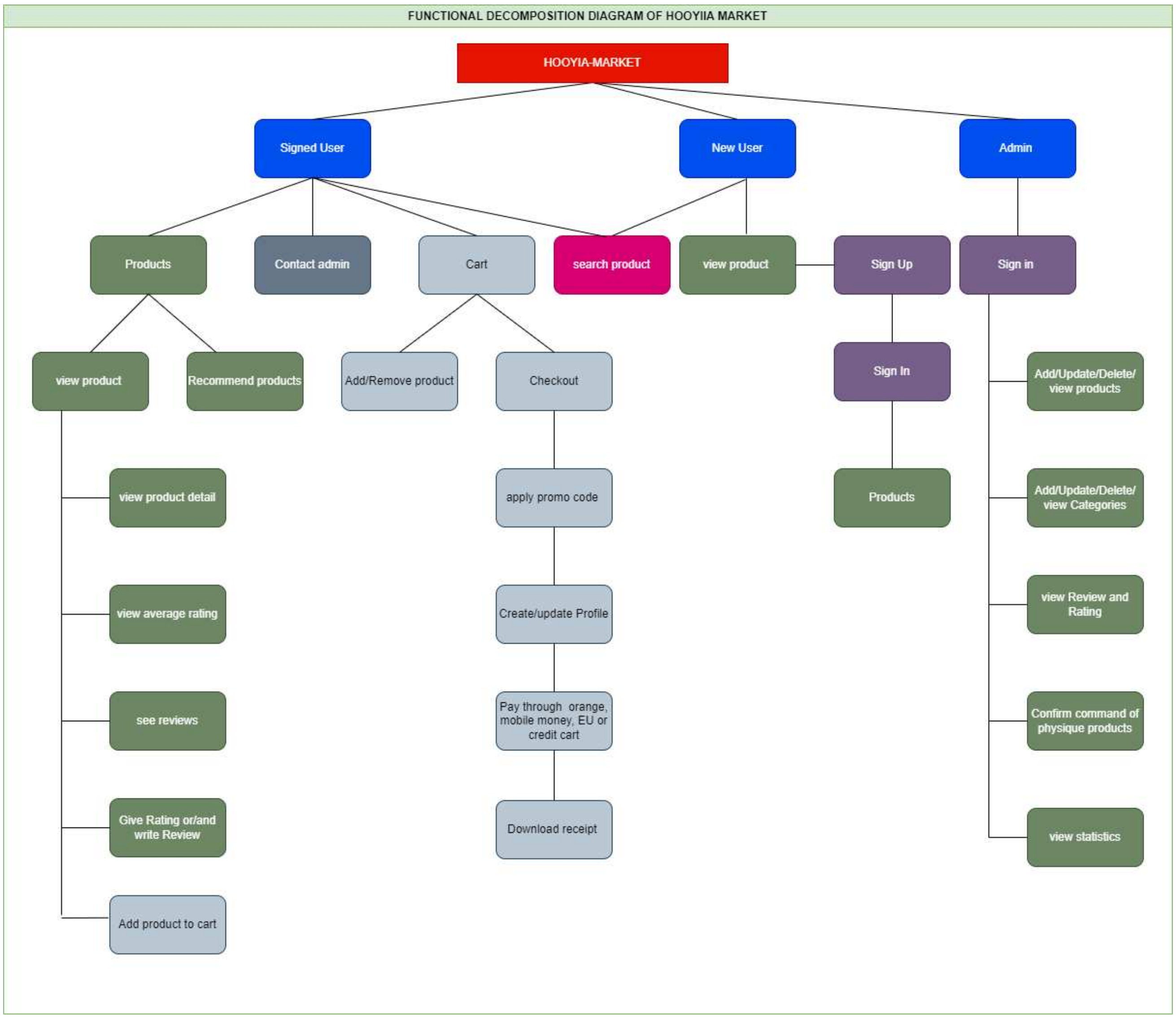


Figure 3. 1 Functional Decomposition Diagram

During this phase, a thorough analysis of the stakeholders' inputs and requirements was conducted to identify the key user requirements. These requirements were defined based on factors such as user preferences, shopping behaviours, and the desired level of personalization.

The user requirements encompassed various aspects of the app's functionality. One of the primary requirements was user registration and authentication, ensuring that users could create accounts, securely log in, and manage their personal information. This feature aimed to enhance the user experience by providing a personalized and secure environment.

Another crucial aspect of the user requirements was product catalogue management. The app needed to offer an extensive and well-organized catalogue, allowing users to browse, search, and filter products based on their preferences. This requirement aimed to simplify the product discovery process and provide users with a seamless shopping experience.

User profile management was also identified as an essential requirement. Users should have the ability to update their profiles, manage their preferences, and view their order history. This feature aimed to empower users by providing them with control over their personal information and allowing them to tailor their shopping experience.

Furthermore, search and filtering capabilities were deemed crucial for enhancing user convenience. Users should be able to search for products using keywords and apply various filters such as price range, brand, and category. This requirement aimed to streamline the product search process and assist users in finding the products that best match their preferences.

The recommendation generation and display were key user requirements for the app. The recommendation system should analyse user behaviours, purchase history, and preferences to generate personalized product recommendations. These recommendations should then be displayed prominently to users, providing them with relevant and enticing suggestions to enhance their shopping experience.

Other user requirements included a seamless cart and checkout process, efficient order management, and timely notifications and updates regarding the order status. These requirements aimed to ensure smooth transactions and provide users with a reliable and transparent purchasing process.

By identifying and documenting these user requirements, the development team gained valuable insights into the specific functionalities and features that needed to be implemented. These requirements formed the foundation for the subsequent design and development phases, guiding the creation of an eCommerce app that would meet the users' expectations and deliver a satisfying shopping experience.

In the next section, 3.2.3 System Requirements, we will shift our focus to identifying the functional and non-functional requirements of the eCommerce app and the recommendation system, drawing upon the insights gained from the observation, research, and studying of successful eCommerce companies.

3.2.3 System Requirements

In order to build a robust and effective eCommerce app with a recommendation system, it is essential to identify and define the system requirements. System requirements encompass both functional and non-functional aspects, ensuring that the app and recommendation system meet the desired objectives and provide a seamless user experience.

3.2.3.1 Functional Requirements

The functional requirements of the system define the specific functionalities and features that the eCommerce app and recommendation system should possess. These requirements directly contribute to the app's core functionality and include:

- **User Registration and Authentication**

Users are able to register and create an account.
Authentication mechanisms, such as email/password or social media login, was implemented to ensure secure access to user accounts.

- **Product Catalogue Management**

The app provides an intuitive interface for managing product catalogues.
Admin users should be able to add, update, and remove products, including their attributes such as name, description, price, and category.

- **User Profile Management**

Users have the ability to view and update their profile information.
User profiles can include personal details, preferences, and saved information for a personalized experience.

- **Search and Filtering**

The app offers a search functionality for users to find products based on keywords, categories, or specific attributes.
Filtering options, such as price range, brand, or ratings, is provided to narrow down search results.

- **Recommendation Generation**

The recommendation system generates personalized product recommendations for each user.
Recommendations can be based on user browsing history, purchase behaviours, preferences, or a combination of these factors.

- **Recommendation Display**

The app displayed recommended products in a visually appealing manner within the user interface. Recommendations can be showcased on the homepage, product pages, or through dedicated recommendation sections.

- **Cart and Checkout**

Users is able to add products to a cart, review cart items, and proceed to checkout for purchasing. Integration with payment gateways is implemented to facilitate secure and seamless transactions.

- **Order Management**

Admin users have access to an order management system to process, track, and manage customer orders. Users received order confirmation and tracking details via email and SMS.

3.2.3.2 Non-Functional Requirements

Non-functional requirements define the qualities and characteristics that the eCommerce app and recommendation system should possess. These requirements focus on aspects such as performance, security, scalability, and user experience. Some examples include:

- **Performance**

The app is responsive and provide fast loading times, even with a large number of products and concurrent users. Recommendation generation and display should be efficient to ensure real-time responsiveness.

- **Security**

User data, including personal and payment information, are securely stored and transmitted using encryption techniques. Access controls and authentication mechanisms are implemented to protect user accounts and prevent unauthorized access.

- **Scalability**

The app and recommendation system are designed to handle increasing user traffic and product catalogues. Scalability measures, such as load balancing and cloud infrastructure, is considered to ensure smooth performance under high loads.

- **User Experience**

The user interface is intuitive, visually appealing, and easy to navigate.
The recommendation system provides relevant and accurate recommendations, enhancing the overall user experience.

- **Compatibility and Cross-Browser Support**

The app is compatible with different web browsers and operating systems, ensuring a consistent user experience across various platforms and devices.

Compatibility testing is conducted to verify the app's functionality and appearance on popular browsers such as Chrome, Firefox, Safari, and Internet Explorer. Responsive design techniques are implemented to adapt the app's layout and content to different screen sizes, including desktops, laptops, tablets, and mobile devices.

- **Error Handling and Logging**

The system includes robust error handling mechanisms to gracefully handle exceptions and errors that may occur during user interactions or system operations.
Logging functionality is implemented to capture and record system events, errors, and user activities for troubleshooting and analysis purposes.

- **Accessibility**

The app is designed and developed following accessibility guidelines to ensure that users with disabilities can access and use the system effectively. Considerations such as providing alternative text for images, keyboard navigation support, and proper color contrast should be implemented.

- **Performance Monitoring and Optimization**

Regular performance monitoring and optimization is carried out to identify bottlenecks, improve system response times, and enhance overall system performance. Techniques such as caching, database optimization, and code profiling is employed to achieve optimal performance. By defining both functional and non-functional requirements, the eCommerce app and recommendation system is developed to meet user expectations, provide a seamless shopping experience, and leverage the capabilities of Python, Django, HTML, CSS, jQuery, Bootstrap, TensorFlow, sklearn, and scikit-surprise library.

3.3 Design

The design phase of the eCommerce app with a recommendation system encompassed various aspects, including process design, use case diagram, class diagram, sequence diagram, state transition diagram, activity diagram, and data dictionary diagram. These design components were crucial in defining the structure, behavior, and flow of the app.

3.3.1 Process Design

Process design involved defining the workflows and interactions between different components of the app. It outlined how users and administrators would interact with the system, including the steps involved in various processes such as user registration, product catalog management, recommendation generation, and order management. This process design ensured that the app operated smoothly and efficiently, meeting user requirements at each stage.

3.3.2 Use Case Diagram

The use case diagram provided a high-level view of the functionalities and interactions between different actors (users, administrators, recommender) and the system. It illustrated the various actions and scenarios that could occur within the app, helping to identify the key features and user interactions. The use case diagram served as a valuable tool for understanding the app's overall functionality and defining the scope of the system.

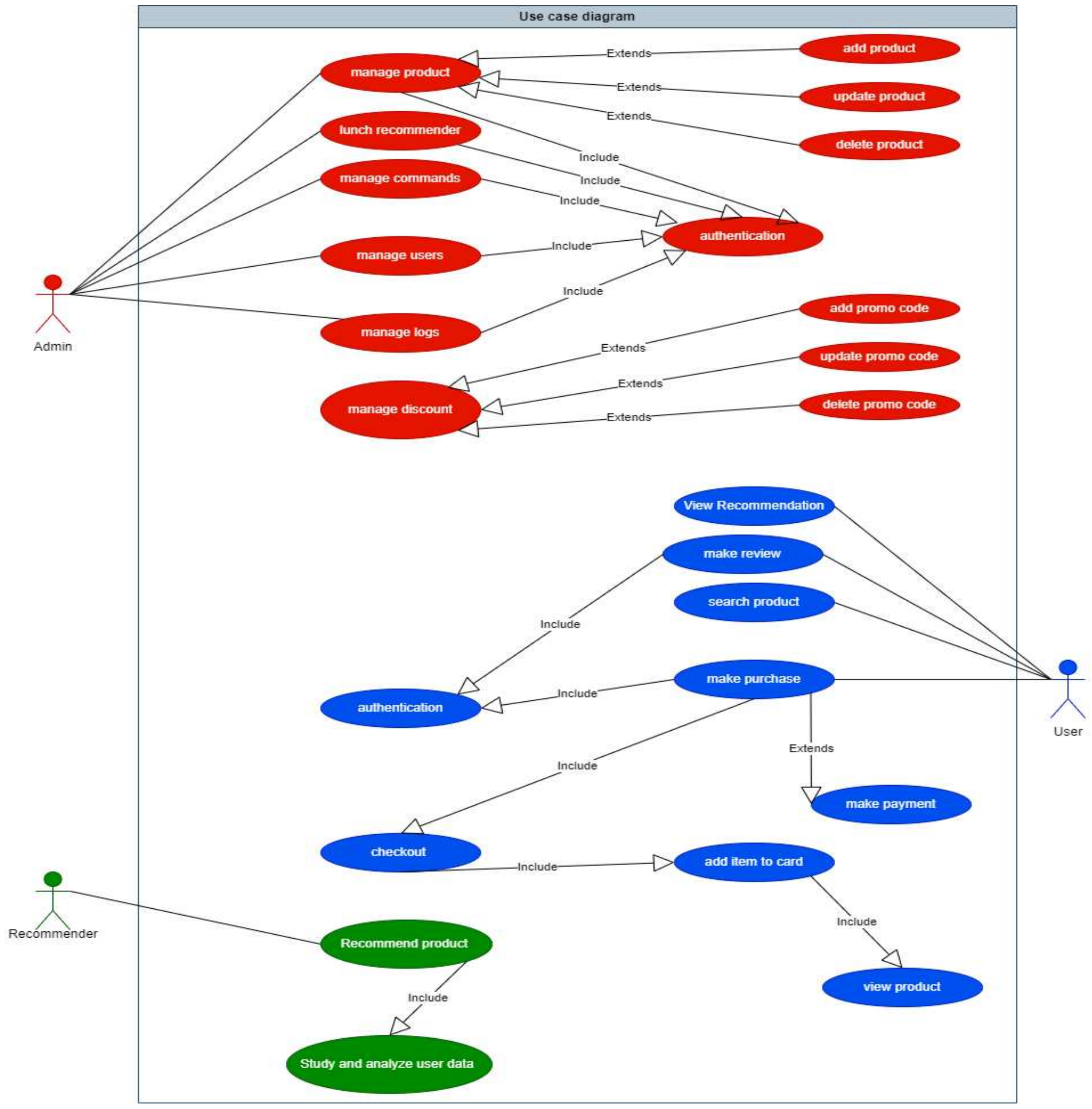


Figure 3. 2: Use case diagram

3.3.3 Class Diagram

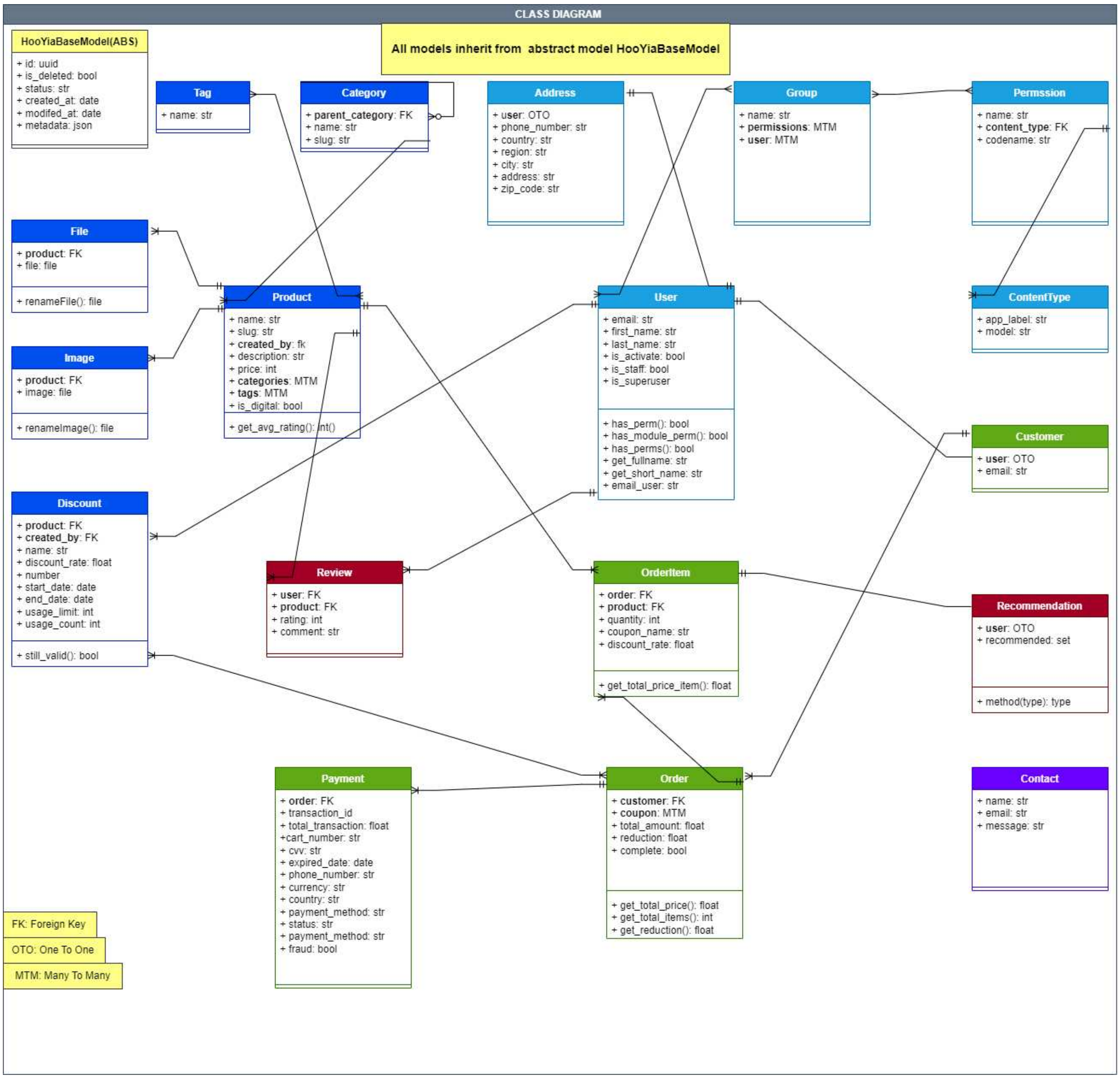


Figure 3. 3: Class diagram

The Figure 3.3 depicted the static structure of the app by illustrating the classes, their attributes, and the relationships between them. It provided a visual representation of the app's data model, defining the entities and their associations. The class diagram served as a blueprint for organizing and managing the app's data and facilitated a clear understanding of the data flow within the system.

3.3.4 Sequence Diagram

The sequence diagram captured the dynamic behavior of the app by showing the sequence of interactions between objects and components. It illustrated how different modules and components collaborated to execute specific functionalities. The sequence diagram helped in identifying the order of operations and the flow of information, ensuring that the app performed the intended actions in a logical and efficient manner.

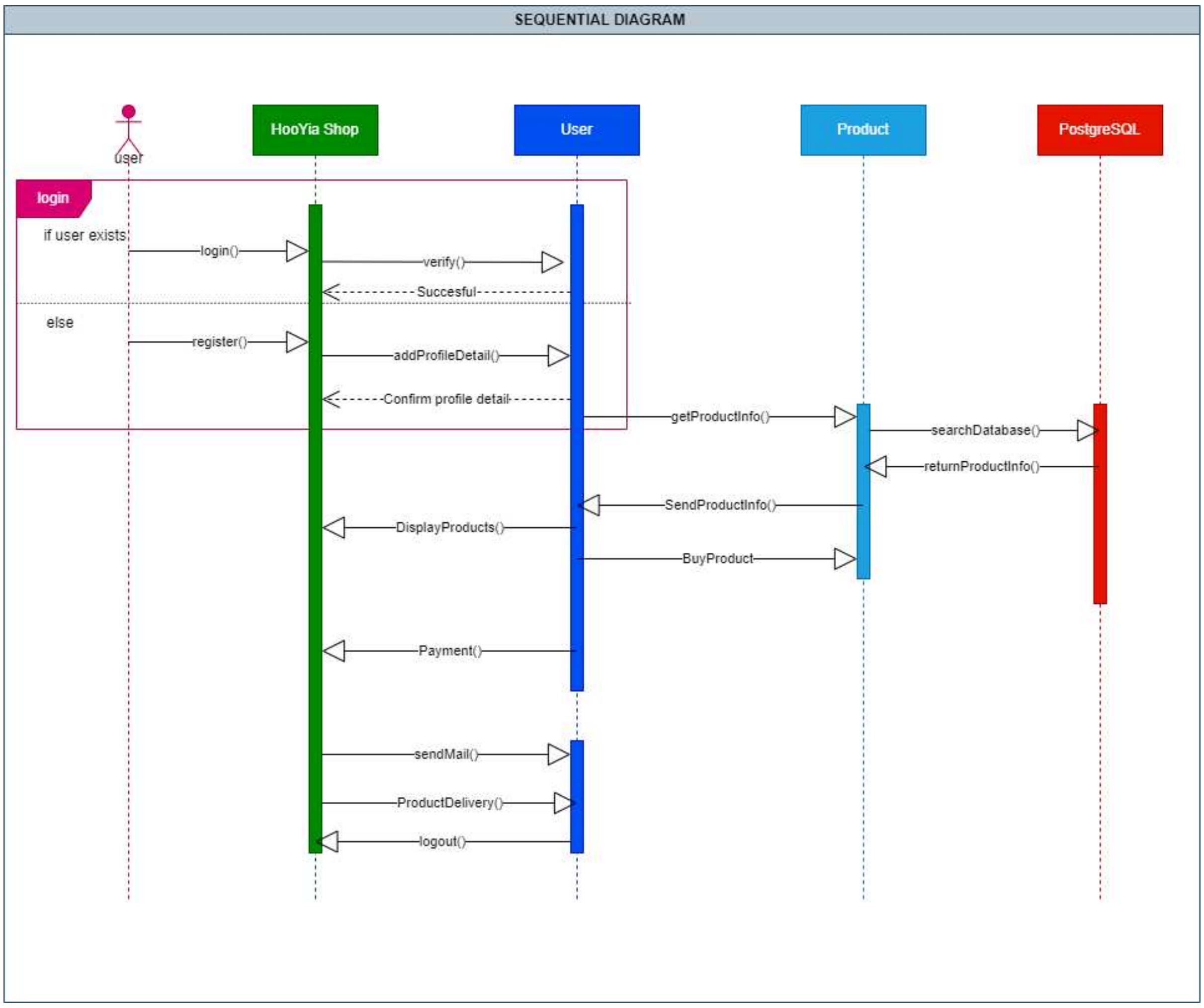


Figure 3. 4: Sequential diagram

3.3.5 State Transition Diagram

The state transition diagram depicted the various states and transitions that an object or system could undergo in response to events or actions. It visually represented the behavior of the app and helped in understanding the system's response to different inputs and events. The state transition diagram provided insights into the app's behavior and allowed for the identification of potential edge cases or error handling scenarios.

