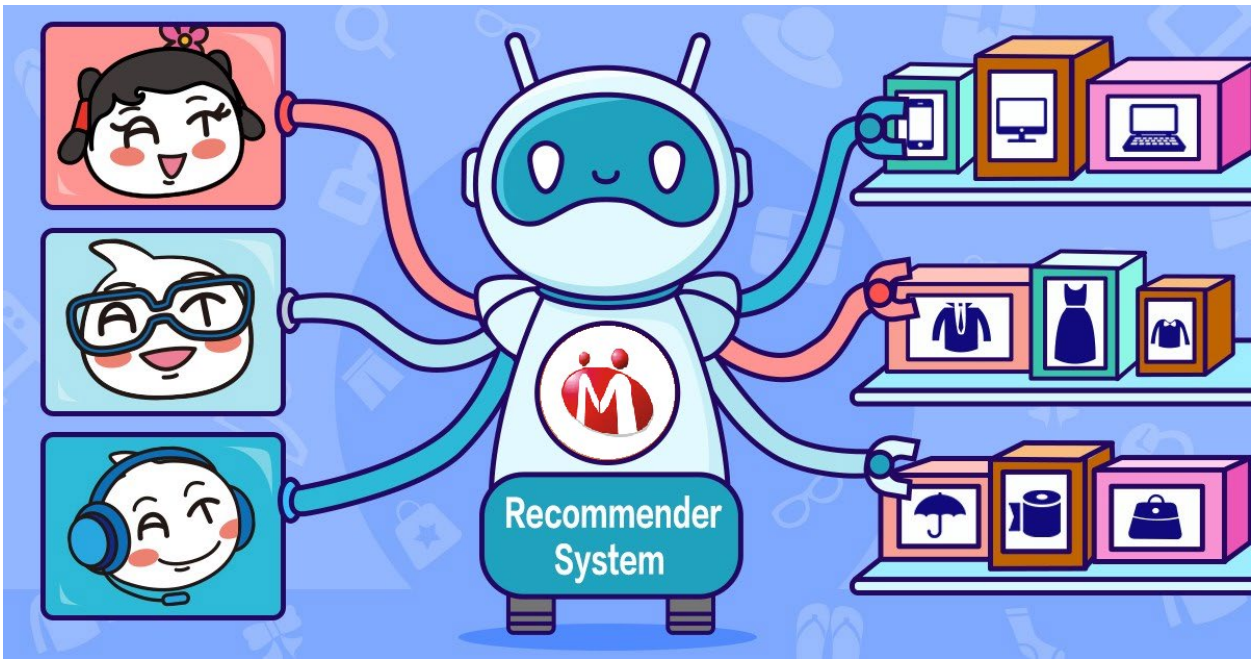
Recommendation System Documentation



Done By:

Ali Essam Ali Abdelmohsen

Contents

1 Introduction

Welcome to the documentation of our recommendation system project developed by me as part of the recruiting process at ConvertedIn. This document provides an in-depth overview of the creation, evaluation, improvement, and deployment of the recommendation system. I embarked on this project with the goal of building a robust recommendation system that enhances user experiences in an e-commerce environment. The system encompasses three distinct recommendation types: "Recommended for You," "Similar Items," and "Frequently Bought Together." These recommendations are generated through collaborative filtering, content-based filtering, and frequent itemset mining techniques.

2 Development Process

My journey in developing this recommendation system involved careful planning, thorough data exploration, effective algorithm implementation, comprehensive evaluation, and seamless deployment.

The development process can be distilled into the following stages:

- Understanding the Data: I began by delving into the provided e-commerce dataset. The dataset contained essential attributes like Invoice Number, Product Code, Product Description, Quantity, Invoice Date, Unit Price, Customer ID, and Country. I identified discrepancies, missing values, and potential issues that needed attention.

- Data Preprocessing: I cleaned the dataset by addressing missing values, correcting data types, and removing irrelevant or erroneous entries. This stage also involved handling canceled orders, negative quantities, and zero-unit prices.

- Exploratory Data Analysis (EDA): To understand the data, I performed exploratory data analysis. Visualizations helped uncover patterns, customer behaviors, and sales trends. The insights gained from EDA informed subsequent stages of development.