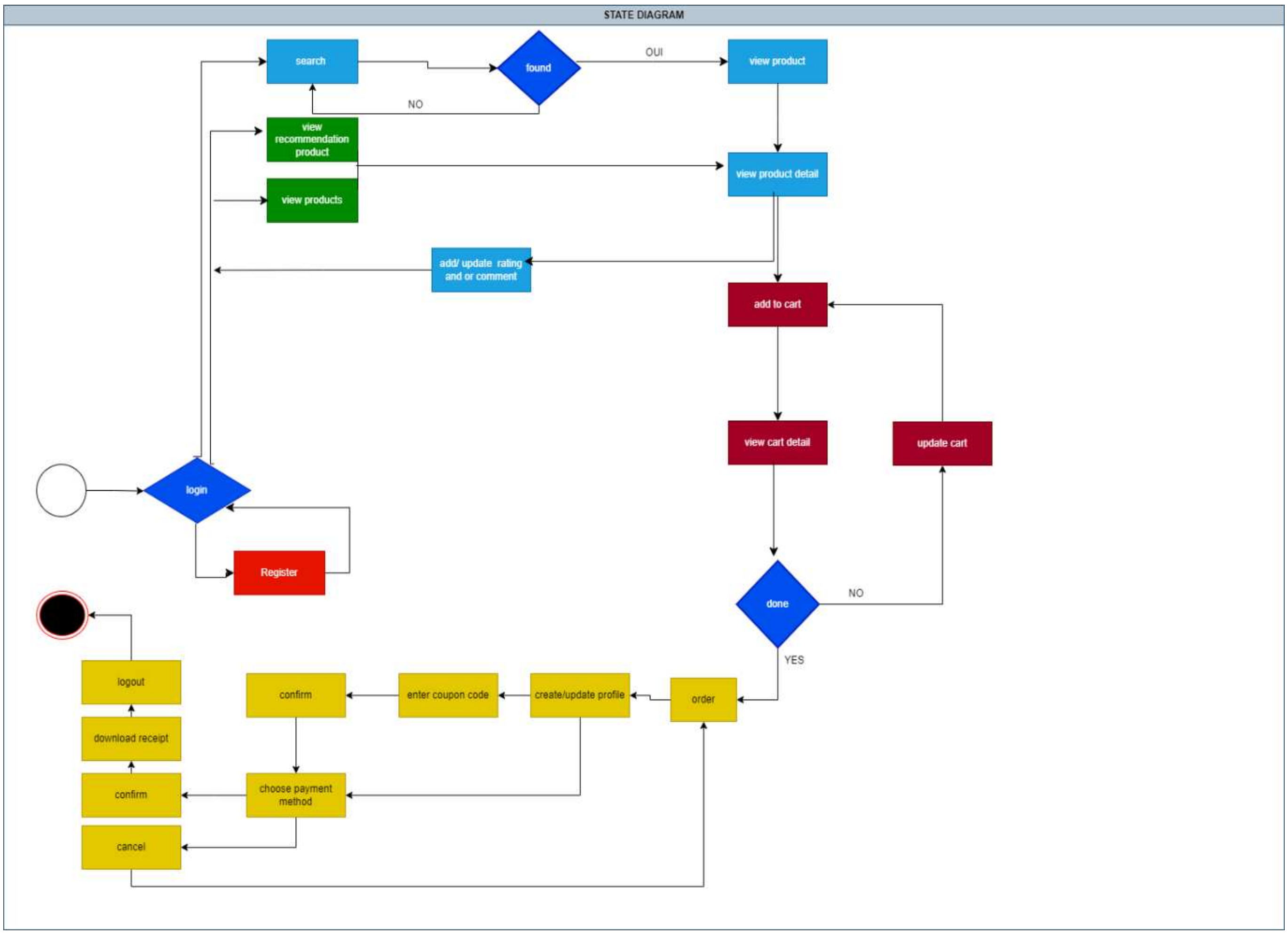


Figure 3. 5: State Diagram

3.3.6 Activity Diagram

The activity diagram illustrated the flow of activities and decision points within specific processes or scenarios of the app. It showcased the sequence of actions and steps taken by users and administrators to accomplish specific tasks. Separate activity diagrams were created for user and admin roles to outline their respective workflows, guiding the design and implementation of the app's user interface and functionality.

3.3.6.1 Activity Diagram for User

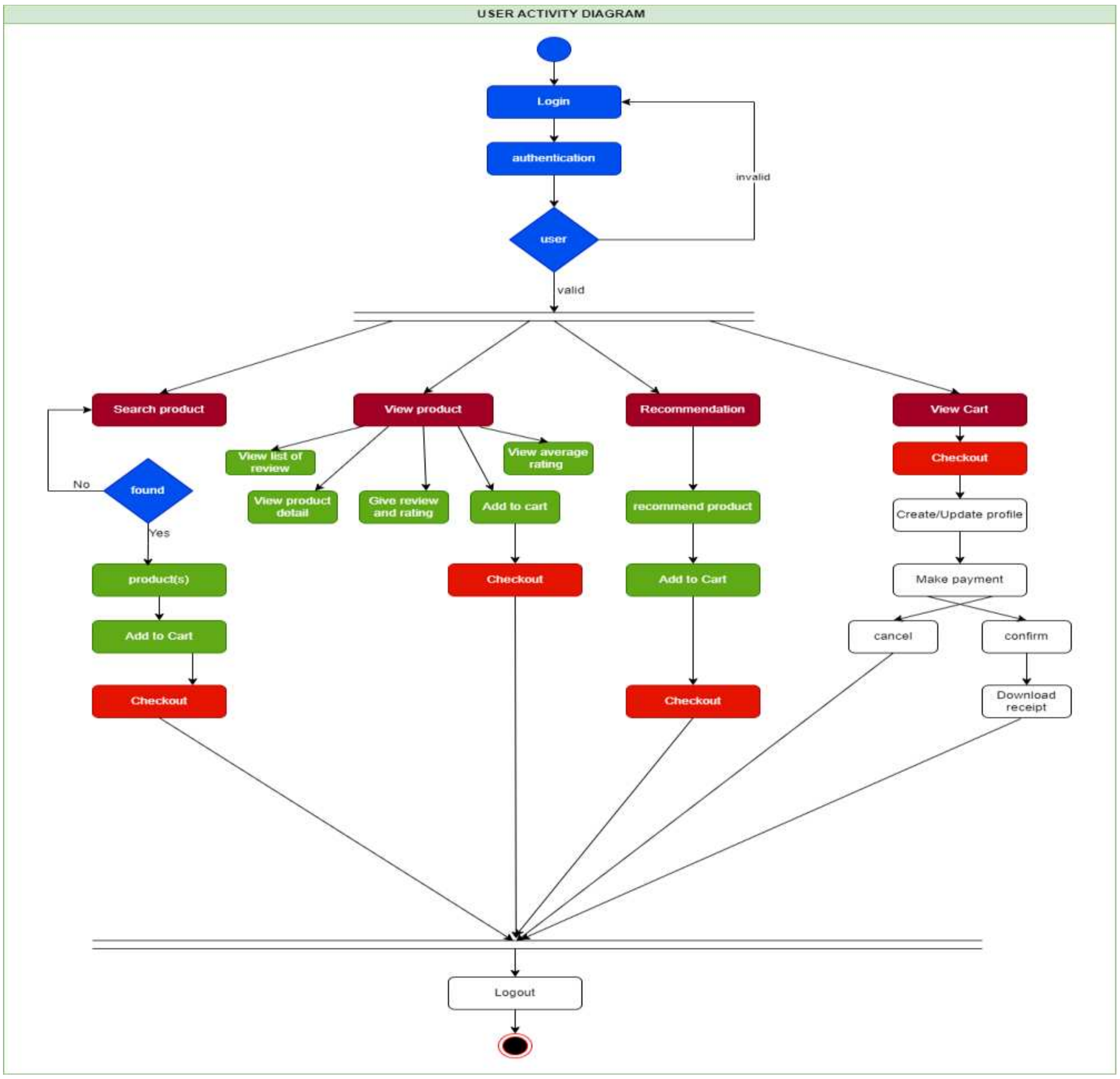


Figure 3. 6: User Activity Diagram

3.3.6.2 Activity Diagram for Admin

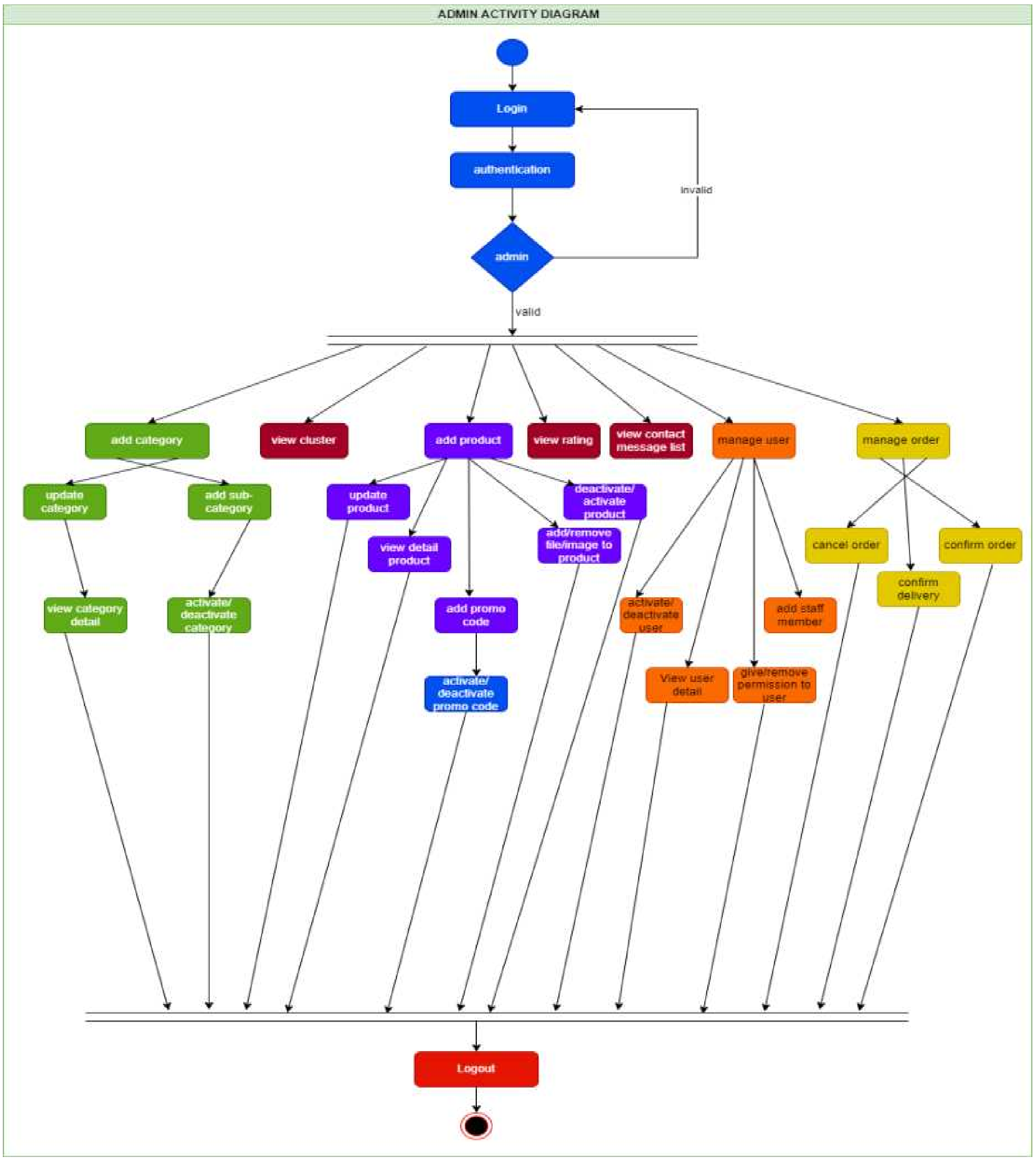


Figure 3. 7: Admin activity diagram

3.4 Architecture

The architecture of the eCommerce app with a recommendation system was designed to ensure efficient functionality, scalability, and seamless integration between different components. This section discusses the architecture of both the eCommerce app and the recommendation system.

3.4.1 Ecommerce App Architecture

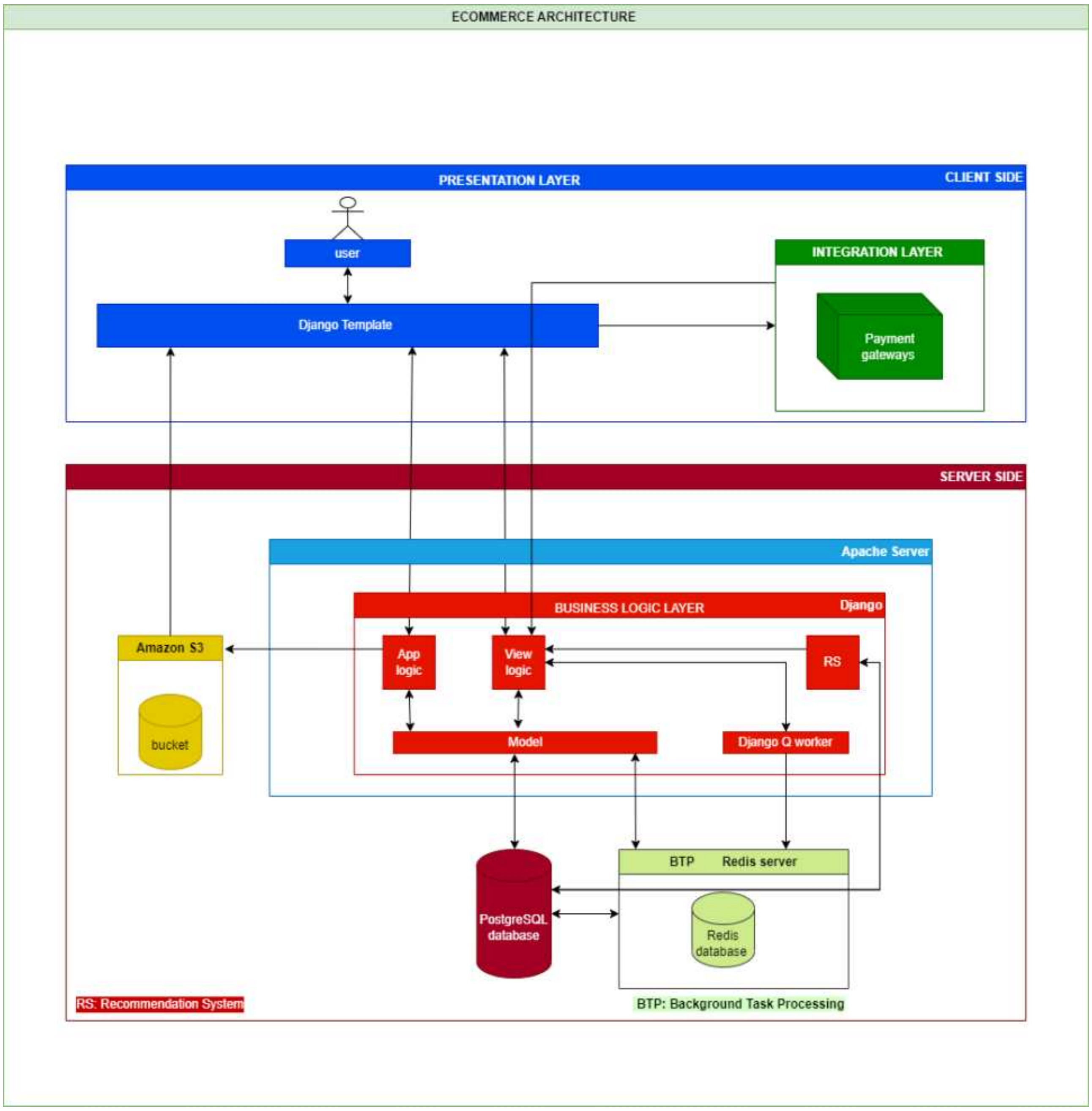


Figure 3. 8: Ecommerce app architecture

The architecture of the eCommerce app followed a client-server model, where the client-side was responsible for presenting the user interface and handling user interactions, while the server-side managed the business logic and data processing. The app utilized the Django framework, which provided a Model-View-Controller (MVC) architectural pattern.

The app's architecture consisted of the following key components:
Presentation Layer: This layer encompassed the user interface elements, including HTML templates, CSS styling, and JavaScript for interactivity. It enabled users to interact with the app and view product listings, make purchases, and manage their profiles.

Business Logic Layer: This layer implemented the core functionalities of the eCommerce app, such as user authentication, product catalog management, cart management, and order processing. It encapsulated the business rules and processes that governed the app's operations.

Data Access Layer: The data access layer facilitated communication with the database and managed data retrieval, storage, and updates. PostgreSQL, the chosen database management system, stored and managed the app's data, including user profiles, product information, order details, and recommendation data.

Integration Layer: The integration layer handled external services and APIs, enabling interactions with payment gateways, shipping providers, and other third-party services required for the app's operations.

Background Task Processing: The architecture included the integration of Redis server, an in-memory data structure store, for background task processing. Redis allowed for efficient handling of time-consuming or resource-intensive tasks, such as sending email notifications, generating reports, and processing large data sets.

This architecture ensured separation of concerns and modularity, making the app more maintainable, extensible, and scalable.

3.4.2 Recommendation System Architecture

The recommendation system architecture focused on delivering personalized product recommendations to app users. It integrated with the eCommerce app architecture and leveraged machine learning techniques for generating accurate and relevant recommendations.

To develop our recommendation system, we followed a series of steps that encompassed data gathering, data preprocessing, feature engineering, algorithm selection, and evaluation. Here are the key steps we undertook:

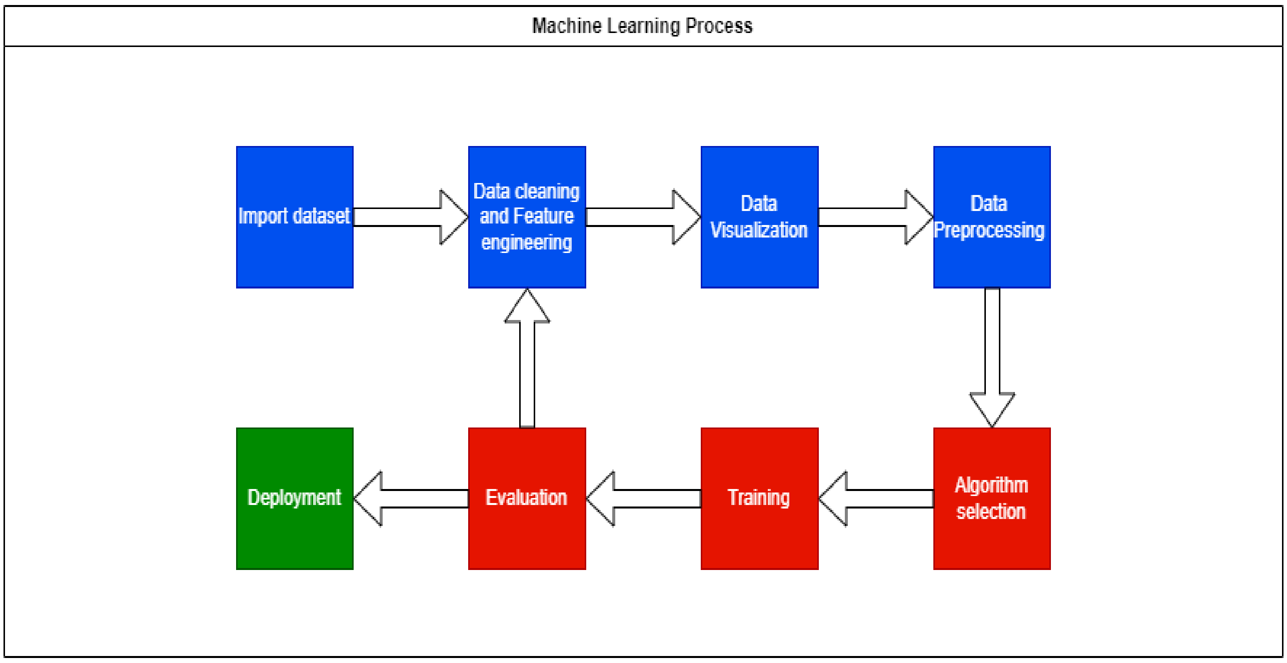


Figure 3. 9: Machine Learning Process Used

3.4.2.1 Data Gathering: We collected data from GroupLens, a prominent e-commerce platform, to obtain a diverse and representative dataset. This dataset consisted of user-item interactions, product features, and contextual information.

Data Cleaning and Feature Engineering: We performed data cleaning tasks to handle missing values, outliers, and inconsistencies in the dataset. Additionally, we conducted feature engineering to extract relevant features from the raw data. This involved transforming and enriching the data to create meaningful representations for users, products, and contextual information.

Data Preprocessing: Before feeding the data into our recommendation system, we preprocessed it to ensure it was in a suitable format. This step included encoding categorical variables, normalizing numerical features, and splitting the dataset into training and testing sets.

Data Visualization: We utilized data visualization techniques to gain insights into the dataset and understand the distribution of user-item interactions, explore feature correlations, and identify any patterns or trends that could inform our recommendation system's design.