- Recommendation Generation: Leveraging collaborative filtering, content-based filtering, and frequent itemset mining, I developed a recommendation system capable of generating three types of recommendations: personalized suggestions, similar items, and frequently bought together items.

- Evaluation and Metrics: I was able to evaluate the recommendation system's performance. Metrics such as precision, recall, and Mean Average Precision (MAP) were employed to assess the system's effectiveness and identify areas for improvement.

- Deployment: The recommendation system was deployed as three interactive Flask apps: "Recommended for You," "Similar Items," and "Frequently Bought Together." These apps showcase the system's functionality and allow users to interact with the recommendations.

- Documentation: This comprehensive document is a testament to my commitment to documenting the development process, showcasing results, analysis, and insights, and providing clear explanations of the recommendation system's functionality.

3 Data Cleaning and Exploration

3.1 Understanding the Data

The provided dataset contained transactions from an e-commerce service, with attributes such as Invoice Number, Product Code, Product Description, Quantity, Invoice Date, Unit Price, Customer ID, and Country.

A quick review of the dataset revealed the following initial observations:

- Invoice Number (InvoiceNo): Each transaction had a unique identifier, with some starting with 'C,' indicating canceled transactions.
- Product Code (StockCode): Each product item had a distinct code.
- Product Description (Description): The name of the product in text format.
- Quantity: The number of each product sold in a transaction.
- Invoice Date (InvoiceDate): The date and time of the transaction.
- Unit Price (UnitPrice): The price of one unit of the product in currency.
- Customer ID (CustomerID): A unique code for each customer.
- Country: The country where the customer resides.

3.2 Data Cleaning

- Missing Values: An early concern was the presence of missing values in the dataset. I discovered that the 'Description' field contained 1454 missing values, while the 'CustomerID' field had a significant 25% of its entries missing. I addressed this by dropping rows with missing values, ensuring a clean dataset for further analysis and recommendation generation.

- Cancelled Orders: I noticed that transactions with 'C' as the first character in the 'InvoiceNo' column represented canceled orders. I isolated these entries, revealing negative quantity values. To maintain data integrity, I removed these canceled orders from the dataset.

- Zero Unit Prices: Another anomaly was the presence of zero-unit prices. I discovered that around 40 transactions had a unit price of zero, which seemed improbable. To maintain data consistency, I excluded these entries as well.

- Duplicate Entries: I identified duplicate entries in the dataset, which amounted to around 10,062 records. By eliminating these duplicates, I ensured the accuracy of subsequent analyses and recommendations.

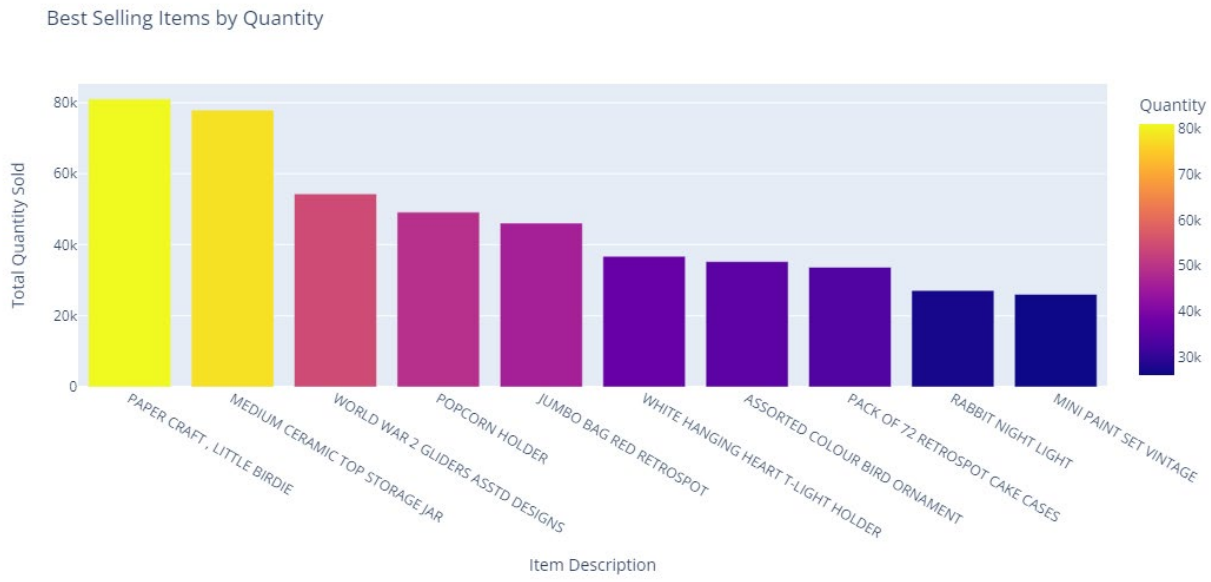
- Data Type Corrections: I converted the 'InvoiceDate' column to a datetime format for accurate time-based analyses. Additionally, I adjusted 'CustomerID' and 'InvoiceNo' columns to integer data types.