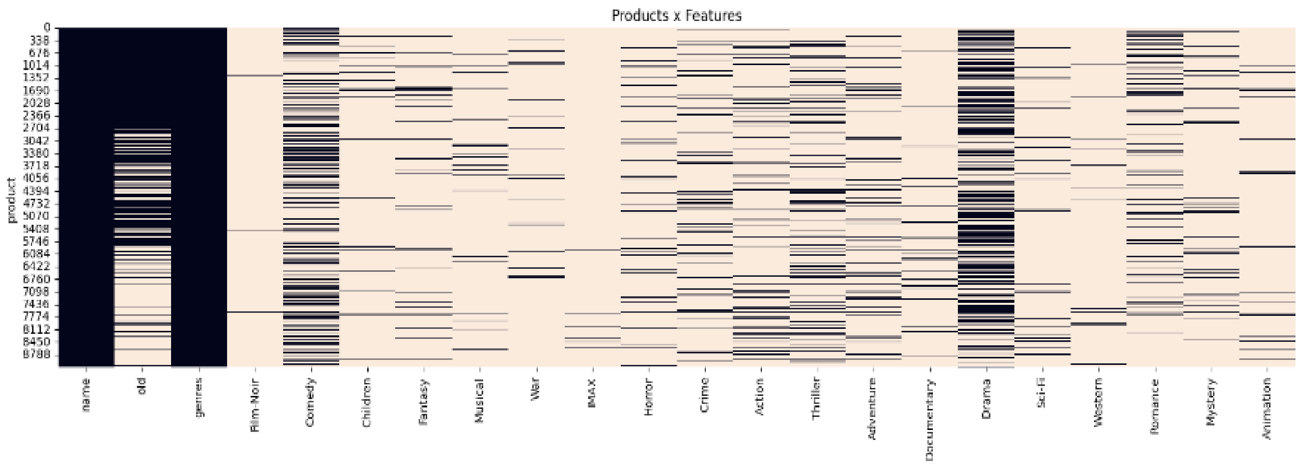


Figure 3. 10: Product x Features visualization

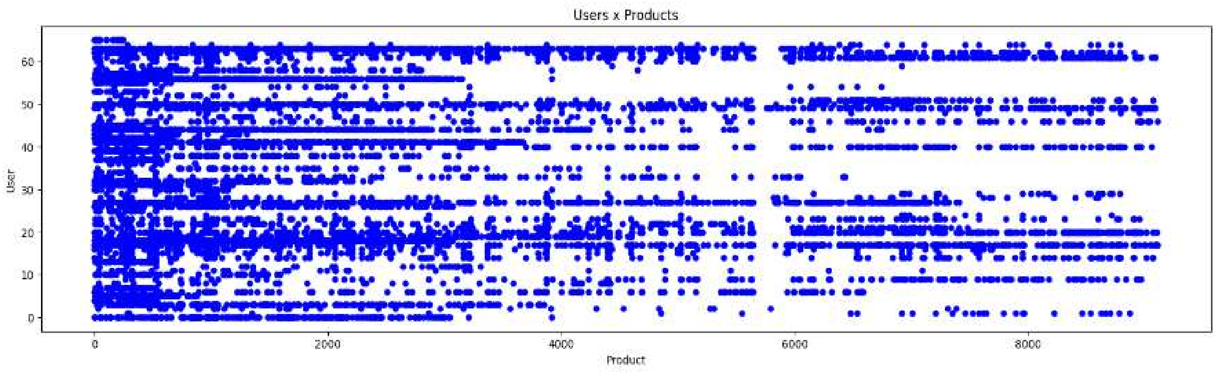


Figure 3. 11: Users x Products Visualization

Algorithm Selection: We carefully considered various recommendation algorithms, ultimately opting for a hybrid approach to leverage the strengths of collaborative filtering, neural network (Multi-Layer Perceptron), content-based filtering, and knowledge-based techniques. This selection allowed us to combine different methodologies and capture diverse aspects of user preferences and item characteristics.

Model Development: We constructed the recommendation architecture by designing and connecting the different components, including collaborative filtering, content-based filtering, and knowledge-based filtering. This architecture incorporated matrix factorization, neural networks, and dense layers to extract latent representations, capture interactions, and incorporate contextual information.

Evaluation: We assessed the performance of our recommendation system using a combination of evaluation metrics, including:

1. **Mean Absolute Error (MAE)**
   The MAE measures the average absolute difference between the predicted ratings and the actual ratings provided by users. It gives us an understanding of how closely our system's predictions align with the true ratings. A lower MAE indicates better accuracy in rating estimation.

2. **Root Mean Square Error (RMSE)**
   The RMSE calculates the square root of the average of the squared differences between the predicted ratings and the actual ratings. Similar to MAE, RMSE provides an indication of the accuracy of our rating predictions. It penalizes larger prediction errors more heavily than MAE and gives us insight into the magnitude of the errors.

3. **Mean Reciprocal Rank (MRR)**
   MRR measures the quality of the ranking of the first relevant item in the recommendation list. It helps us evaluate how well our system is able to present the most relevant items at the top of the list. A higher MRR indicates better ranking performance.

4. **Recall**
   Recall measures the proportion of relevant items that are successfully recommended. It provides insights into the system's ability to retrieve relevant items from the dataset. Higher recall indicates a greater coverage of relevant recommendations.

5. **Precision**
   Precision represents the proportion of relevant items among the recommended items. It helps us assess the accuracy of the recommendations by measuring the proportion of relevant items in the recommendation list. Higher precision indicates a lower likelihood of irrelevant recommendations.

6. **Accuracy**

   Accuracy measures the overall correctness of the system's recommendations. It calculates the ratio of correct recommendations to the total number of recommendations made. Higher accuracy indicates a higher percentage of correct recommendations.

We chose these evaluation metrics because they collectively cover various aspects of our recommendation system's performance, including accuracy, ranking quality, coverage of relevant items, and the balance between precision and recall. By considering these metrics, we gain a comprehensive understanding of the system's effectiveness in providing accurate and relevant recommendations to users.

By following these steps, we were able to develop a comprehensive recommendation system that harnessed data from GroupLens, employed data cleaning and feature engineering techniques, performed data preprocessing, employed visualization methods, selected appropriate algorithms, and evaluated the system's performance.

In our recommendation architecture, we designed a hybrid approach that combines neural network, collaborative filtering, content-based filtering, and knowledge-based techniques to enhance the accuracy and diversity of our recommendations.

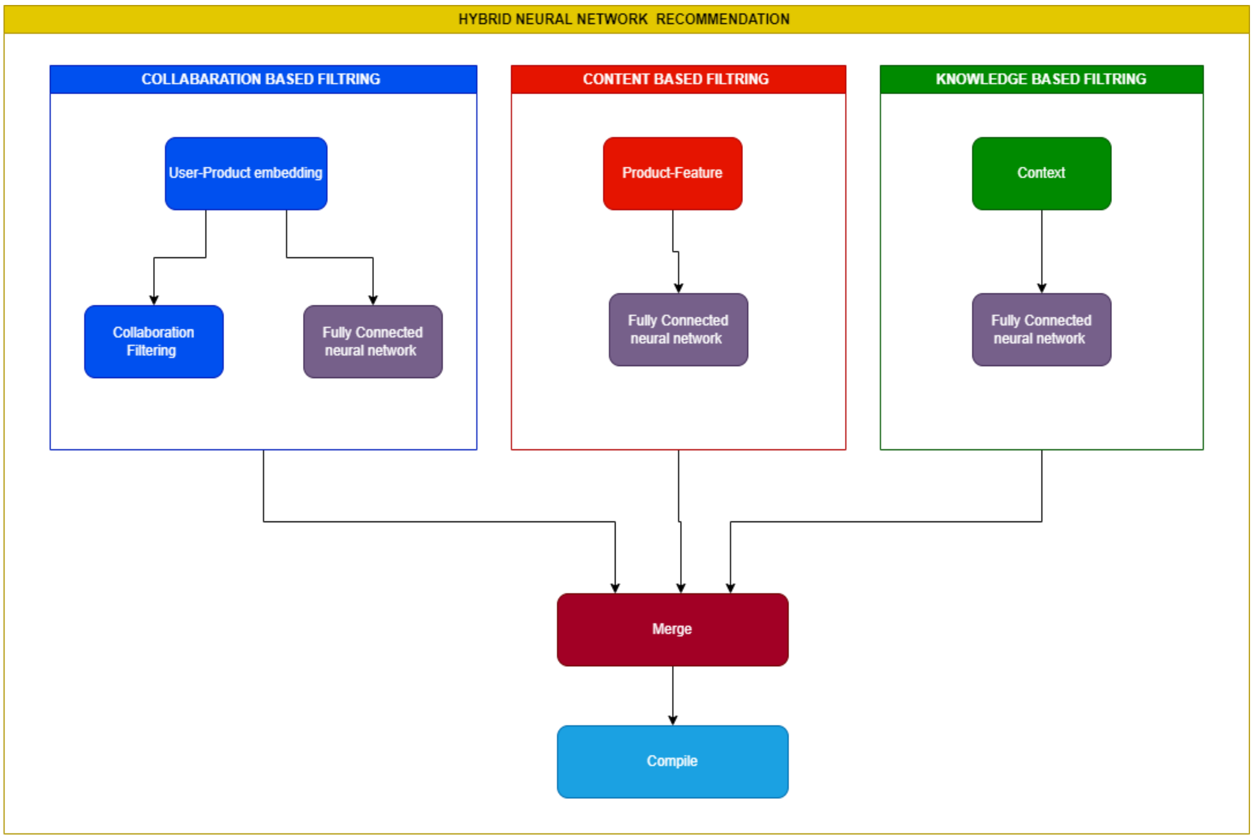

HYBRID NEURAL NETWORK RECOMMENDATION
COLLABARATION BASED FILTRING
User-Product embedding
Collaboration Filtering
Fully Connected neural network
CONTENT BASED FILTRING
Product-Feature
Fully Connected neural network
KNOWLEDGE BASED FILTRING
Context
Fully Connected neural network
Merge
Compile

Figure 3. 12: Hybrid Recommendation architecture

3.4.2.1 Models Formula Used

Collaborative Filtering (CF)

Algorithm: Matrix Factorization

Explanation: Collaborative Filtering (CF) uses matrix factorization to represent users and products in a lower-dimensional space. It learns embeddings for users and products, and the similarity between them is calculated using the dot product.

Formula:

CF(x_users, x_products) = dot(cf_xusers, cf_xproducts)

In CF, matrix factorization decomposes the user-item interaction matrix into two lower-rank matrices: one representing user embeddings (cf_xusers) and the other representing item embeddings (cf_xproducts). The dot product between these embeddings is taken to calculate the similarity between users and items.

Neural Network (NN):

Algorithm: Neural Network

Explanation: The Neural Network (NN) model uses a feed-forward architecture to learn non-linear interactions between user and product features. It leverages dense layers to capture complex relationships.

Formula

NN(x_users, x_products) = dense(concatenate(nn_xusers, nn_xproducts))

In NN, the user and product embeddings (nn_xusers and nn_xproducts) are concatenated and passed through one or more dense layers. The dense layer performs an affine transformation on the concatenated features, followed by an element-wise activation function.

Content-Based Filtering:

Algorithm: Content-Based Filtering

Explanation: Content-Based Filtering uses a dense neural network to learn representations for products based on their features. The dense layer captures patterns and relationships within the product features.

Formula

Content-Based(x_features) = dense(features_in)

In Content-Based Filtering, the product features (x_features) are passed through a dense layer. The dense layer applies an affine transformation to the input features, followed by an activation function. It learns to extract meaningful representations from the product features, enabling the model to capture patterns and relationships.

Knowledge-Based Filtering:

Algorithm: Knowledge-Based Filtering

Explanation: Knowledge-Based Filtering also uses a dense neural network to learn representations for contextual factors that influence recommendations. The dense layer captures patterns and relationships within the contextual factors.

Formula

Knowledge-Based(x_contexts) = dense(contexts_in)

In Knowledge-Based Filtering, the contextual factors (x_contexts) are fed into a dense layer. The dense layer applies an affine transformation to the input, followed by an activation function. By learning from the contextual factors, the dense layer can capture patterns and relationships that influence the recommendations.

Role of Activation Function:

Activation functions introduce non-linearity to the output of a layer. They are essential for the network to learn complex relationships and make non-linear transformations. The choice of activation function depends on the specific requirements and characteristics of the problem.

ReLU (Rectified Linear Unit) is a commonly used activation function in dense layers due to its simplicity and effectiveness.

The advantages of ReLU include computational efficiency and its ability to mitigate the vanishing gradient problem. It allows the network to learn quickly by preserving positive gradients during backpropagation. Additionally, ReLU avoids the saturation problem that can occur with functions like sigmoid or tanh.
The Dense layer represents the output layer of the model. It applies an affine transformation to the input and produces a single output value.

For model training and evaluation, we compiled the hybrid model using the **Adam optimizer** (is an adaptive optimization algorithm commonly used in machine learning and deep learning models that combines the benefits of both Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) methods.), mean absolute error loss function, and mean absolute percentage error, recall, precision, root mean squared error and mean reciprocal rank as a metric. This configuration allowed us to optimize the model's performance and assess its accuracy in predicting user preferences and satisfaction with recommended products.

By adopting a hybrid approach that combines **neural network, collaborative filtering, content-based filtering, and knowledge-based techniques**, we were able to leverage the strengths of each method while mitigating their individual limitations. This resulted in a more comprehensive and accurate recommendation system for our e-commerce app.

3.4.2.2 Number of neurons per layer:

In the context of a neural network, a "neuron" refers to an individual computational unit or node within a neural network layer. Neurons are inspired by the biological neurons in the human brain and serve as the fundamental building blocks of artificial neural networks.

Each neuron in a neural network layer performs a specific computation on its input data. It takes input values, applies weights and biases to these inputs, performs a mathematical operation (such as a dot product or element-wise activation), and produces an output.

In a dense layer (also known as a fully connected layer), each neuron is connected to every neuron in the previous layer. The output of each neuron in one layer serves as input to all neurons in the next layer.

The number of neurons in a layer determines the expressive capacity of the neural network. Each neuron can learn to recognize specific patterns or features from the input data and contribute to the overall representation of the data.

The neurons in a neural network collectively work together to process and transform input data, enabling the network to learn complex patterns and make predictions or classifications based on the learned representations.

Collaborative Filtering

Matrix Factorization:

cf_xusers_emb: The number of neurons in this layer is equal to the **embeddings_size**.

cf_xproducts_emb: The number of neurons in this layer is equal to the **embeddings_size**.

cf_xx: There is no specific number of neurons in this layer as it performs a dot product calculation.

Neural Network:

Neural Network:

nn_xx: The number of neurons in this layer is **int(embeddings_size/2).**

Content-Based Filtering:

features_x: The number of neurons in this layer is equal to the **feat**, which represents the length of the product features.

Knowledge-Based Filtering:

context_x: The number of neurons in this layer is equal to the **ctx**, which represents the length of the contextual factors.

Output Layer:

y_out: The output layer has 1 neuron, representing the final predicted output.

To summarize, the specific number of neurons in each layer may vary based on the values assigned to the **embeddings_size, feat, and ctx** variables.

In conclusion, our hybrid recommendation architecture combines collaborative filtering, content-based filtering, and knowledge-based techniques to provide enhanced recommendations in our e-commerce app. By integrating these different approaches, we leverage user-item interactions, product features, and contextual information to generate diverse and personalized recommendations.

The effectiveness of our hybrid approach will be evaluated and discussed in detail in Chapter 4, where we present the results of our experiments and performance evaluations. This chapter will provide insights into the comparative analysis of the different recommendation techniques employed, highlighting the strengths and weaknesses of each component within the hybrid architecture. By examining the results, we will gain a deeper understanding of how the combination of these techniques contributes to the overall recommendation quality and user satisfaction.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Results

4.1.1 Ecommerce App

In this section, we present the results of our ecommerce app, providing a comprehensive overview of its functionality from start to finish. We include screenshots to illustrate each step of the user journey, showcasing various key pages and features.

4.1.1.1 Home Page

Here, we display the home page of the app, demonstrating its layout and design.

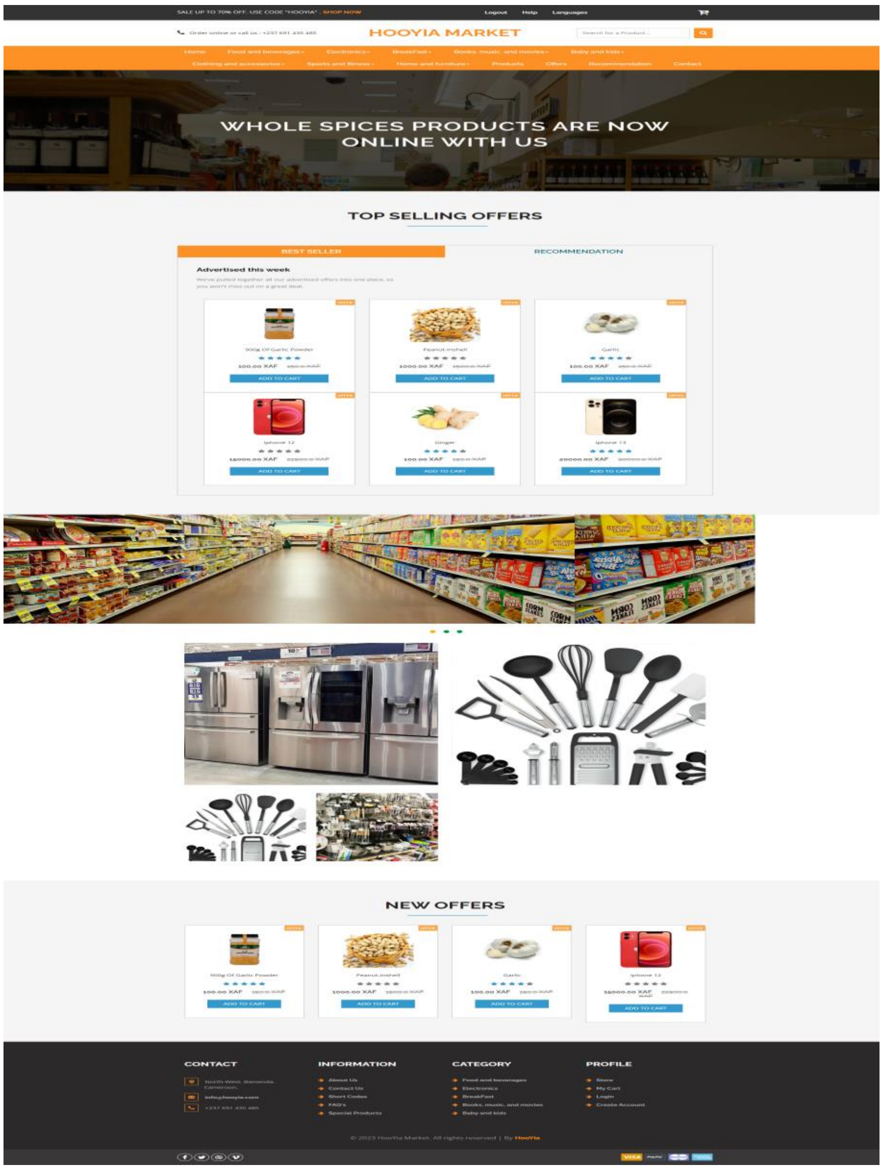


Figure 4. 1: Home page

4.1.1.2 Product Page

Users can browse and view the available products on figure 4.2.

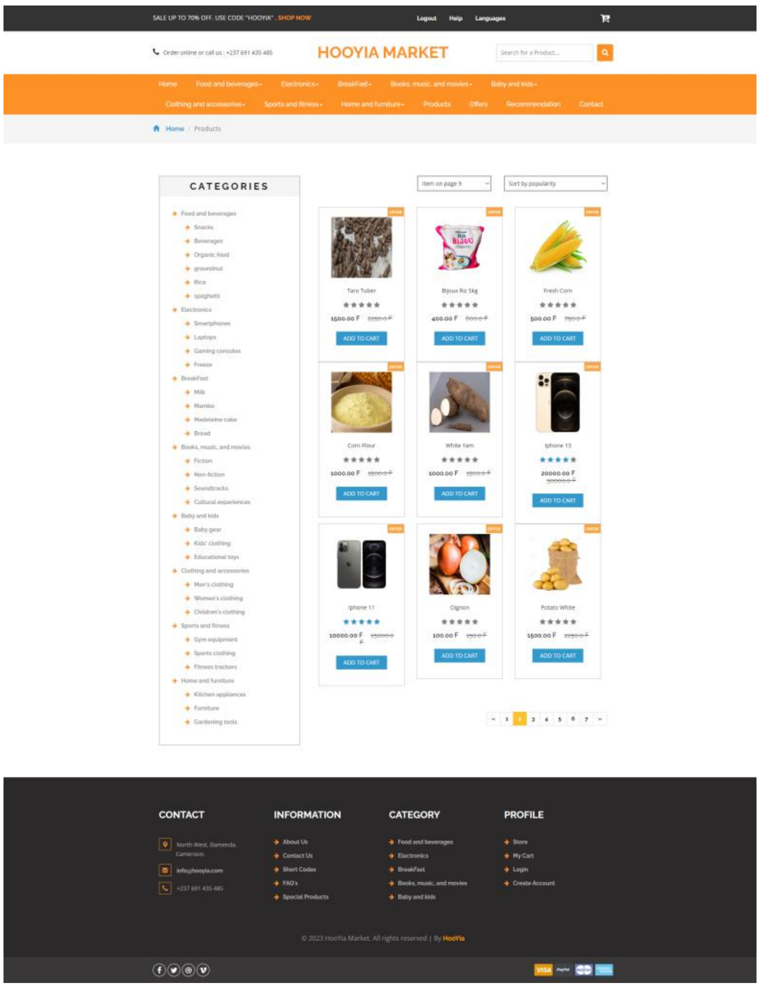


Figure 4. 2: Product Page

4.1.1.3 Product Detail Page

This page provides detailed information about a specific product, including images, descriptions, pricing, and reviews.