- Unique Descriptions per StockCode: A fascinating revelation was that some 'StockCode' values had multiple associated 'Description' values. To ensure uniformity and consistency, I created a dictionary mapping each 'StockCode' to the most common 'Description' value.

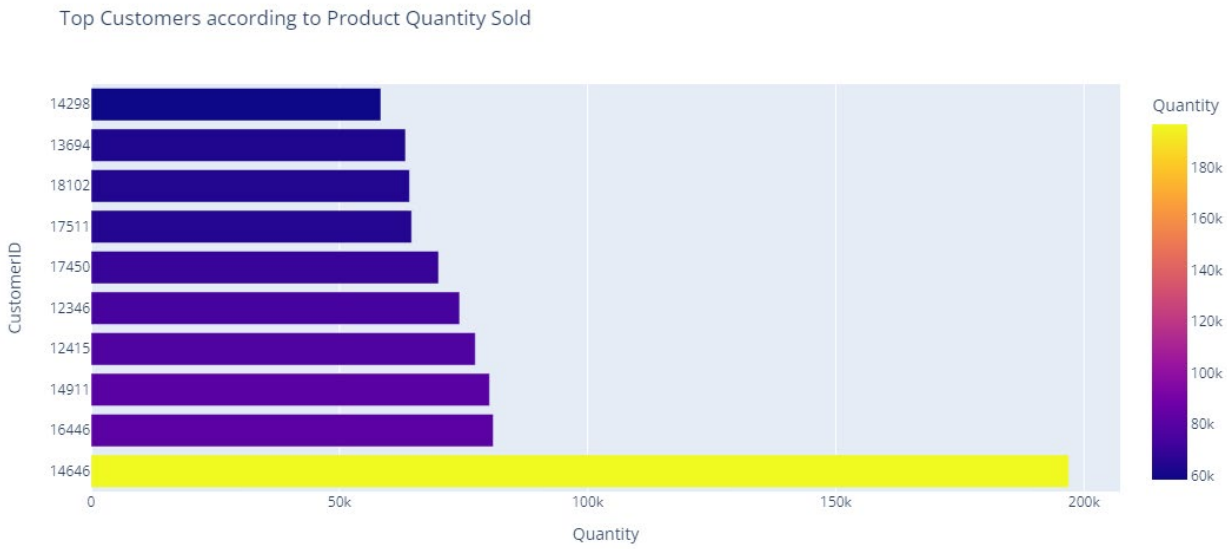
3.3 Data Exploration and Visualization

During the exploratory data analysis phase, I harnessed the power of data visualization to extract insights and trends from the dataset. Visualizations offered a more humanized understanding of the data's characteristics. Some examples of visualizations included:

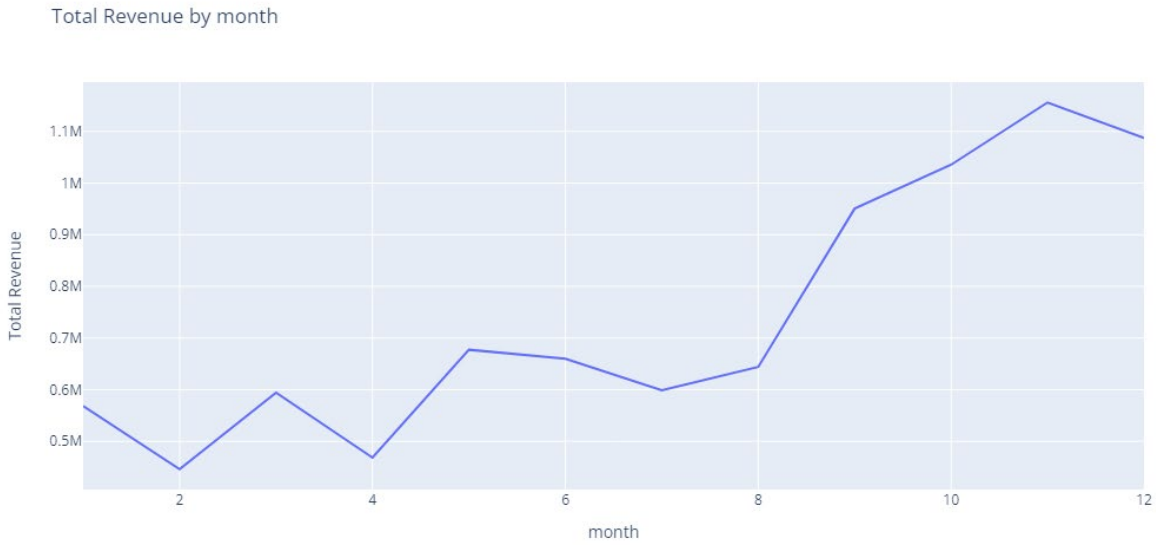
- Top Customers by Quantities: A bar plot illustrating purchase quantity by each customer, shedding light on customer buying behavior. Some customers had bought so few products.