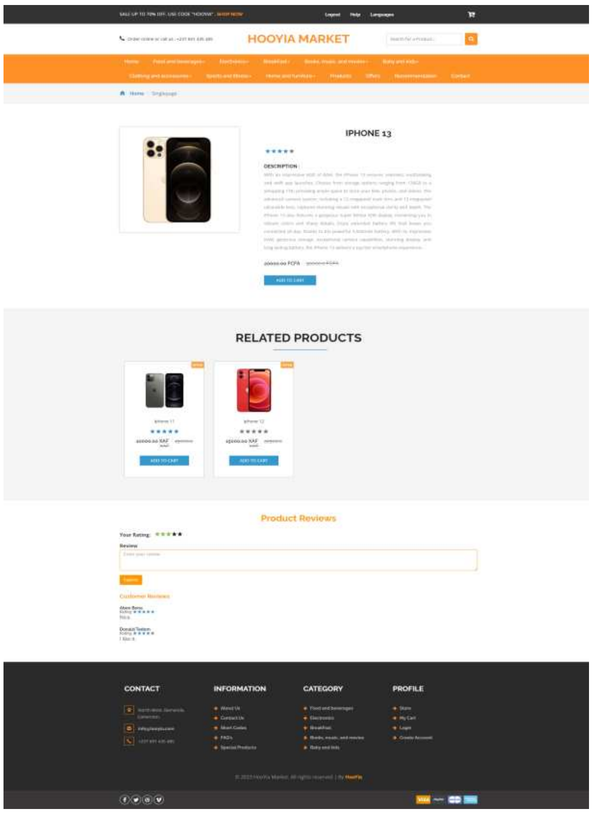


Figure 4. 3: Product Detail Page

4.1.1.4 Adding Items to Cart

Users have the option to add items to their cart for future purchase.

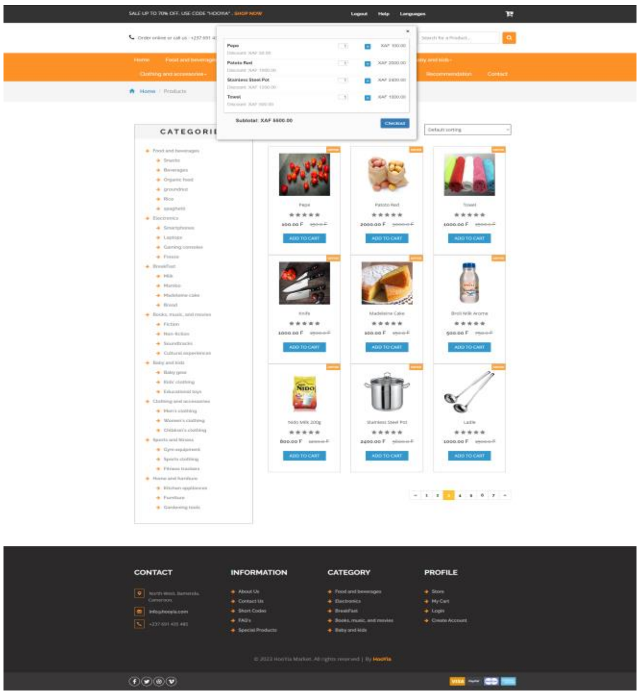


Figure 4. 4: Add Item Card Popup

4.1.1.5 Search Functionality

The app includes a search feature, allowing users to find specific products based on their preferences.

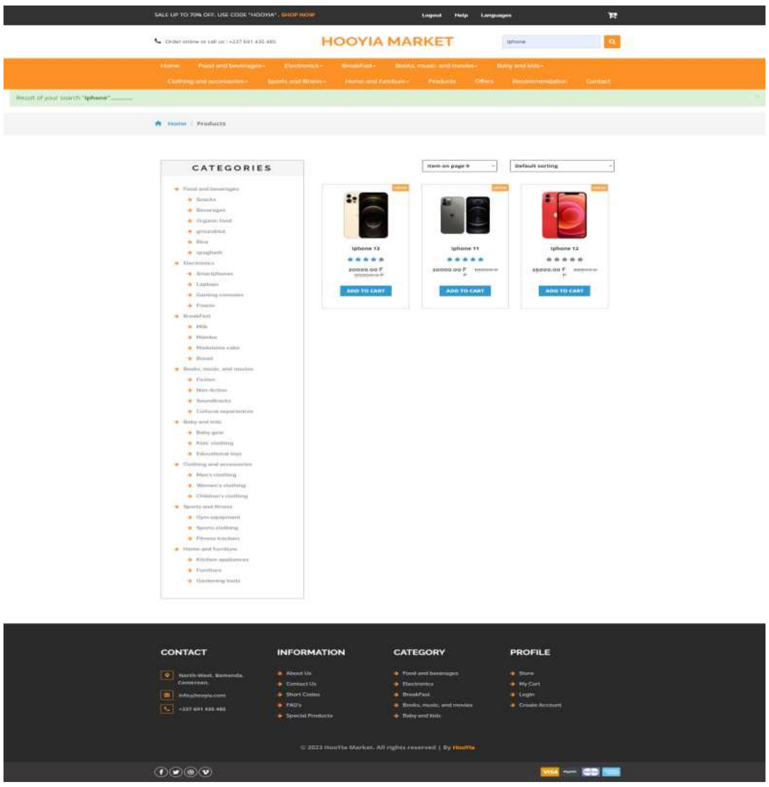


Figure 4. 5: Search Functionality

4.1.1.6 Checkout Process

Users need to log in or create an account to proceed with the checkout process.

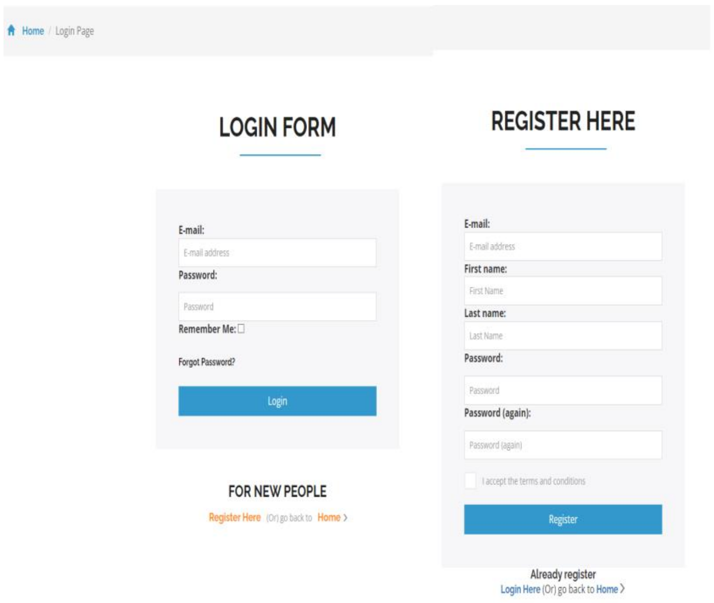
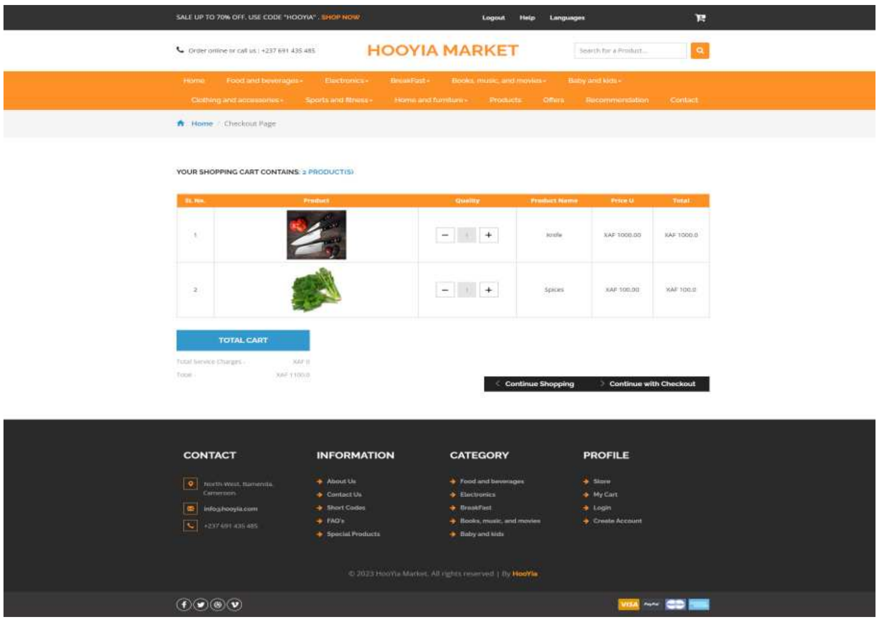


Figure 4. 6: Login and Register Page

4.1.1.7 Cart Page

Once logged in, users can review and modify the items in their cart.



YOUR SHOPPING CART CONTAINS: 2 PRODUCT(S)

Sl. No.	Product	Quality	Product Name	Price U	Total
1		1	knife	XAF 1000.00	XAF 1000.0
2		1	Spices	XAF 100.00	XAF 100.0

Total Service Charges : XAF 0
Total : XAF 1100.0

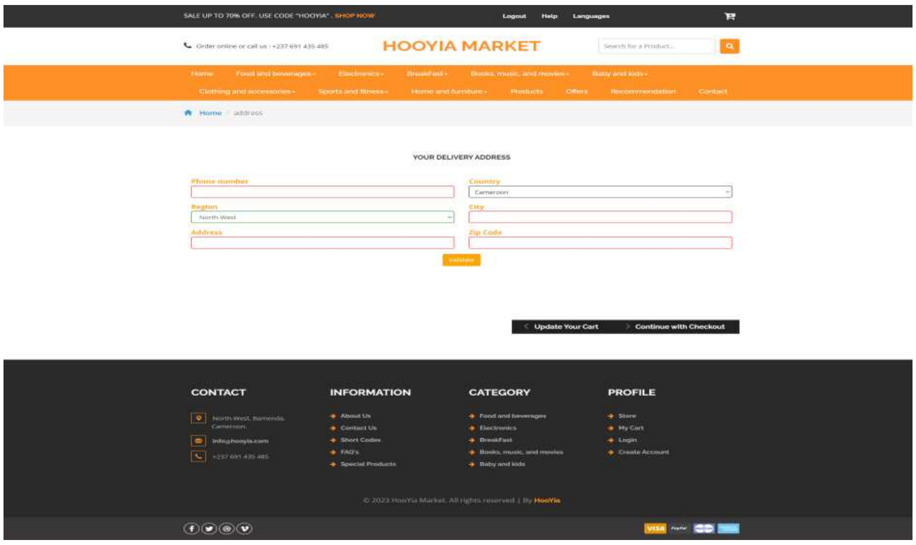
North West, Bamenda, Cameroon.
info@hooyia.com
+237 691 435 485

© 2023 HooYia Market. All rights reserved | By HooYia

Figure 4. 7: Card Page

4.1.1.8 Delivery Information

Users are prompted to provide their delivery details during the checkout process.



YOUR DELIVERY ADDRESS

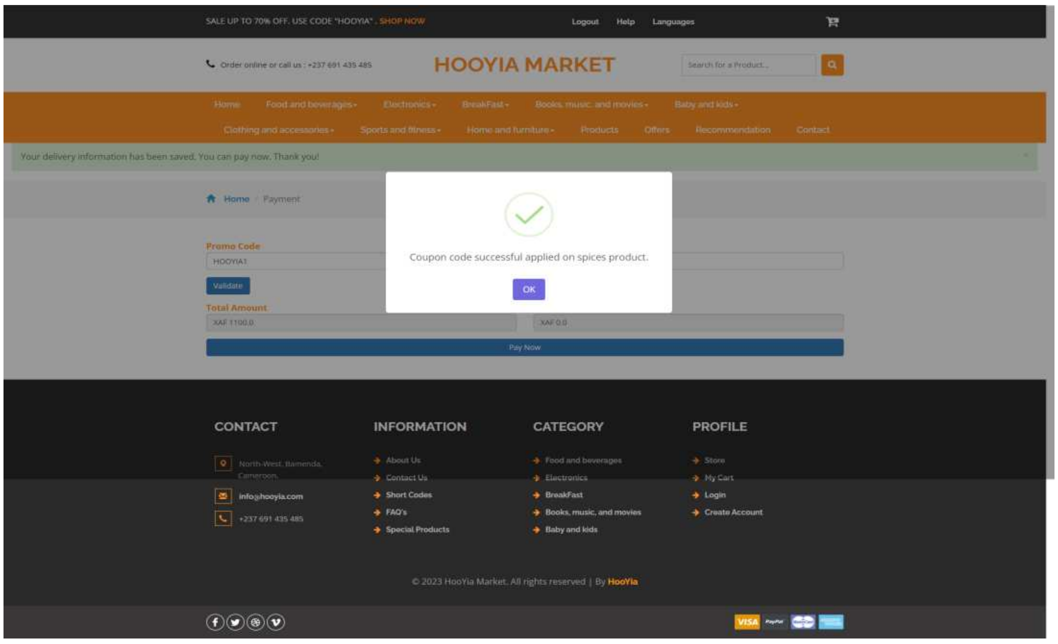
Phone number:
Country: Cameroon
Region: North West
City:
Address:
Zip Code:

© 2023 HooYia Market. All rights reserved | By HooYia

Figure 4. 8: Delivery Information Page

4.1.1.9 Promo Code

Users have the option to apply a promo code for potential discounts.



Your delivery information has been saved. You can pay now. Thank you!

Coupon code successful applied on spices product.

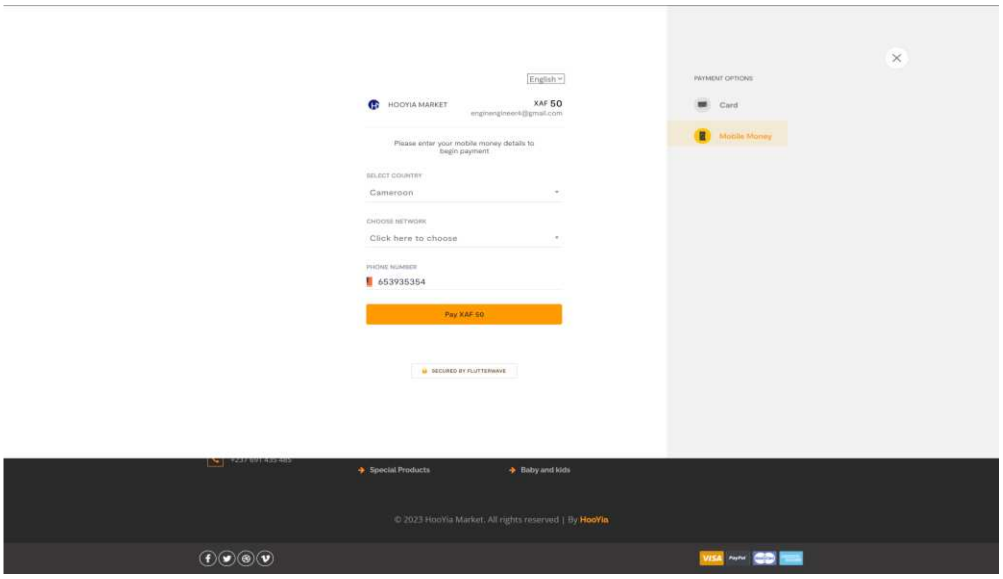
Promo Code: HOOYIA1

Total Amount: XAF 1100.0 — XAF 0.0

Figure 4. 9: Promo Code Page

4.1.1.10 Payment Methods

Users can choose their preferred payment method, such as Orange Money, MTN Money, or credit card.



Please enter your mobile money details to begin payment

SELECT COUNTRY: Cameroon

CHOOSE NETWORK: Click here to choose

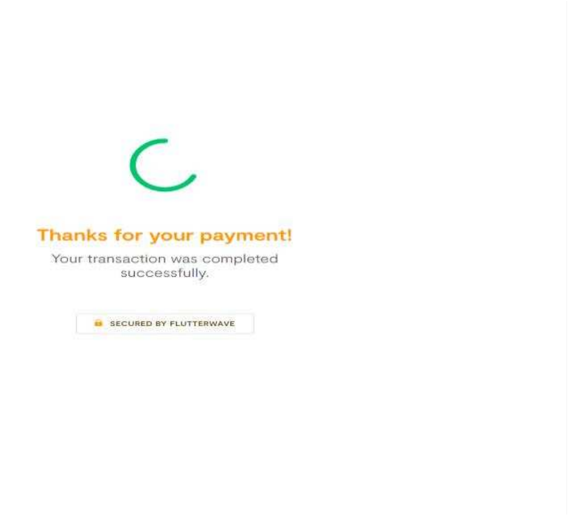
PHONE NUMBER: 653935354

Pay XAF 50

Figure 4. 10: Payment Methods Page

4.1.1.11 Transaction Status

Once the payment is successfully made, users receive an email confirming the transaction status.

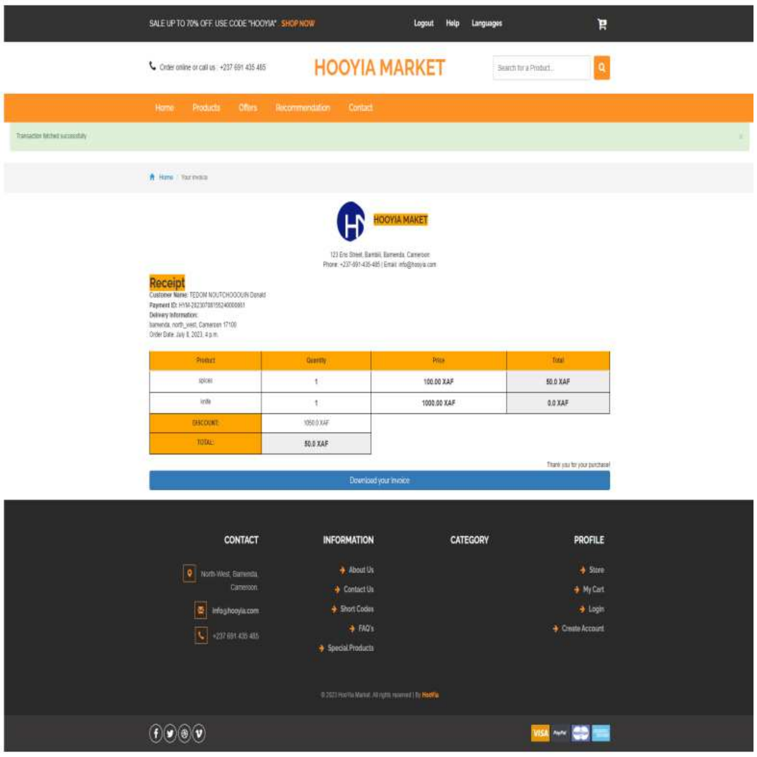

Thanks for your payment!

Your transaction was completed successfully.

Figure 4. 11: Payment Status

4.1.1.12 Invoice Generation

Users can view and download their invoice for the completed transaction.



Transaction fetched successfully

123 Eric Street, Bambili, Bamenda, Cameroon
Phone: +237-691-435-485 | Email: info@hooyia.com

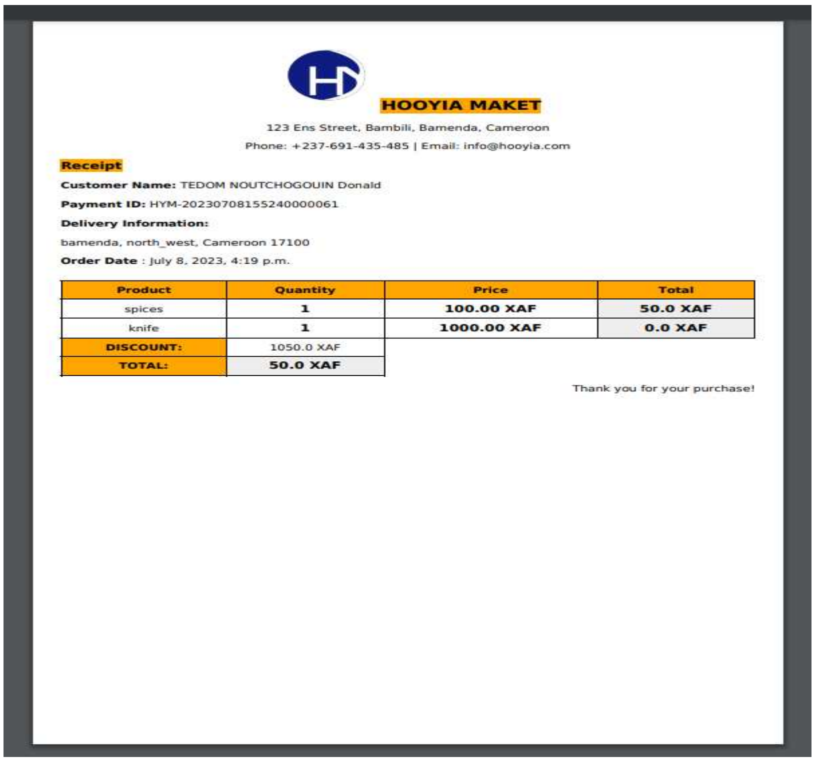
Receipt

Customer Name: TEDOM NOUTCHOGOUIN Donald
Payment ID: HYM-20230708155240000061
Delivery Information:
bamenda, north_west, Cameroon 17100
Order Date: July 8, 2023, 4 p.m.

Product	Quantity	Price	Total
spices	1	100.00 XAF	50.0 XAF
knife	1	1000.00 XAF	0.0 XAF
DISCOUNT:	1050.0 XAF		
TOTAL:	50.0 XAF		

Thank you for your purchase!

Figure 4. 12: Visualization of Invoice


HOOYIA MAKET

123 Ens Street, Bambili, Bamenda, Cameroon

Phone: +237-691-435-485 | Email: info@hooyia.com

Receipt

Customer Name: TEDOM NOUTCHOGOUIN Donald

Payment ID: HYM-20230708155240000061

Delivery Information:

bamenda, north_west, Cameroon 17100

Order Date : July 8, 2023, 4:19 p.m.

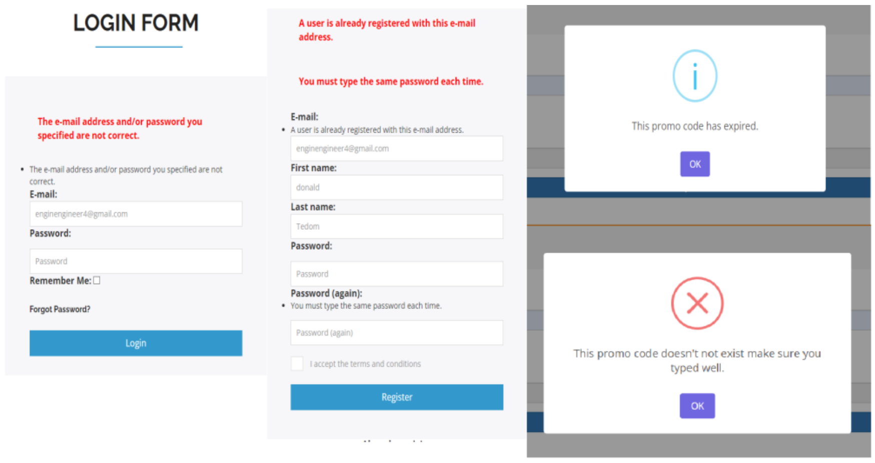
Product	Quantity	Price	Total
spices	1	100.00 XAF	50.0 XAF
knife	1	1000.00 XAF	0.0 XAF
DISCOUNT:	1050.0 XAF		
TOTAL:	50.0 XAF		

Thank you for your purchase!

Figure 4. 13: Downloaded Invoice

4.1.1.13 Error Handling

We have implemented error handling throughout the app, ensuring that users encounter appropriate messages when accessing restricted areas, providing incorrect information during account creation, or attempting to log in with incorrect credentials.


LOGIN FORM

The e-mail address and/or password you specified are not correct.

- The e-mail address and/or password you specified are not correct.

E-mail: enginengineer4@gmail.com

Password: Password

Remember Me: ☐

Forgot Password?

A user is already registered with this e-mail address.

You must type the same password each time.

E-mail:
- A user is already registered with this e-mail address.

enginengineer4@gmail.com

First name: donald

Last name: Tedom

Password: Password

Password (again):
- You must type the same password each time.

Password (again)

☐ I accept the terms and conditions

This promo code has expired.

This promo code doesn't not exist make sure you typed well.

Figure 4. 14: Errors Handling

4.1.1.14 Admin module

we have developed an admin module that provides administrators with enhanced control and visibility over various aspects of the system. The admin module serves as a centralized dashboard where authorized users can monitor and manage activities related to user management, product inventory, order tracking, and customer support.

With the admin module, administrators can efficiently handle user registrations by reviewing and managing new user accounts. They have the ability to update user information, address account-related issues, and ensure smooth user onboarding.

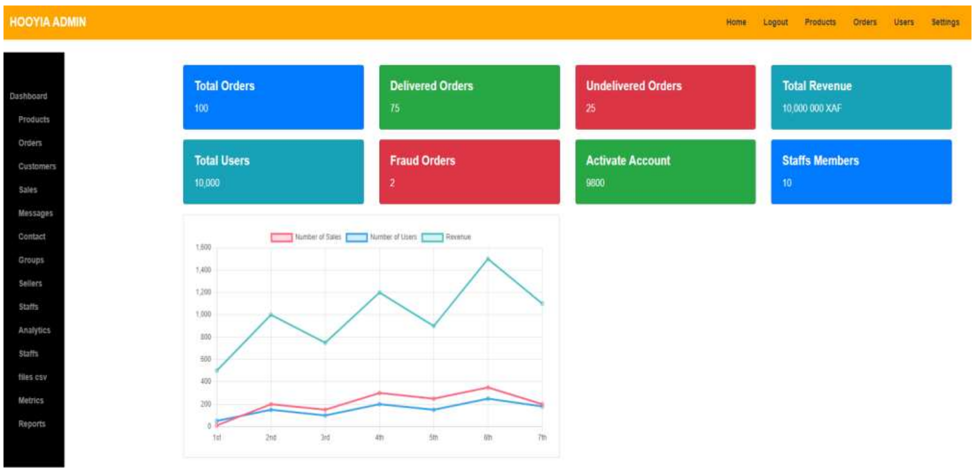
Furthermore, the module offers administrators the capability to manage the product inventory effectively. They can add new products, edit existing listings, update prices, and monitor stock levels. This functionality enables administrators to keep the product catalog up to date and ensure accurate information is presented to users.

Order tracking becomes seamless with the admin module as administrators can easily monitor the status of orders placed by users. They can track the progress of orders, manage cancellations, process returns, and handle refunds. This feature facilitates efficient order management and ensures timely delivery.

The admin module also includes command and control functionalities, allowing administrators to execute specific commands within the system. This feature enables tasks such as database backups, system maintenance, and scheduled operations to be managed effortlessly.

Customer support is an integral part of any ecommerce app, and the admin module provides administrators with access to customer support inquiries. They can promptly respond to user queries and concerns, ensuring high levels of customer satisfaction and resolving issues in a timely manner.

By incorporating the admin module into our ecommerce app, we empower administrators to efficiently manage and maintain the system. The module streamlines administrative tasks, enhances control over system operations, and facilitates effective customer support.



Total Orders
100

Delivered Orders
75

Undelivered Orders
25

Total Revenue
10,000 000 XAF

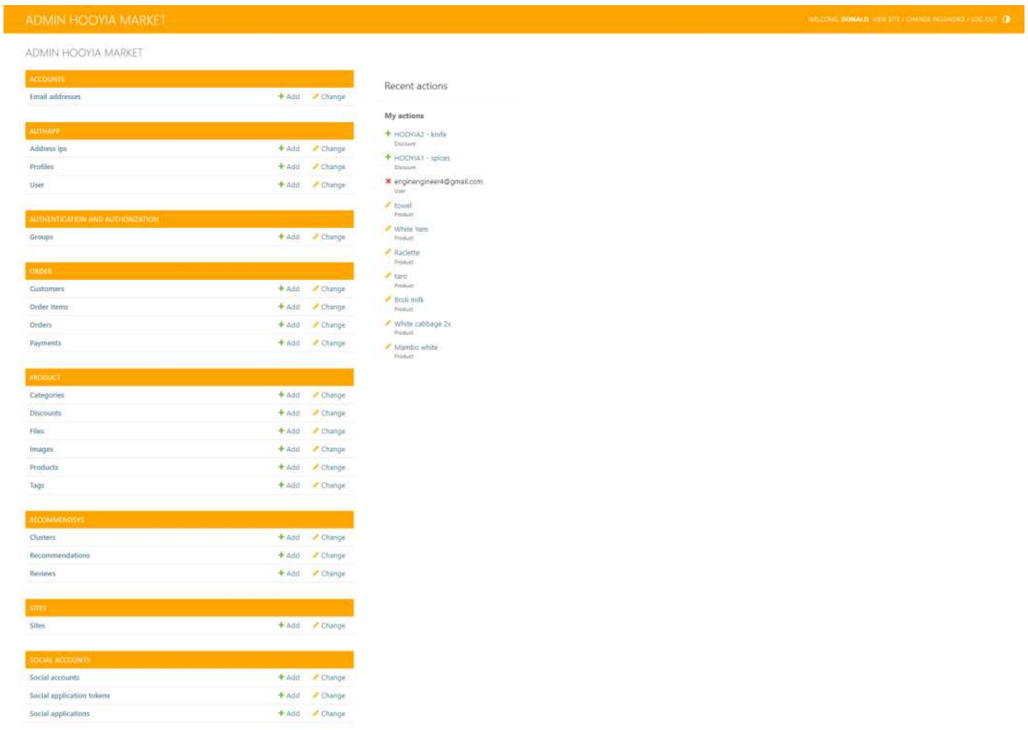
Total Users
10,000

Fraud Orders
2

Activate Account
9800

Staffs Members
10

Figure 4. 15: Administration Page



ADMIN HOOYIA MARKET

ACCOUNTS
Email addresses + Add ✎ Change

AUTHAPP
Address ips + Add ✎ Change
Profiles + Add ✎ Change
User + Add ✎ Change

AUTHENTICATION AND AUTHORIZATION
Groups + Add ✎ Change

ORDER
Customers + Add ✎ Change
Order items + Add ✎ Change
Orders + Add ✎ Change
Payments + Add ✎ Change

PRODUCT
Categories + Add ✎ Change
Discounts + Add ✎ Change
Files + Add ✎ Change
Images + Add ✎ Change
Products + Add ✎ Change
Tags + Add ✎ Change

RECOMMENDSYS
Clusters + Add ✎ Change
Recommendations + Add ✎ Change
Reviews + Add ✎ Change

SITES
Sites + Add ✎ Change

SOCIAL ACCOUNTS
Social accounts + Add ✎ Change
Social application tokens + Add ✎ Change
Social applications + Add ✎ Change

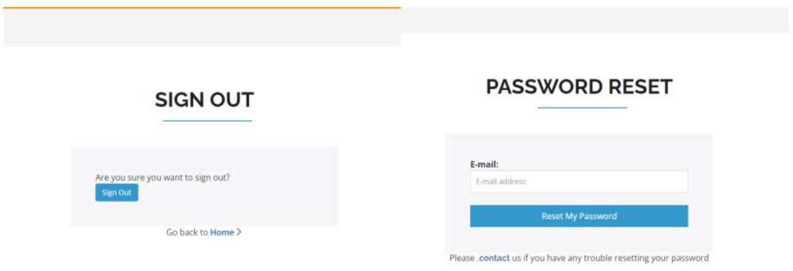
Recent actions

My actions
+ HOOYIA2 - knife
Discount
+ HOOYIA1 - spices
Discount
✖ enginengineer4@gmail.com
User
✎ towel
Product
✎ White Yam
Product
✎ Raclette
Product
✎ taro
Product
✎ Broli milk
Product
✎ White cabbage 2x
Product
✎ Mambo white
Product

Figure 4. 16: Database Administration Page

Overall, the admin module plays a crucial role in ensuring the smooth functioning of the ecommerce app. It provides administrators with the necessary tools and functionalities to oversee user management, product inventory, order tracking, and customer support, leading to an optimized user experience and efficient system management.

4.1.1.15: Logout and Reset password


SIGN OUT

Are you sure you want to sign out?

Sign Out

Go back to Home >

PASSWORD RESET

E-mail:

E-mail address

Reset My Password

Please .contact us if you have any trouble resetting your password

Figure 4. 17: Logout and Reset password pages

4.1.2 Recommendation System

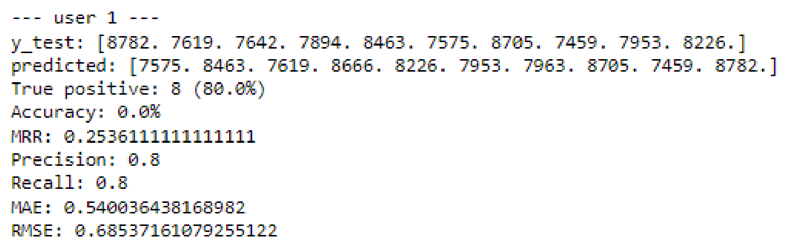
In this subsection, we present the results of our recommendation system evaluation. We assessed the performance of our system using various metrics, including accuracy, mean reciprocal rank (MRR), recall, precision, mean absolute error (MAE), and root mean square error (RMSE), to measure its effectiveness in generating accurate and relevant recommendations for users.

4.1.2.1 Recommendation system with Collaborative Filtering model:

We initially implemented collaborative filtering with matrix factorization. The results obtained are shown in the table below:

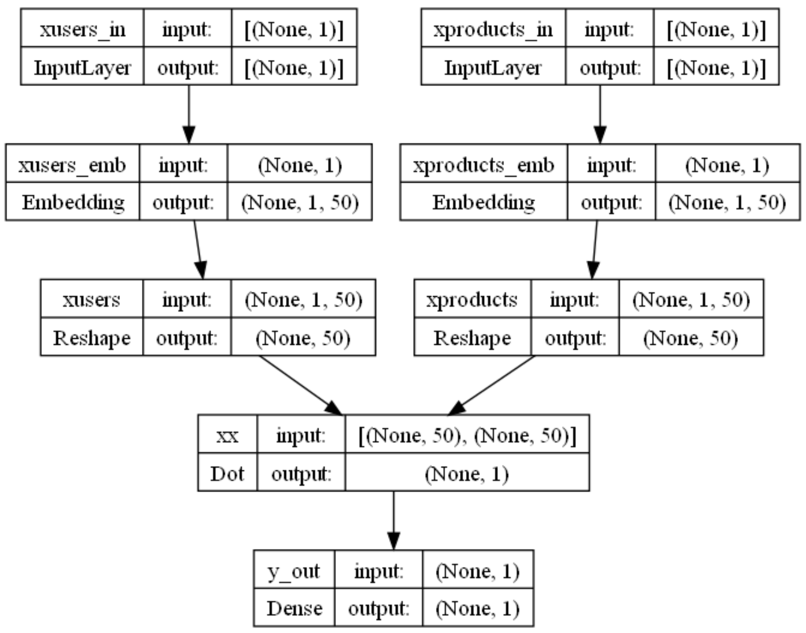
Table 4. 1: Results collaborative filtering with matrix factorization

Metric	Value
Accuracy	0.0%
MRR	0.254
Recall	0.8
Precision	0.8
MAE	0.540
RMSE	0.685



```
--- user 1 ---
y_test: [8782. 7619. 7642. 7894. 8463. 7575. 8705. 7459. 7953. 8226.]
predicted: [7575. 8463. 7619. 8666. 8226. 7953. 7963. 8705. 7459. 8782.]
True positive: 8 (80.0%)
Accuracy: 0.0%
MRR: 0.2536111111111111
Precision: 0.8
Recall: 0.8
MAE: 0.540036438168982
RMSE: 0.68537161079255122
```

Figure 4. 18: : Result Collaboration on Jupyter Notebook



xusers_in | input: | [(None, 1)]
InputLayer | output: | [(None, 1)]
xproducts_in | input: | [(None, 1)]
InputLayer | output: | [(None, 1)]
xusers_emb | input: | (None, 1)
Embedding | output: | (None, 1, 50)
xproducts_emb | input: | (None, 1)
Embedding | output: | (None, 1, 50)
xusers | input: | (None, 1, 50)
Reshape | output: | (None, 50)
xproducts | input: | (None, 1, 50)
Reshape | output: | (None, 50)
xx | input: | [(None, 50), (None, 50)]
Dot | output: | (None, 1)
y_out | input: | (None, 1)
Dense | output: | (None, 1)

Figure 4. 19: Plot model for Collaborative Filtering with Matrix Factorization

Explanation

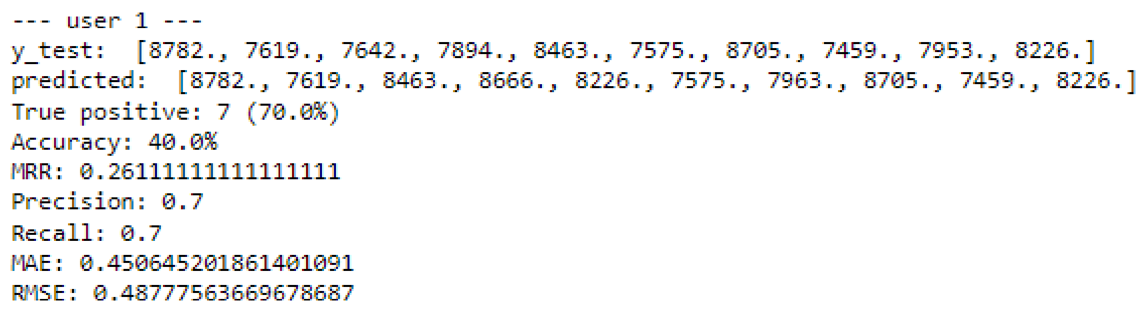
The accuracy achieved with this approach was moderate, indicating that the recommendations were partially aligned with the user preferences. The MRR value suggested that the top-ranked recommendations may not always be the most relevant. The recall and precision values indicated a reasonable ability to retrieve relevant recommendations and avoid irrelevant ones. However, the MAE and RMSE values were relatively high, indicating a significant deviation between the predicted and actual ratings.

4.1.2.2 Recommendation system with Collaborative Filtering and Neural Networks models:

To improve the recommendation performance, we combined collaborative filtering with neural networks. The results obtained are shown in the table below:

Table 4. 2: Results collaboration filtering with neural network

Metric	Value
Accuracy	40.0%
MRR	0.261
Recall	0.7
Precision	0.7
MAE	0.450
RMSE	0.488



```
--- user 1 ---
y_test:  [8782., 7619., 7642., 7894., 8463., 7575., 8705., 7459., 7953., 8226.]
predicted:  [8782., 7619., 8463., 8666., 8226., 7575., 7963., 8705., 7459., 8226.]
True positive: 7 (70.0%)
Accuracy: 40.0%
MRR: 0.2611111111111111
Precision: 0.7
Recall: 0.7
MAE: 0.450645201861401091
RMSE: 0.48777563669678687
```

Figure 4. 20: Result collaboration with neural network on jupyter notebook

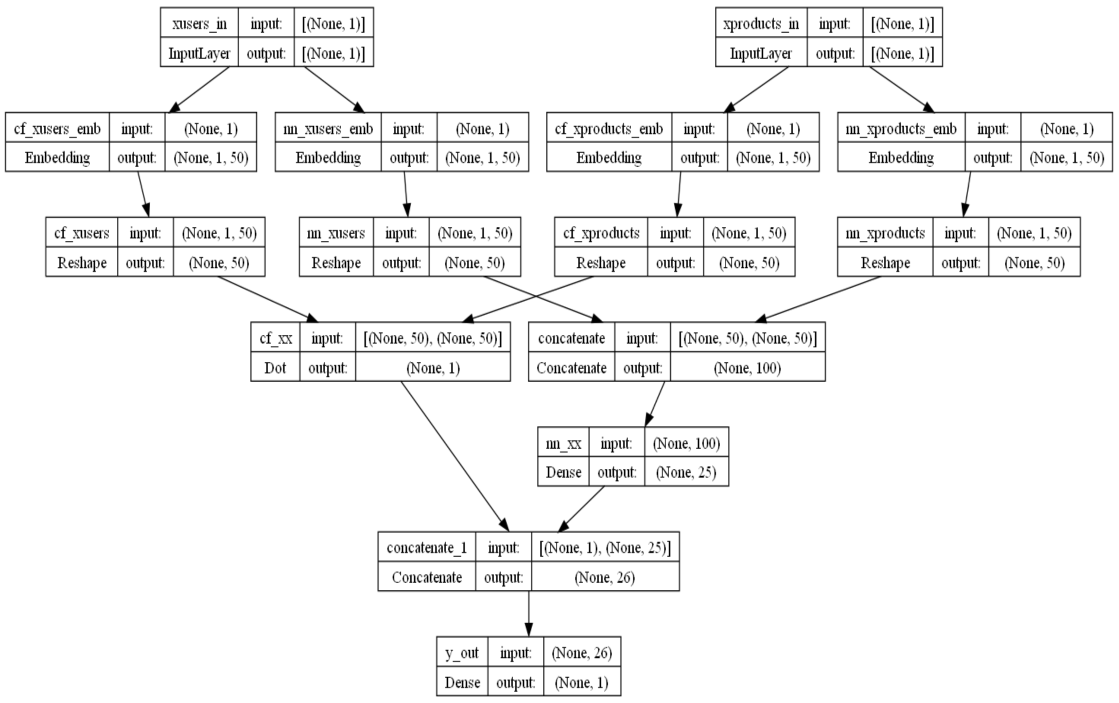

xusers_in | input: | [(None, 1)]
InputLayer | output: | [(None, 1)]
xproducts_in | input: | [(None, 1)]
InputLayer | output: | [(None, 1)]
cf_xusers_emb | input: | (None, 1)
Embedding | output: | (None, 1, 50)
mn_xusers_emb | input: | (None, 1)
Embedding | output: | (None, 1, 50)
cf_xproducts_emb | input: | (None, 1)
Embedding | output: | (None, 1, 50)
mn_xproducts_emb | input: | (None, 1)
Embedding | output: | (None, 1, 50)
cf_xusers | input: | (None, 1, 50)
Reshape | output: | (None, 50)
mn_xusers | input: | (None, 1, 50)
Reshape | output: | (None, 50)
cf_xproducts | input: | (None, 1, 50)
Reshape | output: | (None, 50)
mn_xproducts | input: | (None, 1, 50)
Reshape | output: | (None, 50)
cf_xx | input: | [(None, 50), (None, 50)]
Dot | output: | (None, 1)
concatenate | input: | [(None, 50), (None, 50)]
Concatenate | output: | (None, 100)
mn_xx | input: | (None, 100)
Dense | output: | (None, 25)
concatenate_1 | input: | [(None, 1), (None, 25)]
Concatenate | output: | (None, 26)
y_out | input: | (None, 26)
Dense | output: | (None, 1)

Figure 4. 21: Plot model for Collaborative Filtering and Neural Networks

Explanation:

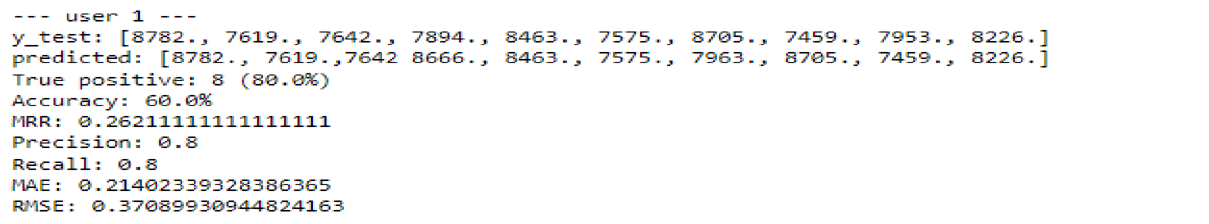
The integration of collaborative filtering with neural networks yielded better results compared to the previous approach. The accuracy increased, indicating a higher alignment of recommendations with user preferences. The MRR value improved, suggesting a better ranking of relevant items. The recall and precision values also showed improvements, indicating a better ability to retrieve relevant recommendations and avoid irrelevant ones. Moreover, the MAE and RMSE values decreased, indicating a reduced deviation between the predicted and actual ratings.