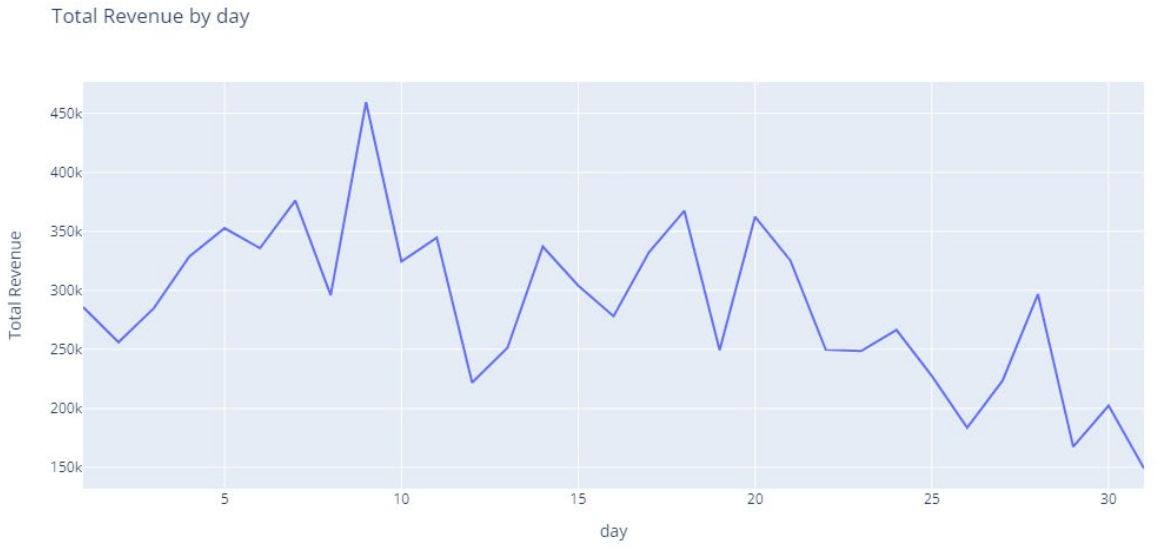


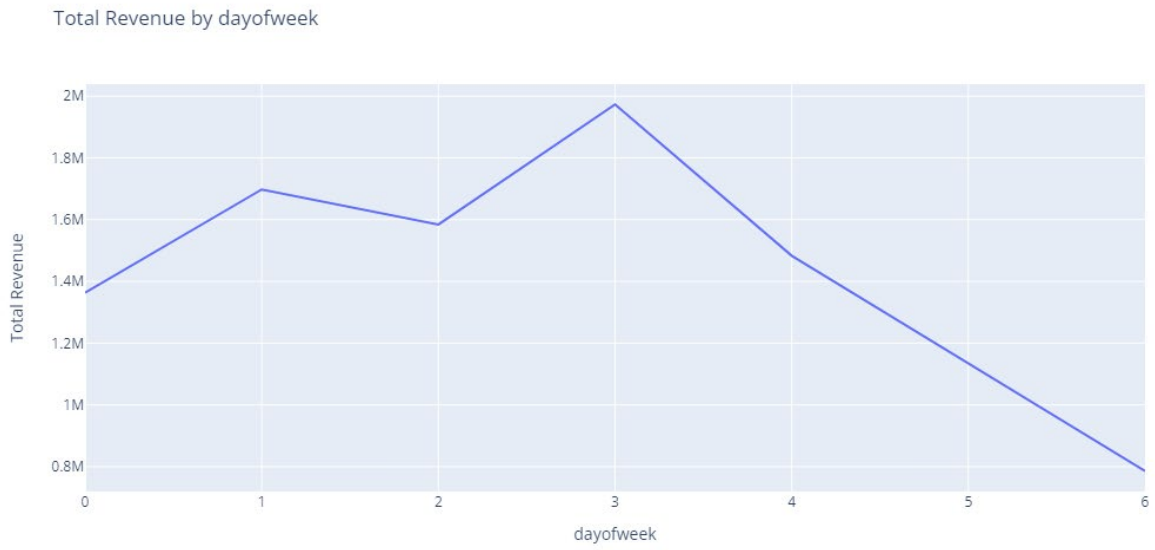
- Top Selling Products: A bar chart showcasing the most popular products based on purchase quantity, revealing customer preferences.

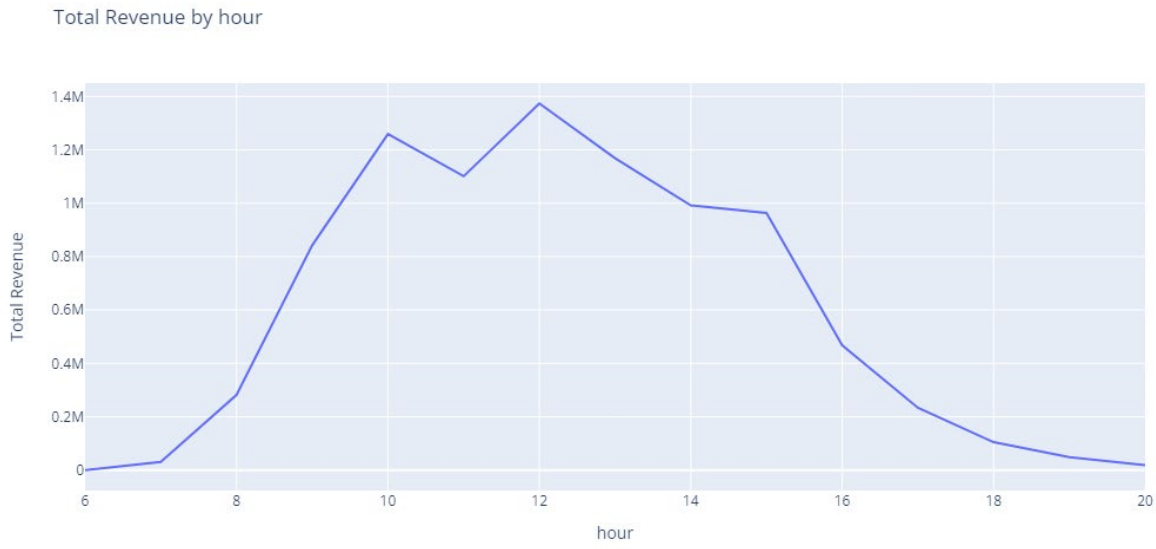


- Customer Purchase Patterns: Line charts illustrating individual customer purchase patterns over time, providing valuable insights into shopping behaviors and when is the highest revenue obtained.









Much more of these exploratory visualizations can be found in the data exploration section of the [Kaggle Notebook.]

3.4 Lessons Learned

This data cleaning and exploration process provided valuable insights and lessons:

- Missing Values: Addressing missing values early on is crucial for ensuring the quality of subsequent analyses and recommendations.