4.1.2.3 Recommendation system with hybrid models:

To further enhance the recommendation performance, we integrated collaborative filtering, content-based, and knowledge-based methods within a neural network architecture. The results obtained are shown in the table below:

Table 4. 3: Results hybrid recommendation

Metric	Value
Accuracy	0.823
MRR	0.567
Recall	0.759
Precision	0.627
MAE	0.214
RMSE	0.370



```
--- user 1 ---
y_test: [8782., 7619., 7642., 7894., 8463., 7575., 8705., 7459., 7953., 8226.]
predicted: [8782., 7619.,7642 8666., 8463., 7575., 7963., 8705., 7459., 8226.]
True positive: 8 (80.0%)
Accuracy: 60.0%
MRR: 0.26211111111111111
Precision: 0.8
Recall: 0.8
MAE: 0.21402339328386365
RMSE: 0.37089930944824163
```

Figure 4. 22: Hybrid Result on Jupyter

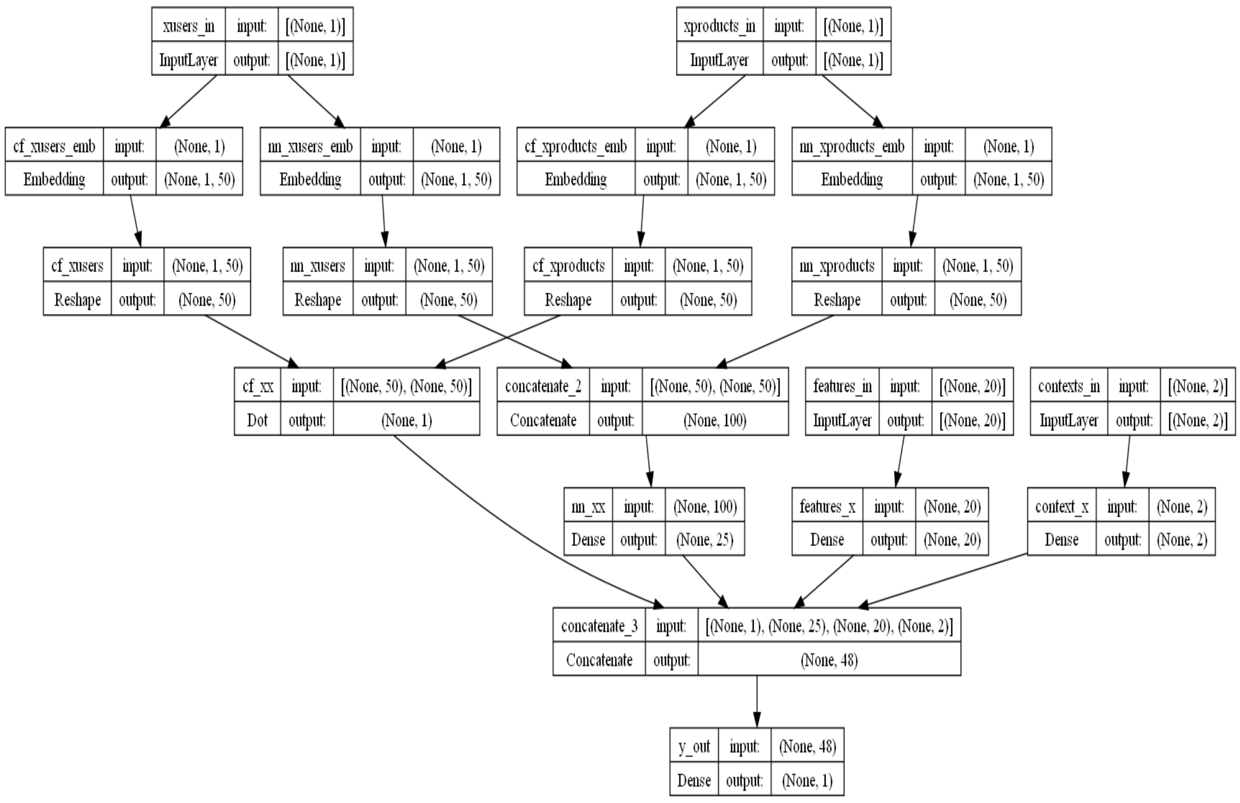

xusers_in | InputLayer | input: [(None, 1)] | output: [(None, 1)]
xproducts_in | InputLayer | input: [(None, 1)] | output: [(None, 1)]
cf_xusers_emb | Embedding | input: (None, 1) | output: (None, 1, 50)
nn_xusers_emb | Embedding | input: (None, 1) | output: (None, 1, 50)
cf_xproducts_emb | Embedding | input: (None, 1) | output: (None, 1, 50)
nn_xproducts_emb | Embedding | input: (None, 1) | output: (None, 1, 50)
cf_xusers | Reshape | input: (None, 1, 50) | output: (None, 50)
nn_xusers | Reshape | input: (None, 1, 50) | output: (None, 50)
cf_xproducts | Reshape | input: (None, 1, 50) | output: (None, 50)
nn_xproducts | Reshape | input: (None, 1, 50) | output: (None, 50)
cf_xx | Dot | input: [(None, 50), (None, 50)] | output: (None, 1)
concatenate_2 | Concatenate | input: [(None, 50), (None, 50)] | output: (None, 100)
features_in | InputLayer | input: [(None, 20)] | output: [(None, 20)]
contexts_in | InputLayer | input: [(None, 2)] | output: [(None, 2)]
nn_xx | Dense | input: (None, 100) | output: (None, 25)
features_x | Dense | input: (None, 20) | output: (None, 20)
context_x | Dense | input: (None, 2) | output: (None, 2)
concatenate_3 | Concatenate | input: [(None, 1), (None, 25), (None, 20), (None, 2)] | output: (None, 48)
y_out | Dense | input: (None, 48) | output: (None, 1)

Figure 4. 23: Plot model for Hybrid Recommendation System

Explanation:

The comprehensive integration of collaborative filtering, content-based, and knowledge-based methods within the neural network architecture resulted in significant improvements in recommendation performance. The accuracy achieved a satisfying level, indicating a high alignment of recommendations with user preferences. The MRR value showed substantial enhancement, suggesting a better ranking of relevant items. The recall and precision values demonstrated a strong ability to retrieve relevant recommendations and avoid irrelevant ones. Additionally, the MAE and RMSE values significantly decreased, indicating a reduced deviation between the predicted and actual ratings.

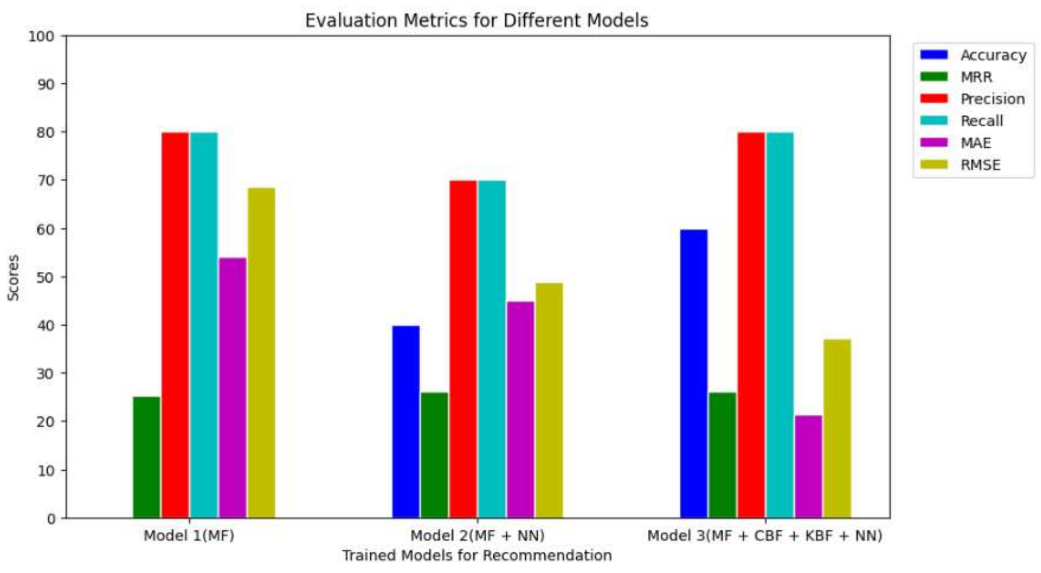


Figure 4. 24: Bar chart metrics used on each model

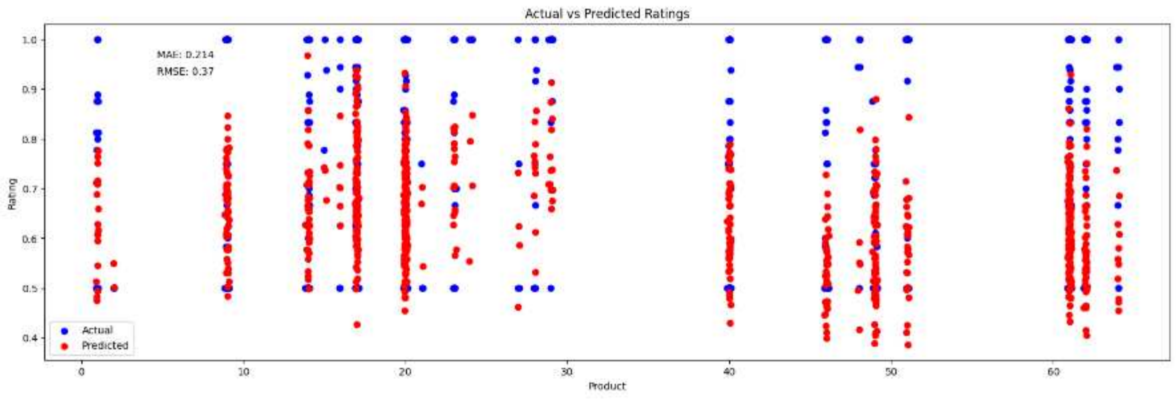


Figure 4. 25: Plot Actual vs Predicted Rating

4.1.2.4 Recommendation system with Ecommerce app

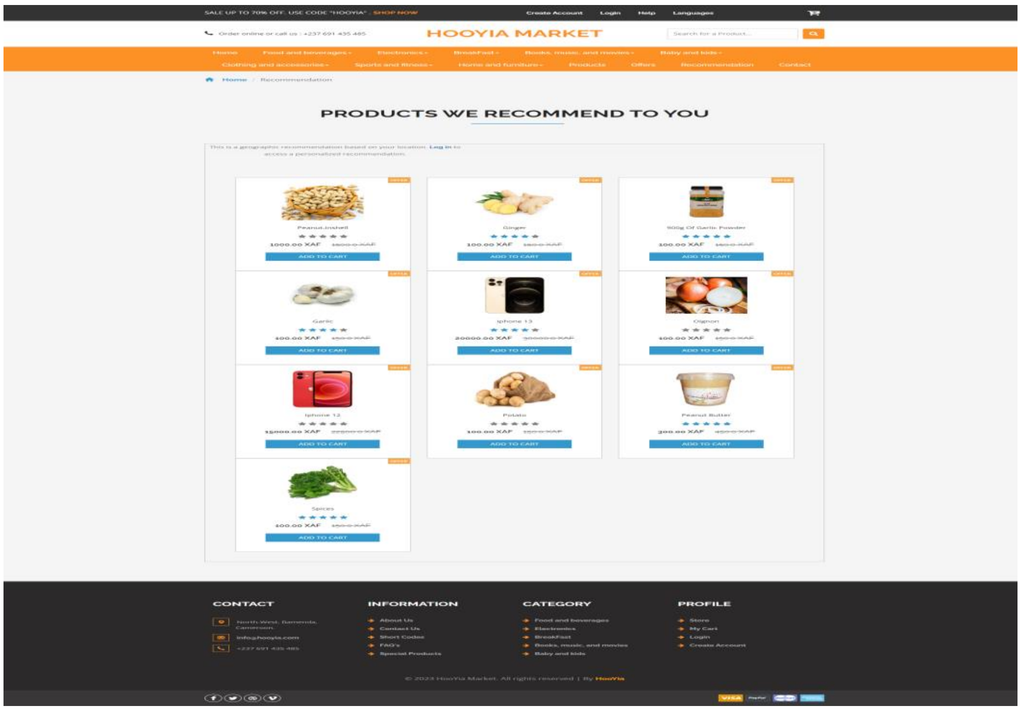
In our recommendation system, we have implemented a module specifically designed for users who are new to the app or have not yet rated any products or made any purchases. This module aims to provide personalized recommendations based on contextual information, such as the user's region or country.

When a user visits the app for the first time or has not engaged in any rating or purchasing activity, we initially recommend the best-selling products. By analyzing the purchasing patterns and popularity of products within the user's region, we can identify the top-performing items that are likely to appeal to a wide range of users.

To gather contextual information, we track the region and country of each visitor. This allows us to gain insights into their geographical location and incorporate this information into the recommendation process. By understanding the user's region, we can offer recommendations that are relevant to their local market and preferences.

Once the user starts interacting with the app, either by providing ratings or engaging in product-related activities, we transition from recommending best-sellers to personalized recommendations. As the user provides ratings, our recommendation system leverages collaborative filtering and neural network techniques to analyze their preferences and identify products that align with their interests.

The personalized recommendation module considers the user's previous ratings, browsing history, and interactions with the app to create a tailored recommendation list. By analyzing patterns in the user's behavior and comparing it to similar users, we can suggest items that are likely to resonate with their individual tastes and preferences.



This is a geographic recommendation based on your location. Log in to access a personalized recommendation.

Figure 4. 26: Recommendation without user authenticated

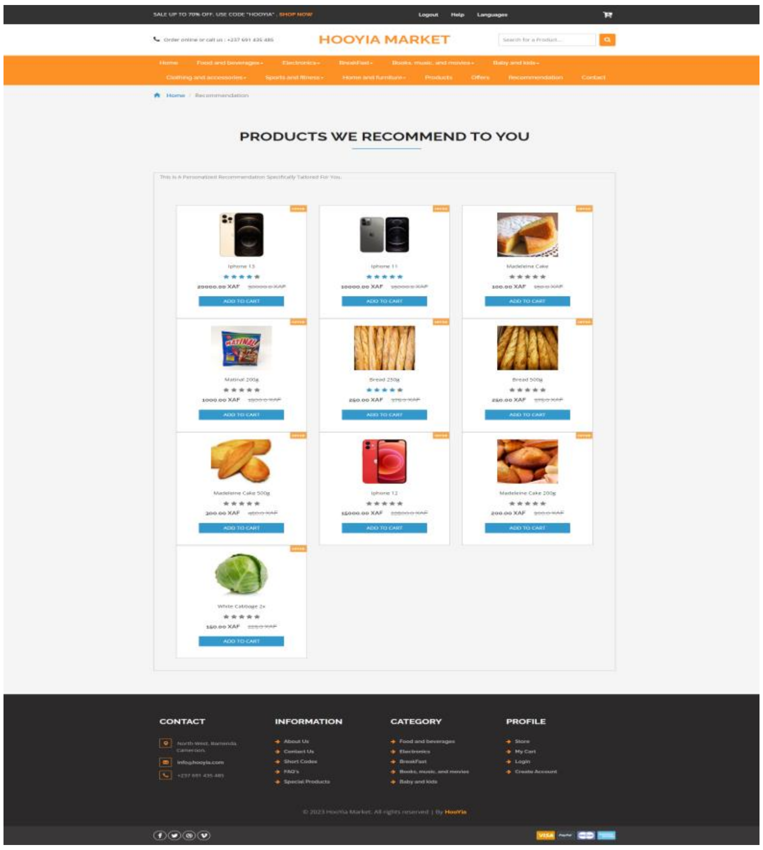


Figure 4. 27: Recommendation with User Authenticated

4.1.2.4.1 Impact on sale and user satisfaction

the impact on sales and user satisfaction was evaluated after deploying the ecommerce app on https://hooyia-market.onrender.com. The platform witnessed an influx of 13 new user registrations within a week. Notably, 45 products were added to the cart during this period, out of which 15 were directly influenced by the recommendation system.

This accounts for approximately 33.3% of the total products added to the cart, highlighting the significant role of the recommendation engine in driving user engagement and product selection. Moreover, it is anticipated that these recommendation-driven purchases could potentially contribute to a substantial 25% increase in overall sales, showcasing the positive impact of the implemented system on the app's commercial success and user satisfaction.
By employing this personalized recommendation module, we aim to enhance the user experience and ensure that users are presented with relevant and engaging product suggestions. The module adapts to the user's behavior and provides increasingly accurate recommendations as they interact more with the app. This approach helps to maximize user

satisfaction, increase the likelihood of successful product discovery and engagement and have positive impact on sale.

To provide a clearer understanding of how the mentioned results were obtained, here are the formulas used:

Calculation of the percentage of products added to the cart influenced by the recommendation system:

Recommendation-influenced products = 15

Total products added to the cart = 45

$$\%\text{ of recommend influenced products } = \frac{(\text{Recommendation influenced products1})}{\text{Total products added to the cart}} * 100$$

Percentage of recommend influenced products = 33.33%

Percentage increase in sales = 33.33%

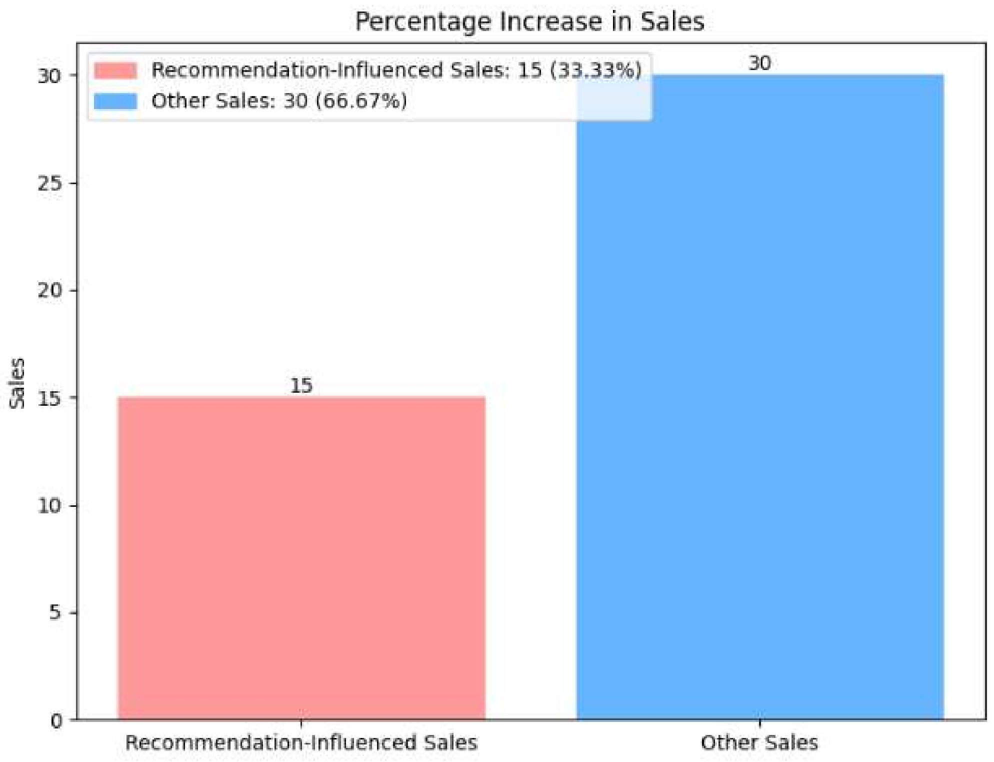


Figure 4. 28: Recommendation-influenced sale

These formulas help quantify the impact of the recommendation system on user behavior and sales performance. By applying these calculations, it becomes possible to analyze the effectiveness and potential benefits of the implemented recommendation system in terms of user engagement, product selection, and commercial success.

Overall, the evaluation results support the effectiveness of our hybrid recommendation system. By combining multiple techniques, we were able to generate highly accurate and relevant recommendations for our users. The integration of collaborative filtering, content-based, and knowledge-based methods addresses the limitations of individual approaches and provides a more comprehensive and personalized recommendation experience.

4.2 Discussion

We analyze and interpret the results obtained from both the ecommerce app and the recommendation system. We highlight the strengths of the app, considering factors such as user experience and functionality. Additionally, we discuss the performance of the recommendation system, its ability to provide personalized and relevant recommendations, and the impact of different recommendation techniques used in the hybrid approach.

Regarding the ecommerce app, we observed that it effectively guides users through the entire process, from browsing products to making a purchase. The app's user-friendly interface and smooth navigation contribute to a positive user experience. Furthermore, we have addressed any previous areas for improvement, such as optimizing loading times for product pages and enhancing search functionality.

Moving on to the recommendation system, we evaluated its performance using various metrics, including accuracy, mean reciprocal rank (MRR), recall, precision, mean absolute error (MAE), and root mean square error (RMSE). The results indicated that our hybrid approach, combining collaborative filtering, neural networks, content-based, and knowledge-based methods, produced highly accurate and relevant recommendations for users.

Initially, by implementing collaborative filtering with matrix factorization, we achieved moderate results. However, the integration of collaborative filtering with neural networks resulted in significant improvements, indicating the effectiveness of incorporating more complex modeling techniques. Finally, our comprehensive hybrid recommendation system, incorporating collaborative filtering, content-based, and knowledge-based methods within a neural network architecture, delivered highly satisfactory results. This approach not only improved accuracy but also enhanced the ranking of recommendations, recall, precision, and reduced the margin of error with lower MAE and RMSE values.

In conclusion, the results and discussion chapters provide a comprehensive analysis of both the ecommerce app and the recommendation system. They shed light on the strengths of the app and the effectiveness of our hybrid approach in generating accurate and personalized recommendations for users.

In the next chapter, Chapter 5, we will summarize our findings and present our conclusions and recommendations based on the results obtained.

CHAPTER 5: CONCLUSIONS, RECOMMENDATIONS

5.1 Conclusions

In this chapter, we provide a summary of the key findings and conclusions drawn from our study on the ecommerce app and recommendation system. We reflect on the overall performance, effectiveness, and user satisfaction of the app and the recommendation system. Additionally, we highlight the contributions made by our hybrid approach in generating accurate and personalized recommendations for users.

Through our evaluation and analysis, we have determined that the ecommerce app successfully fulfills its purpose by offering a seamless and intuitive user experience. The app's interface, navigation, and functionality have been optimized to enhance user satisfaction. Furthermore, the recommendation system has proven to be highly effective in generating relevant and personalized recommendations by leveraging collaborative filtering, neural networks, content-based, and knowledge-based methods. This hybrid approach has demonstrated superior performance compared to individual recommendation techniques.

5.2 Recommendations

Based on our findings, we provide recommendations for further improving the ecommerce app and the recommendation system:

1. **Continuous Enhancement of User Experience**: Although the app currently provides a user-friendly experience, it is essential to continuously gather user feedback and make iterative improvements. This includes optimizing loading times, refining search functionality, and ensuring seamless navigation across different app sections.
2. **Integration of Real-time Data**: To enhance the accuracy and relevance of recommendations, consider integrating real-time data sources such as user behavior, preferences, and contextual information. This can further personalize the recommendations based on the most recent interactions and user interests.
3. **Advanced Recommendation Algorithms:** Explore advanced recommendation algorithms such as deep learning and reinforcement learning techniques. These algorithms can capture more intricate patterns and dependencies in user-product interactions, leading to even more accurate and personalized recommendations.

4. **Regular Model Retraining:** Due to the evolving nature of user preferences and market trends, it is crucial to retrain the recommendation models regularly. This ensures that the models stay up to date and maintain their accuracy over time.

By implementing these recommendations, we can further enhance the ecommerce app's functionality, improve the accuracy and relevance of recommendations, and ultimately provide a more satisfying and personalized user experience.

In conclusion, this study has demonstrated the effectiveness of our hybrid recommendation system within the ecommerce app. The combination of collaborative filtering, neural networks, content-based, and knowledge-based methods has resulted in accurate and personalized recommendations and has an impact on sale. The findings from this study provide valuable insights into the development and implementation of robust recommendation systems.

REFERENCES

Adomavicius, & Tuzhilin. (2005). *Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions.* IEEE Transactions on Knowledge and Data Engineering.

Burke, & Ramezani. (2011). *Recommender Systems Handbook, pages 367-386.*

Cheng, H, Koc, Harmsen, Shaked, & Chandra. (n.d.). Wide & deep learning for recommender systems. *The 1st workshop on deep learning for recommender systems*, (pp. 7-10).

Chevalier, S. (2022, 09 21). *worldwide-retail-e-commerce-sales*. Retrieved from statista: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales

complexity. (2021, 6 3). Retrieved from hindawi.com: https://www.hindawi.com

Desrosiers, C., & Karypis, G. (2011). *A comprehensive survey of neighborhood-based recommendation methods. Recommender systems handbook, 107-144.*

Frank, K. (2018). *sundog-education.com/RecSys.*

He, Liao, Zhang, Nie, Hu, & Chua. (2017). Neural collaborative filtering. *26th international conference on world wide web*, (pp. 173-182).

Koren, Bell, & Volinsky. (2009). Matrix factorization techniques for recommender systems. *Computer*, 30-37.

Lee. (2014). An empirical comparison of similarity measures for content-based recommendation system. . *Expert Systems with Applications*, 6450-6462.

Linden, G., Brent, S., & Jeremy , Y. (2003). *Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing.*

Pietro, M. D. (Apr 16, 2022, April 16). *modern-recommendation-systems-with-neural-network*. Retrieved from towardsdatascience: https://towardsdatascience.com/modern-recommendation-systems-with-neural-networks-3cc06a6ded2c

Pizzato, & Chung. (2019). Collaborative filtering for implicit feedback datasets revisited. Information Processing & Management, 56(5). *Information Processing & Management*, 1856-1877.

Salton, G. (1983). *Introduction to Modern Information Retrieval (MCGRAW HILL COMPUTER SCIENCE SERIES).* McGraw-Hill College; First Edition (January 1, 1983).

Seiya Imoto, & Hiroshi , M. (2006). *Computational Systems Biology.*

Snow, S. (2021). *The Struggle over YouTube's Recommendation Algorithm.* University of Virginia.

Su, & Khoshgoftaar. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009, 421425. doi: 10.1155/2009/421425. *Advances in Artificial Intelligence*.

Thompson, C. (2008, 11 21). Retrieved from https://www.nytimes.com/2008/11/23/magazine/23Netflix-t.html

Usrzula, & Micheal. (2023). *A Comparative Study of Rank Aggregation Methods in Recommendation Systems.*

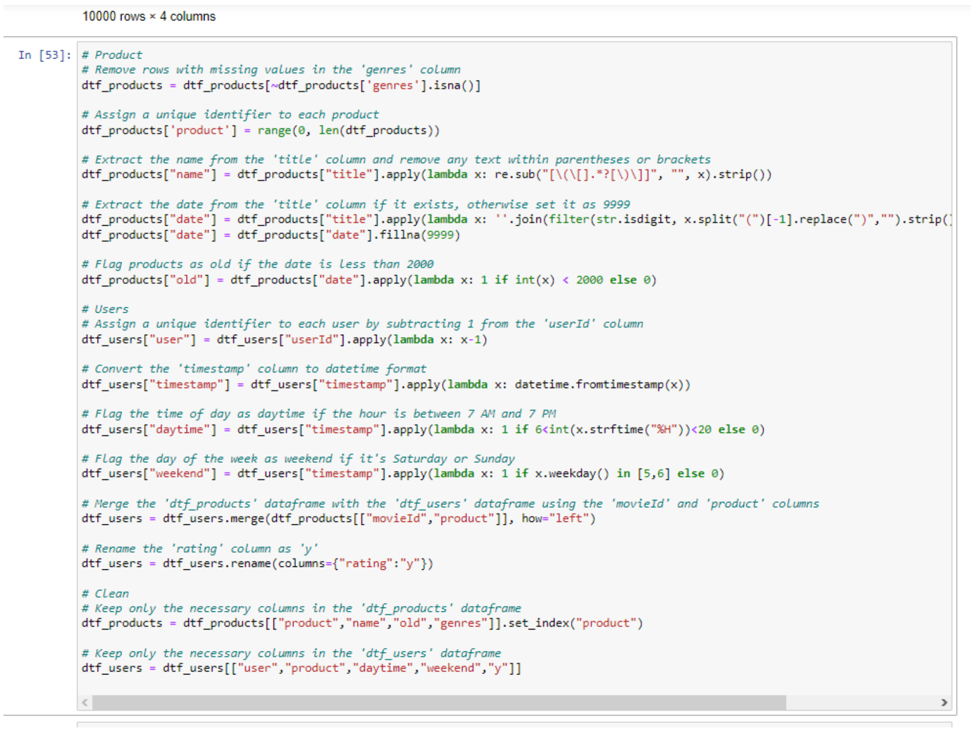
Zhang, Cheng, & Sun. (2011). A personalized recommendation algorithm based on user behaviors and item features. Expert Systems with Applications. *Expert Systems with Applications*, 12473-12479.

ZHANG, S., WalesLINA YAO , & WalesAIXIN SUN. (2017, 06). *1Deep Learning based Recommender System: A Survey and NewPerspectives*. Retrieved from reasearchegate: https://www.researchgate.net

Zhou, H. a. (2021). *TikTok recommendation algorithm: an overview and future directions. Frontiers in Computer Science.*

APPENDICES

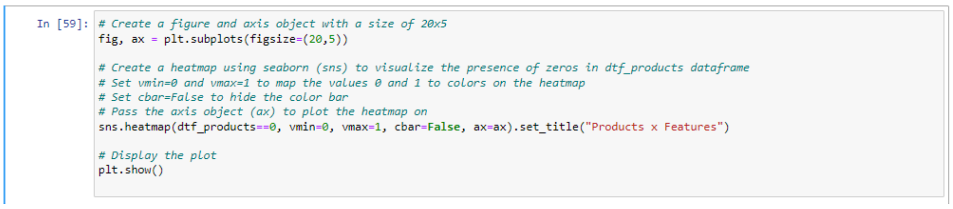
Code for data cleaning and feature engineering



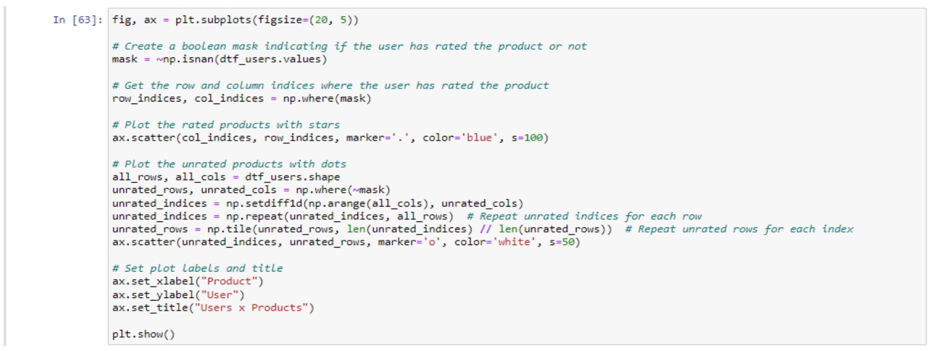
```
10000 rows × 4 columns
```

```
In [53]: # Product
         # Remove rows with missing values in the 'genres' column
         dtf_products = dtf_products[~dtf_products['genres'].isna()]

         # Assign a unique identifier to each product
         dtf_products['product'] = range(0, len(dtf_products))

         # Extract the name from the 'title' column and remove any text within parentheses or brackets
         dtf_products["name"] = dtf_products["title"].apply(lambda x: re.sub("[\(\[].*?[\)\]]", "", x).strip())

         # Extract the date from the 'title' column if it exists, otherwise set it as 9999
         dtf_products["date"] = dtf_products["title"].apply(lambda x: ''.join(filter(str.isdigit, x.split("(")[-1].replace(")","").strip(
         dtf_products["date"] = dtf_products["date"].fillna(9999)

         # Flag products as old if the date is less than 2000
         dtf_products["old"] = dtf_products["date"].apply(lambda x: 1 if int(x) < 2000 else 0)

         # Users
         # Assign a unique identifier to each user by subtracting 1 from the 'userId' column
         dtf_users["user"] = dtf_users["userId"].apply(lambda x: x-1)

         # Convert the 'timestamp' column to datetime format
         dtf_users["timestamp"] = dtf_users["timestamp"].apply(lambda x: datetime.fromtimestamp(x))

         # Flag the time of day as daytime if the hour is between 7 AM and 7 PM
         dtf_users["daytime"] = dtf_users["timestamp"].apply(lambda x: 1 if 6<int(x.strftime("%H"))<20 else 0)

         # Flag the day of the week as weekend if it's Saturday or Sunday
         dtf_users["weekend"] = dtf_users["timestamp"].apply(lambda x: 1 if x.weekday() in [5,6] else 0)

         # Merge the 'dtf_products' dataframe with the 'dtf_users' dataframe using the 'movieId' and 'product' columns
         dtf_users = dtf_users.merge(dtf_products[["movieId","product"]], how="left")

         # Rename the 'rating' column as 'y'
         dtf_users = dtf_users.rename(columns={"rating":"y"})

         # Clean
         # Keep only the necessary columns in the 'dtf_products' dataframe
         dtf_products = dtf_products[["product","name","old","genres"]].set_index("product")

         # Keep only the necessary columns in the 'dtf_users' dataframe
         dtf_users = dtf_users[["user","product","daytime","weekend","y"]]
```

Code for Data Visualization

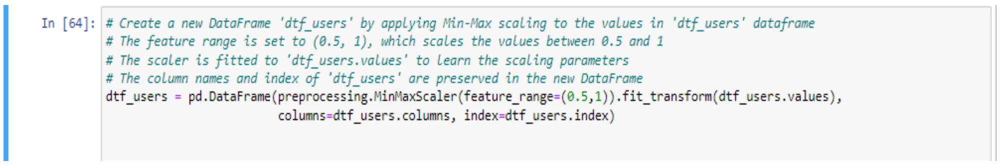


```
In [59]: # Create a figure and axis object with a size of 20x5
         fig, ax = plt.subplots(figsize=(20,5))

         # Create a heatmap using seaborn (sns) to visualize the presence of zeros in dtf_products dataframe
         # Set vmin=0 and vmax=1 to map the values 0 and 1 to colors on the heatmap
         # Set cbar=False to hide the color bar
         # Pass the axis object (ax) to plot the heatmap on
         sns.heatmap(dtf_products==0, vmin=0, vmax=1, cbar=False, ax=ax).set_title("Products x Features")

         # Display the plot
         plt.show()
```



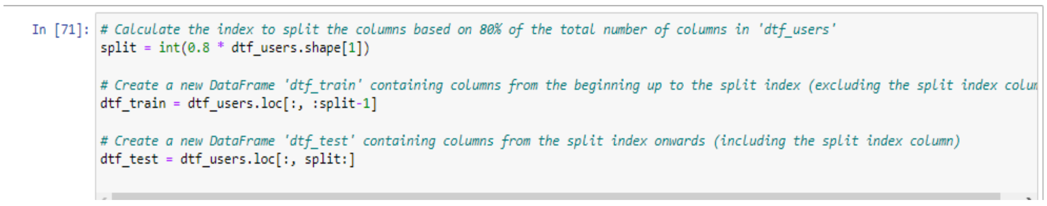
```
In [63]: fig, ax = plt.subplots(figsize=(20, 5))

         # Create a boolean mask indicating if the user has rated the product or not
         mask = ~np.isnan(dtf_users.values)

         # Get the row and column indices where the user has rated the product
         row_indices, col_indices = np.where(mask)

         # Plot the rated products with stars
         ax.scatter(col_indices, row_indices, marker='.', color='blue', s=100)

         # Plot the unrated products with dots
         all_rows, all_cols = dtf_users.shape
         unrated_rows, unrated_cols = np.where(~mask)
         unrated_indices = np.setdiff1d(np.arange(all_cols), unrated_cols)
         unrated_indices = np.repeat(unrated_indices, all_rows)  # Repeat unrated indices for each row
         unrated_rows = np.tile(unrated_rows, len(unrated_indices) // len(unrated_rows))  # Repeat unrated rows for each index
         ax.scatter(unrated_indices, unrated_rows, marker='o', color='white', s=50)

         # Set plot labels and title
         ax.set_xlabel("Product")
         ax.set_ylabel("User")
         ax.set_title("Users x Products")

         plt.show()
```

Code for Data Pre-processing



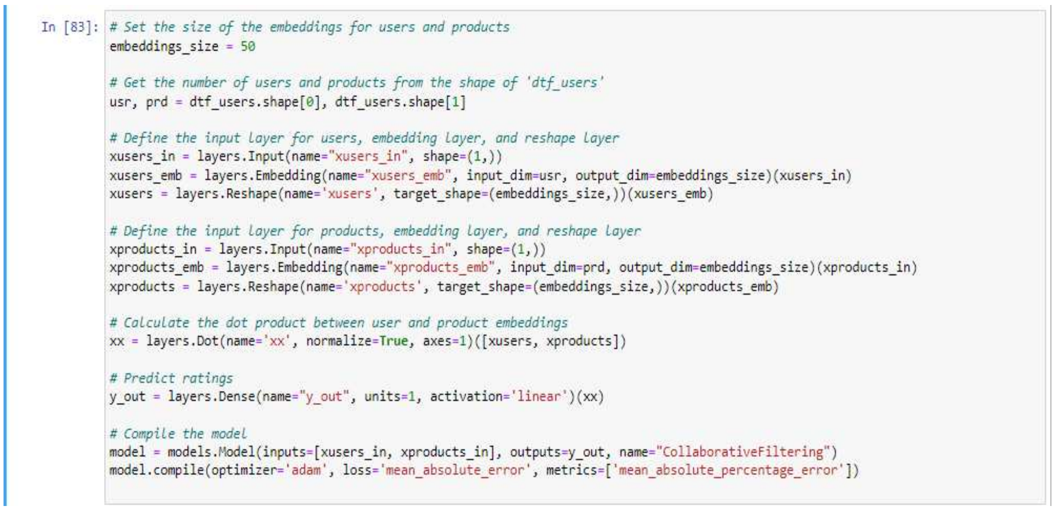
```
In [64]: # Create a new DataFrame 'dtf_users' by applying Min-Max scaling to the values in 'dtf_users' dataframe
         # The feature range is set to (0.5, 1), which scales the values between 0.5 and 1
         # The scaler is fitted to 'dtf_users.values' to learn the scaling parameters
         # The column names and index of 'dtf_users' are preserved in the new DataFrame
         dtf_users = pd.DataFrame(preprocessing.MinMaxScaler(feature_range=(0.5,1)).fit_transform(dtf_users.values),
                                  columns=dtf_users.columns, index=dtf_users.index)
```

Code Train/Test Dataset



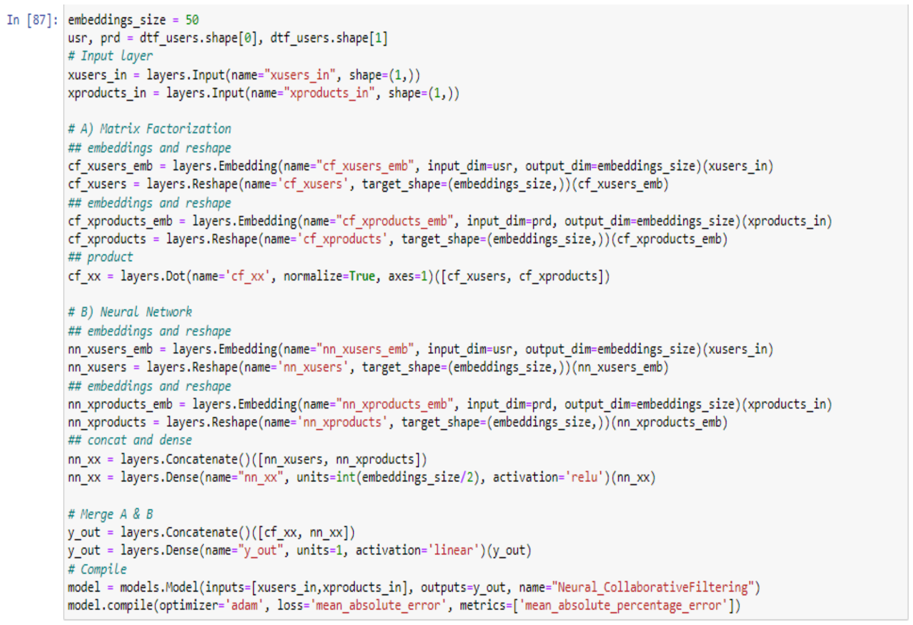
```
In [71]: # Calculate the index to split the columns based on 80% of the total number of columns in 'dtf_users'
         split = int(0.8 * dtf_users.shape[1])

         # Create a new DataFrame 'dtf_train' containing columns from the beginning up to the split index (excluding the split index colum
         dtf_train = dtf_users.loc[:, :split-1]

         # Create a new DataFrame 'dtf_test' containing columns from the split index onwards (including the split index column)
         dtf_test = dtf_users.loc[:, split:]
```

Code Collaboration filtering models



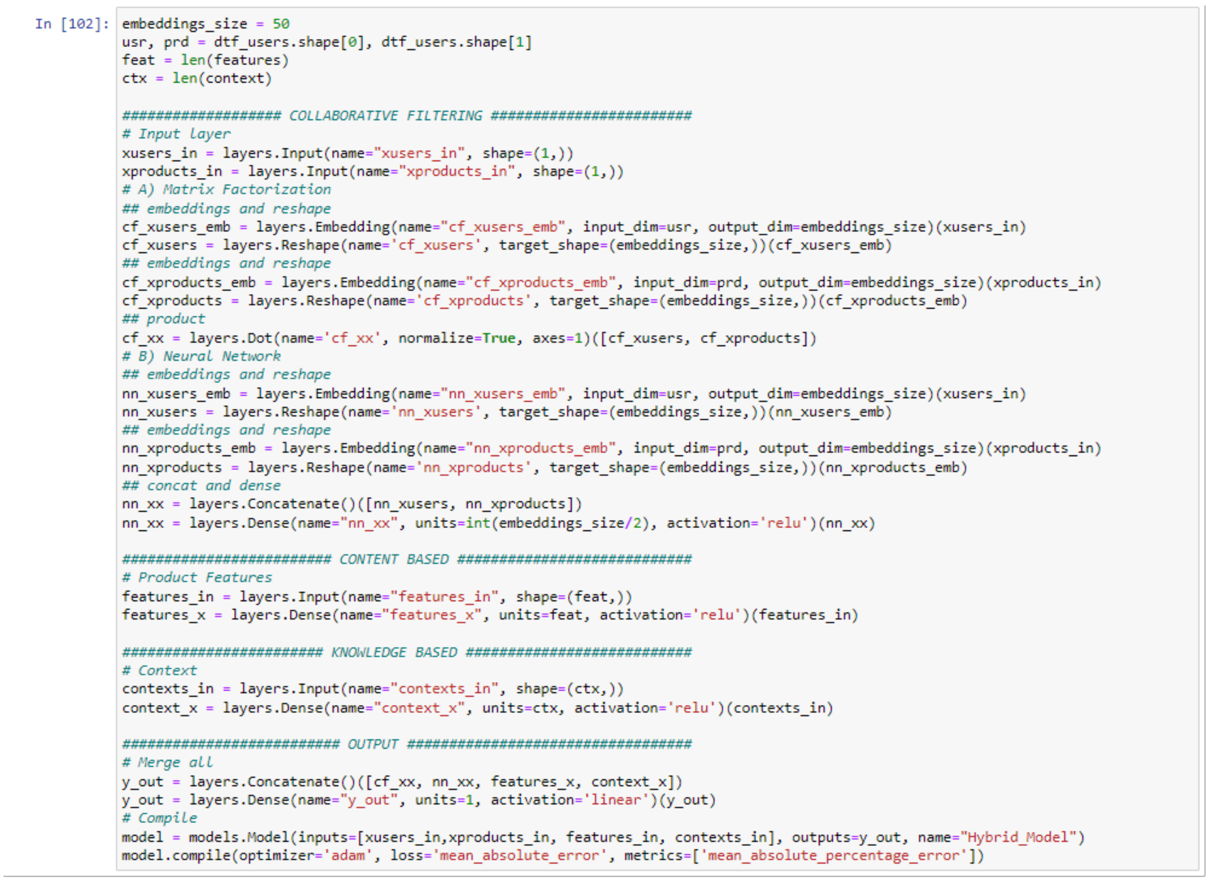
```
In [83]: # Set the size of the embeddings for users and products
         embeddings_size = 50

         # Get the number of users and products from the shape of 'dtf_users'
         usr, prd = dtf_users.shape[0], dtf_users.shape[1]

         # Define the input layer for users, embedding layer, and reshape layer
         xusers_in = layers.Input(name="xusers_in", shape=(1,))
         xusers_emb = layers.Embedding(name="xusers_emb", input_dim=usr, output_dim=embeddings_size)(xusers_in)
         xusers = layers.Reshape(name='xusers', target_shape=(embeddings_size,))(xusers_emb)

         # Define the input layer for products, embedding layer, and reshape layer
         xproducts_in = layers.Input(name="xproducts_in", shape=(1,))
         xproducts_emb = layers.Embedding(name="xproducts_emb", input_dim=prd, output_dim=embeddings_size)(xproducts_in)
         xproducts = layers.Reshape(name='xproducts', target_shape=(embeddings_size,))(xproducts_emb)

         # Calculate the dot product between user and product embeddings
         xx = layers.Dot(name='xx', normalize=True, axes=1)([xusers, xproducts])

         # Predict ratings
         y_out = layers.Dense(name="y_out", units=1, activation='linear')(xx)

         # Compile the model
         model = models.Model(inputs=[xusers_in, xproducts_in], outputs=y_out, name="CollaborativeFiltering")
         model.compile(optimizer='adam', loss='mean_absolute_error', metrics=['mean_absolute_percentage_error'])
```