- Data Integrity: Recognizing and handling anomalies like canceled orders and zero-unit prices is essential for maintaining accurate data integrity.

- Data Consistency: Removing duplicate entries ensures that analyses and recommendations are based on accurate and consistent data.

- Data Understanding: Exploratory visualizations aid in understanding data characteristics, patterns, and trends that inform subsequent stages of development.

The data cleaning and exploration process was not only necessary for ensuring data quality but also provided me with a deeper understanding of the dataset's nuances. By addressing missing values, anomalies, and inconsistencies, I set the foundation for developing a robust recommendation system. The lessons learned during this process reinforced the importance of thorough data preparation and exploration in building effective data-driven solutions.

4 Recommendation Generation

The heart of our recommendation system lies in its ability to generate three types of recommendations tailored to user preferences and interactions:

4.1 Recommended for You

Personalized recommendations are generated by sending a User ID to the Flask endpoint. The "Recommended for You" recommendation system employs collaborative filtering, a popular technique that leverages the behavior and preferences of multiple users to make personalized suggestions.

4.1.1 How it works

1. User-Item Interaction Matrix: The system begins by creating a matrix where rows represent users and columns represent items. Each entry in the matrix reflects the interaction (purchase) between a user and an item.

2. Similarity Calculation: The system computes the similarity between users based on their interactions. Common similarity metrics include cosine similarity or Pearson correlation.

3. Neighborhood Selection: For a given user, the system identifies a set of similar users based on their interaction patterns. This forms the user's "neighborhood."

4. Item Ranking: The system ranks items that the user's neighbors have interacted with but the user has not. Items that are highly rated by the neighbors are more likely to be recommended.

5. Final Recommendations: The recommendations are generated by selecting the top-ranked items. The system filters out items the user has already interacted with to prevent redundancy.

4.1.2 Evaluation

4.1.2.1 Overall Model Metrics:

- Mean Average Precision: 0.34
- Total Precision: 0.16
- Total Recall: 0.02

4.1.2.2 Strengths:

- Mean Average Precision (MAP): The MAP of 0.34 indicates that, on average, the engine provides moderately accurate and relevant recommendations across all users. Users can expect to see items that are aligned with their preferences.

- Precision: With a total precision of 0.16, the engine's recommendations have a fair chance of being accurate. This means that there's a reasonable likelihood of users finding some of the recommended items appealing.

4.1.2.3 Weaknesses:

- Recall: The low total recall of 0.02 highlights the engine's challenge in capturing all the items that a user might purchase in the future. This indicates that there's room for improvement in terms of comprehensively identifying items that users are likely to buy.

4.1.2.4 Insights:

- The MAP suggests that, while the recommendations are somewhat accurate, there's still room for enhancing the engine's ability to suggest items that users will purchase in the future.

- The engine struggles with recall, indicating the need for methods to capture a broader range of user preferences and behaviors.

4.1.2.5 Improvement Strategies:

- Exploring more advanced recommendation algorithms, such as collaborative filtering, matrix factorization, or deep learning models, can potentially capture intricate user behavior patterns and improve recommendation quality.

- Utilizing more user features and contextual data, if available, can enhance personalization and provide more accurate recommendations.

- The engine could benefit from experiments with different algorithms, parameters, and data sources to fine-tune its accuracy and relevance.

4.2 Similar Items

By inputting an Item ID into the Flask endpoint, users receive suggestions for items similar to their selection. The "Similar Items" recommendation system focuses on item-item similarity to suggest products similar to those a user has shown interest in.

4.2.1 How it works

1. Item-Item Similarity Matrix: The system constructs a matrix where rows and columns represent items. The entries in the matrix indicate the similarity between pairs of items based on user interactions.

2. Item Similarity Calculation: The system calculates item similarity using various metrics such as cosine similarity or Jaccard coefficient. This quantifies how closely items are related based on user behavior.

3. Item Ranking: The system ranks similar items based on their similarity scores. Items with higher similarity are given precedence.

4. Recommendation Generation: The system generates recommendations by selecting the top-ranked items. These recommendations include items that are similar to what the user has already shown interest in.