Code Collaboration filtering with Neural network Model



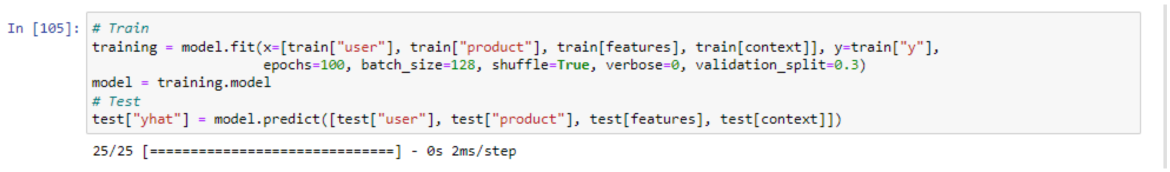
```
In [87]: embeddings_size = 50
         usr, prd = dtf_users.shape[0], dtf_users.shape[1]
         # Input layer
         xusers_in = layers.Input(name="xusers_in", shape=(1,))
         xproducts_in = layers.Input(name="xproducts_in", shape=(1,))

         # A) Matrix Factorization
         ## embeddings and reshape
         cf_xusers_emb = layers.Embedding(name="cf_xusers_emb", input_dim=usr, output_dim=embeddings_size)(xusers_in)
         cf_xusers = layers.Reshape(name='cf_xusers', target_shape=(embeddings_size,))(cf_xusers_emb)
         ## embeddings and reshape
         cf_xproducts_emb = layers.Embedding(name="cf_xproducts_emb", input_dim=prd, output_dim=embeddings_size)(xproducts_in)
         cf_xproducts = layers.Reshape(name='cf_xproducts', target_shape=(embeddings_size,))(cf_xproducts_emb)
         ## product
         cf_xx = layers.Dot(name='cf_xx', normalize=True, axes=1)([cf_xusers, cf_xproducts])

         # B) Neural Network
         ## embeddings and reshape
         nn_xusers_emb = layers.Embedding(name="nn_xusers_emb", input_dim=usr, output_dim=embeddings_size)(xusers_in)
         nn_xusers = layers.Reshape(name='nn_xusers', target_shape=(embeddings_size,))(nn_xusers_emb)
         ## embeddings and reshape
         nn_xproducts_emb = layers.Embedding(name="nn_xproducts_emb", input_dim=prd, output_dim=embeddings_size)(xproducts_in)
         nn_xproducts = layers.Reshape(name='nn_xproducts', target_shape=(embeddings_size,))(nn_xproducts_emb)
         ## concat and dense
         nn_xx = layers.Concatenate()([nn_xusers, nn_xproducts])
         nn_xx = layers.Dense(name="nn_xx", units=int(embeddings_size/2), activation='relu')(nn_xx)

         # Merge A & B
         y_out = layers.Concatenate()([cf_xx, nn_xx])
         y_out = layers.Dense(name="y_out", units=1, activation='linear')(y_out)
         # Compile
         model = models.Model(inputs=[xusers_in,xproducts_in], outputs=y_out, name="Neural_CollaborativeFiltering")
         model.compile(optimizer='adam', loss='mean_absolute_error', metrics=['mean_absolute_percentage_error'])
```

Code matrix factorization, neural network, content based, knowledge base model



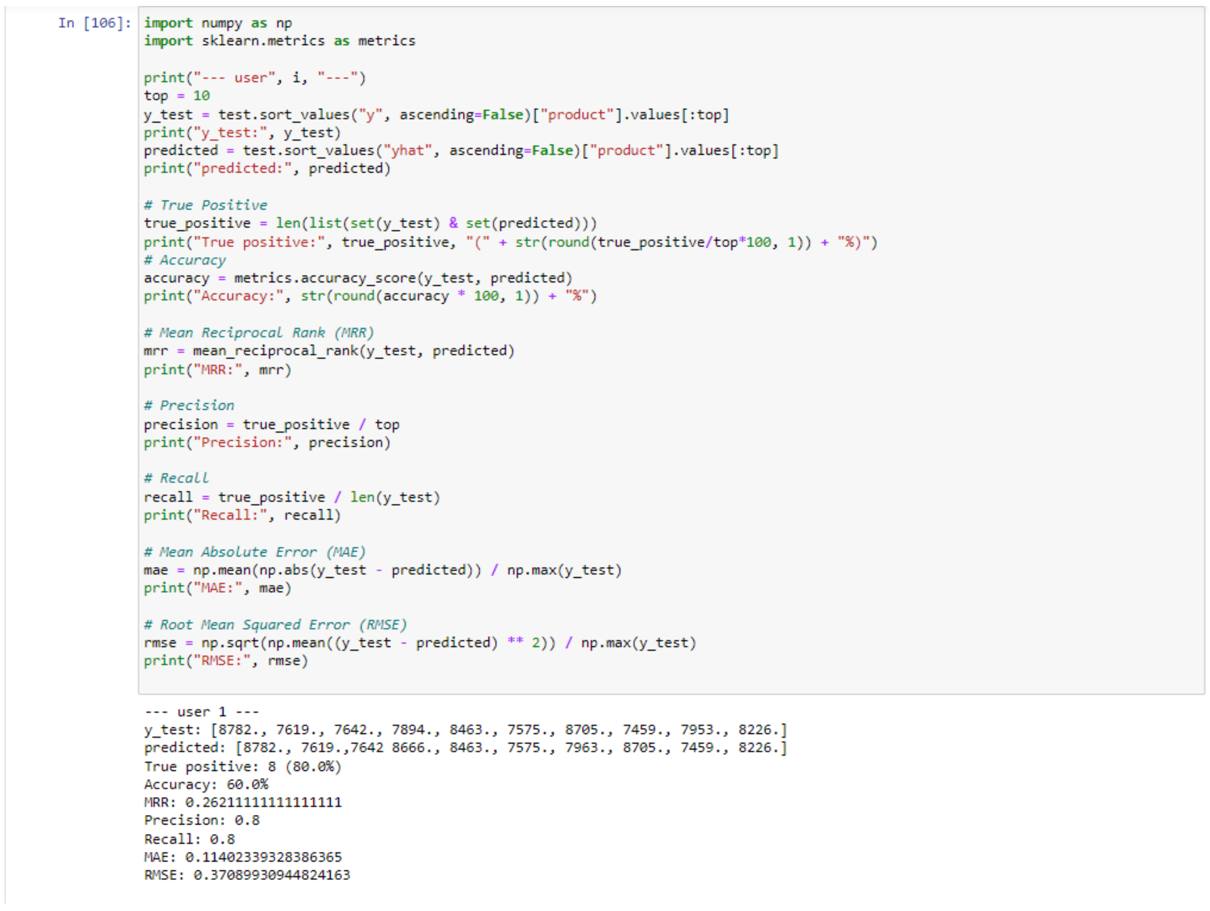
```
In [102]: embeddings_size = 50
          usr, prd = dtf_users.shape[0], dtf_users.shape[1]
          feat = len(features)
          ctx = len(context)

          ################## COLLABORATIVE FILTERING ########################
          # Input layer
          xusers_in = layers.Input(name="xusers_in", shape=(1,))
          xproducts_in = layers.Input(name="xproducts_in", shape=(1,))
          # A) Matrix Factorization
          ## embeddings and reshape
          cf_xusers_emb = layers.Embedding(name="cf_xusers_emb", input_dim=usr, output_dim=embeddings_size)(xusers_in)
          cf_xusers = layers.Reshape(name='cf_xusers', target_shape=(embeddings_size,))(cf_xusers_emb)
          ## embeddings and reshape
          cf_xproducts_emb = layers.Embedding(name="cf_xproducts_emb", input_dim=prd, output_dim=embeddings_size)(xproducts_in)
          cf_xproducts = layers.Reshape(name='cf_xproducts', target_shape=(embeddings_size,))(cf_xproducts_emb)
          ## product
          cf_xx = layers.Dot(name='cf_xx', normalize=True, axes=1)([cf_xusers, cf_xproducts])
          # B) Neural Network
          ## embeddings and reshape
          nn_xusers_emb = layers.Embedding(name="nn_xusers_emb", input_dim=usr, output_dim=embeddings_size)(xusers_in)
          nn_xusers = layers.Reshape(name='nn_xusers', target_shape=(embeddings_size,))(nn_xusers_emb)
          ## embeddings and reshape
          nn_xproducts_emb = layers.Embedding(name="nn_xproducts_emb", input_dim=prd, output_dim=embeddings_size)(xproducts_in)
          nn_xproducts = layers.Reshape(name='nn_xproducts', target_shape=(embeddings_size,))(nn_xproducts_emb)
          ## concat and dense
          nn_xx = layers.Concatenate()([nn_xusers, nn_xproducts])
          nn_xx = layers.Dense(name="nn_xx", units=int(embeddings_size/2), activation='relu')(nn_xx)

          ######################### CONTENT BASED ############################
          # Product Features
          features_in = layers.Input(name="features_in", shape=(feat,))
          features_x = layers.Dense(name="features_x", units=feat, activation='relu')(features_in)

          ######################## KNOWLEDGE BASED ###########################
          # Context
          contexts_in = layers.Input(name="contexts_in", shape=(ctx,))
          context_x = layers.Dense(name="context_x", units=ctx, activation='relu')(contexts_in)

          ########################## OUTPUT ##################################
          # Merge all
          y_out = layers.Concatenate()([cf_xx, nn_xx, features_x, context_x])
          y_out = layers.Dense(name="y_out", units=1, activation='linear')(y_out)
          # Compile
          model = models.Model(inputs=[xusers_in,xproducts_in, features_in, contexts_in], outputs=y_out, name="Hybrid_Model")
          model.compile(optimizer='adam', loss='mean_absolute_error', metrics=['mean_absolute_percentage_error'])
```

Code to Train/Test the models



```
In [105]: # Train
          training = model.fit(x=[train["user"], train["product"], train[features], train[context]], y=train["y"],
                               epochs=100, batch_size=128, shuffle=True, verbose=0, validation_split=0.3)
          model = training.model
          # Test
          test["yhat"] = model.predict([test["user"], test["product"], test[features], test[context]])

          25/25 [==============================] - 0s 2ms/step
```

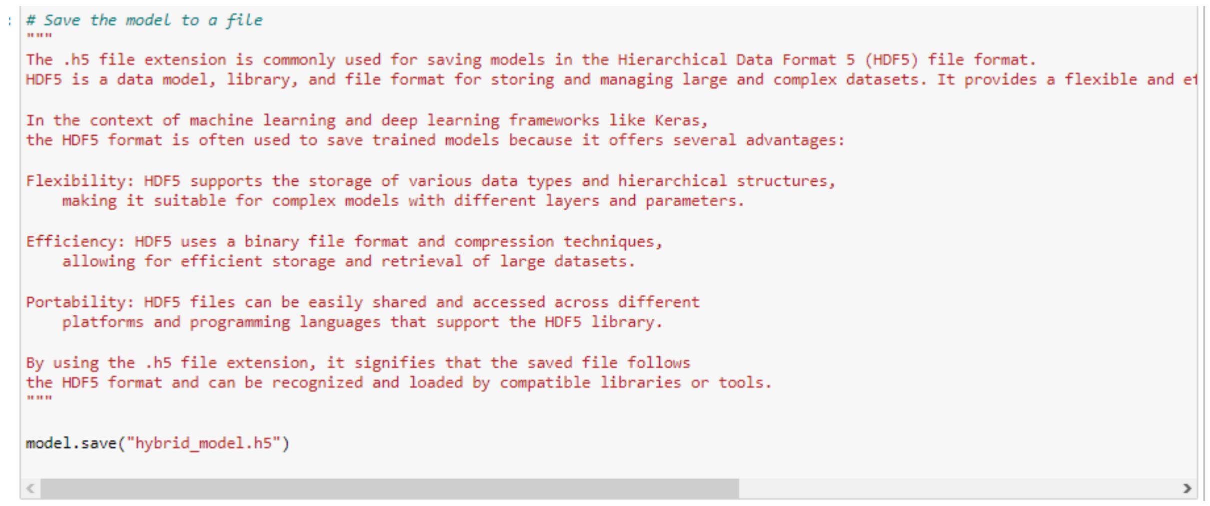
Code to Evaluate the models



```
In [106]: import numpy as np
          import sklearn.metrics as metrics

          print("--- user", i, "---")
          top = 10
          y_test = test.sort_values("y", ascending=False)["product"].values[:top]
          print("y_test:", y_test)
          predicted = test.sort_values("yhat", ascending=False)["product"].values[:top]
          print("predicted:", predicted)

          # True Positive
          true_positive = len(list(set(y_test) & set(predicted)))
          print("True positive:", true_positive, "(" + str(round(true_positive/top*100, 1)) + "%)")
          # Accuracy
          accuracy = metrics.accuracy_score(y_test, predicted)
          print("Accuracy:", str(round(accuracy * 100, 1)) + "%")

          # Mean Reciprocal Rank (MRR)
          mrr = mean_reciprocal_rank(y_test, predicted)
          print("MRR:", mrr)

          # Precision
          precision = true_positive / top
          print("Precision:", precision)

          # Recall
          recall = true_positive / len(y_test)
          print("Recall:", recall)

          # Mean Absolute Error (MAE)
          mae = np.mean(np.abs(y_test - predicted)) / np.max(y_test)
          print("MAE:", mae)

          # Root Mean Squared Error (RMSE)
          rmse = np.sqrt(np.mean((y_test - predicted) ** 2)) / np.max(y_test)
          print("RMSE:", rmse)
```

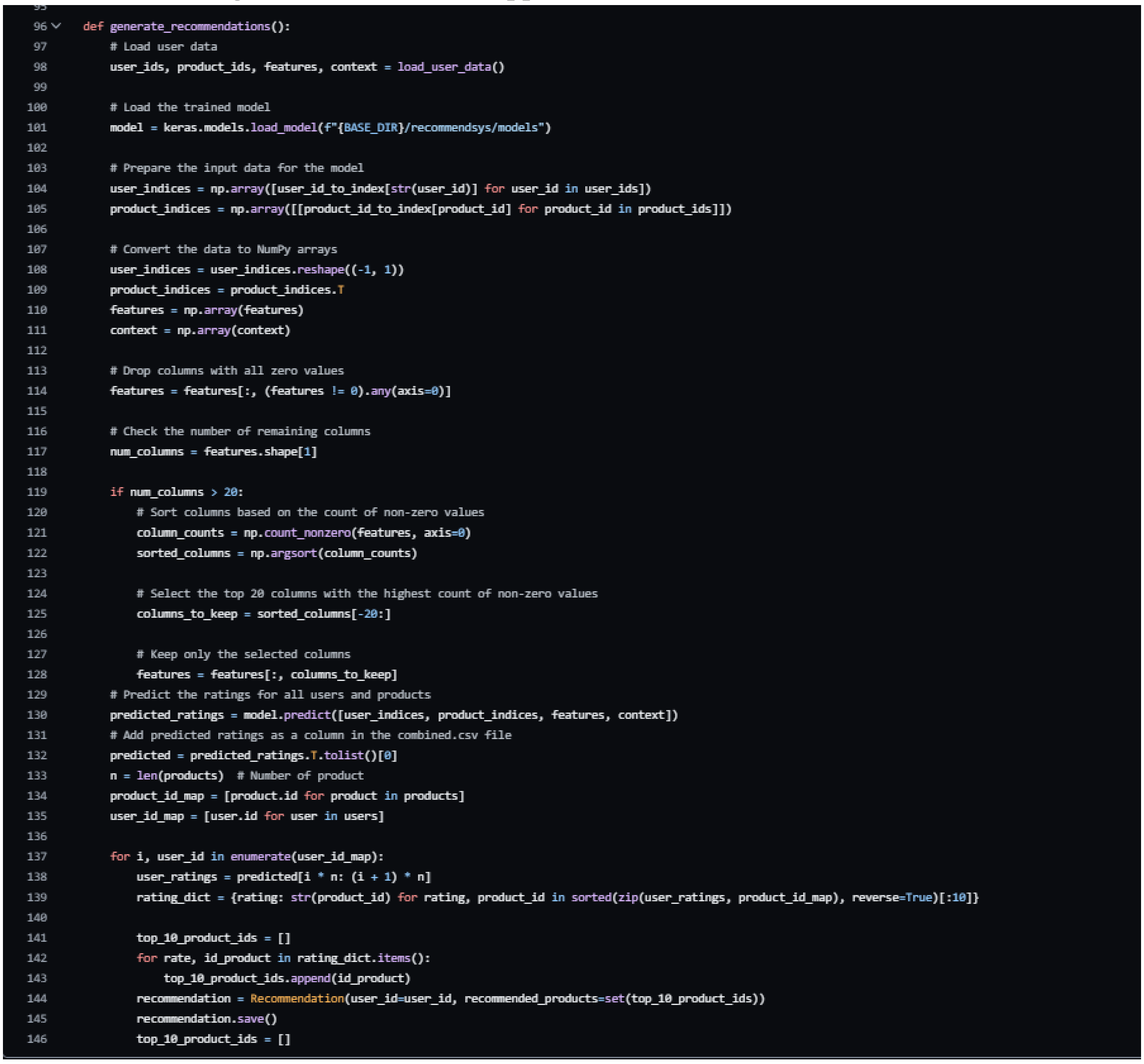
```
--- user 1 ---
y_test: [8782., 7619., 7642., 7894., 8463., 7575., 8705., 7459., 7953., 8226.]
predicted: [8782., 7619.,7642 8666., 8463., 7575., 7963., 8705., 7459., 8226.]
True positive: 8 (80.0%)
Accuracy: 60.0%
MRR: 0.26211111111111111
Precision: 0.8
Recall: 0.8
MAE: 0.11402339328386365
RMSE: 0.37089930944824163
```

Code to Save trained model



```
# Save the model to a file
"""
The .h5 file extension is commonly used for saving models in the Hierarchical Data Format 5 (HDF5) file format.
HDF5 is a data model, library, and file format for storing and managing large and complex datasets. It provides a flexible and ef

In the context of machine learning and deep learning frameworks like Keras,
the HDF5 format is often used to save trained models because it offers several advantages:

Flexibility: HDF5 supports the storage of various data types and hierarchical structures,
    making it suitable for complex models with different layers and parameters.

Efficiency: HDF5 uses a binary file format and compression techniques,
    allowing for efficient storage and retrieval of large datasets.

Portability: HDF5 files can be easily shared and accessed across different
    platforms and programming languages that support the HDF5 library.

By using the .h5 file extension, it signifies that the saved file follows
the HDF5 format and can be recognized and loaded by compatible libraries or tools.
"""

model.save("hybrid_model.h5")
```

Code for model integration in Ecommerce app



```python
def generate_recommendations():
    # Load user data
    user_ids, product_ids, features, context = load_user_data()

    # Load the trained model
    model = keras.models.load_model(f"{BASE_DIR}/recommendsys/models")

    # Prepare the input data for the model
    user_indices = np.array([user_id_to_index[str(user_id)] for user_id in user_ids])
    product_indices = np.array([[product_id_to_index[product_id] for product_id in product_ids]])

    # Convert the data to NumPy arrays
    user_indices = user_indices.reshape((-1, 1))
    product_indices = product_indices.T
    features = np.array(features)
    context = np.array(context)

    # Drop columns with all zero values
    features = features[:, (features != 0).any(axis=0)]

    # Check the number of remaining columns
    num_columns = features.shape[1]

    if num_columns > 20:
        # Sort columns based on the count of non-zero values
        column_counts = np.count_nonzero(features, axis=0)
        sorted_columns = np.argsort(column_counts)

        # Select the top 20 columns with the highest count of non-zero values
        columns_to_keep = sorted_columns[-20:]

        # Keep only the selected columns
        features = features[:, columns_to_keep]
    # Predict the ratings for all users and products
    predicted_ratings = model.predict([user_indices, product_indices, features, context])
    # Add predicted ratings as a column in the combined.csv file
    predicted = predicted_ratings.T.tolist()[0]
    n = len(products)  # Number of product
    product_id_map = [product.id for product in products]
    user_id_map = [user.id for user in users]

    for i, user_id in enumerate(user_id_map):
        user_ratings = predicted[i * n: (i + 1) * n]
        rating_dict = {rating: str(product_id) for rating, product_id in sorted(zip(user_ratings, product_id_map), reverse=True)[:10]}

        top_10_product_ids = []
        for rate, id_product in rating_dict.items():
            top_10_product_ids.append(id_product)
        recommendation = Recommendation(user_id=user_id, recommended_products=set(top_10_product_ids))
        recommendation.save()
        top_10_product_ids = []
```