4.2.2 Evaluation

4.2.2.1 Overall Model Metrics:

- Mean Average Precision: 0.32
- Total Precision: 0.17
- Total Recall: 0.14

4.2.2.2 Strengths:

- Mean Average Precision (MAP): With a MAP of 0.32, the engine ranks recommended items well, placing relevant products higher in the list.

- Precision: The total precision of 0.17 indicates that around 17% of the recommended items are relevant to users, showcasing the engine's ability to identify relevant products.

- Recall: A recall of 0.14 suggests that the engine effectively captures around 14% of the relevant items for each user, ensuring that it doesn't miss many relevant products.

4.2.2.3 Weaknesses:

- Low Precision and Recall: While the engine provides some relevant recommendations, there's room for improvement in terms of identifying a wider range of relevant items for users.

4.2.2.4 Insights:

- The engine's precision and recall values suggest that it offers a reasonable proportion of relevant recommendations, but there's potential to enhance its ability to identify more relevant items.

- Bias toward popular items, a cold start problem, and limited exploration beyond users' existing preferences are areas that could benefit from improvement.

4.2.2.5 Improvement Strategies:

- Hybrid approaches, combining item-item similarity with collaborative or content-based filtering, can enhance recommendation quality.

- Matrix factorization techniques like Singular Value Decomposition (SVD) can identify latent factors influencing user preferences, leading to better recommendations.

- Personalization techniques based on user behavior, demographics, and context can provide more relevant suggestions.

- Introducing diversity-aware strategies ensures a mix of popular and niche items, enhancing user exploration.

- Real-time data integration can improve recommendations by adapting to changes in user preferences.

4.3 Frequently Bought Together

You can send an Item ID to the Flask endpoint and receive suggestions for items frequently purchased alongside the selected item. The "Frequently Bought Together" recommendation system employs association analysis to identify items that tend to be purchased together.

4.3.1 How it works

1. Purchase History Data: The system gathers data on user purchase histories, capturing instances where users buy multiple items in a single transaction.

2. Frequent Itemset Mining: The system applies algorithms like Apriori or FP-Growth to identify sets of items that are frequently purchased together. These sets are known as "frequent itemsets."

3. Association Rule Generation: From the frequent itemsets, the system generates association rules. An association rule expresses the likelihood of one item being purchased when another item is already bought.

4. Confidence and Support: The system evaluates the strength of association rules using metrics like confidence and support. Confidence indicates the likelihood of the consequent item being purchased when the antecedent item is bought.

5. Recommendation Generation: The system generates recommendations based on the association rules with high confidence values. When a user adds an item to their cart or purchases an item, the system suggests other items associated with the purchased item.

4.3.2 Evaluation

4.3.2.1 Overall Model Metrics:

- Mean Average Precision: 1.00
- Total Precision: 1.00
- Total Recall: 0.00

4.3.2.2 Strengths:

- High Precision: The engine boasts a high precision score, indicating that the suggested items are closely related to the user's initial purchase.

- Mean Average Precision (MAP): The perfect MAP of 1.00 demonstrates consistent accuracy in recommendations, aligning with items that are actually bought together.

4.3.2.3 Weaknesses:

- Low Recall: The low recall score of 0.00 suggests that the engine is missing opportunities to recommend additional products that are frequently bought together with the main item.

4.3.2.4 Insights:

- Precision-Recall Trade-off: Striking a balance between precision and recall is essential. While high precision ensures relevant recommendations, efforts to improve recall without compromising precision are vital to capturing a broader range of items.

- Diversification of Recommendations: Techniques focusing on diversification, considering user preferences, browsing history, and trending items, can enhance recall and improve overall recommendations.

4.3.2.5 Improvement Strategies:

- User Feedback and Testing: User testing and feedback collection can provide insights into user preferences and improve recommendation accuracy.

- Continuous Improvement: Regular analysis of user interactions, performance metrics, and algorithm fine-tuning are essential for ongoing recommendation engine improvement.

- Exploration of Advanced Techniques: Exploring advanced methods like collaborative filtering, content-based filtering, and hybrid approaches can lead to more comprehensive recommendations.

- Evaluation on Larger Dataset: Evaluating the engine on a diverse and larger dataset will ensure the generalizability of results.

5 Implementation and Deployment

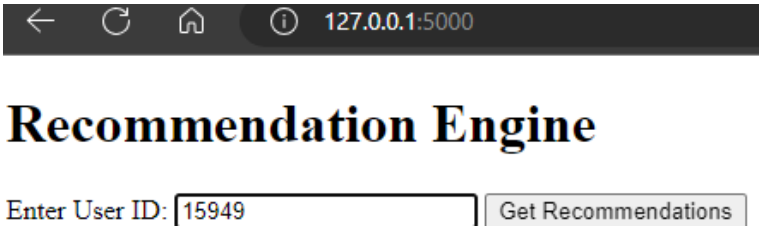
As part of this recommendation system, I developed three interactive Flask apps that showcase the system's functionality:

5.1 Recommended for You App

This app delivers personalized recommendations to users based on their preferences and interactions.

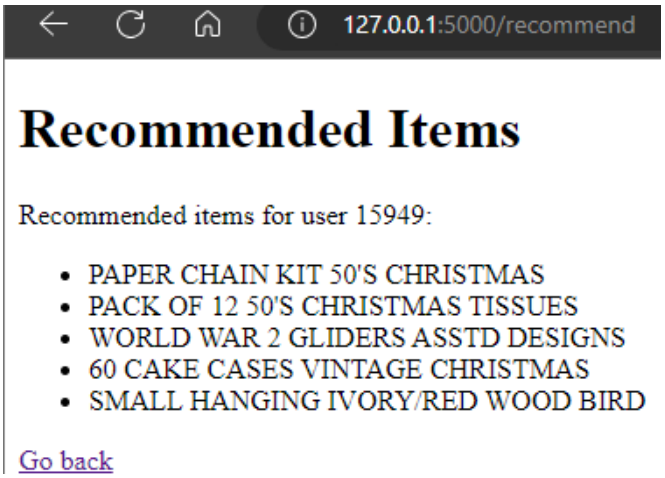
5.1.1 Data Entry

The app takes user ids as an input and utilizes the button to get recommendations based on other users.



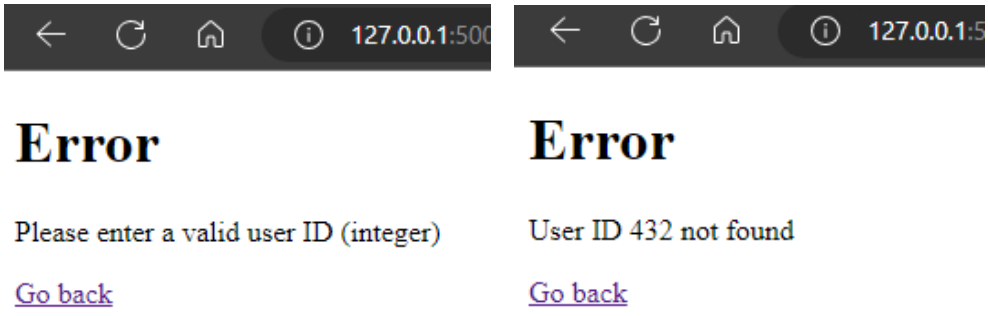
5.1.2 Data Retrieval

The data is retrieved and printed on the screen. We can also see how the items are related to each other.



5.1.3 Error Handling

Error handling for wrong inputs like wrong format or missing id.

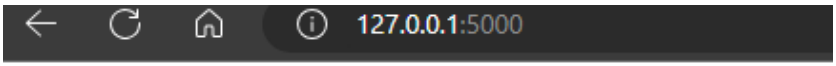


5.2 Similar Items App

Users can explore items similar to their selected product by providing its Item ID.

5.2.1 Data Entry

The app takes user ids as an input and utilizes the button to get recommendations based on similar items.

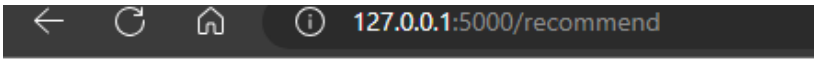


Similar Items Recommendation

Enter Item ID: 21704 Get Similar Items

5.2.2 Data Retrieval

The data is retrieved and printed on the screen. We can also see how the items are related to each other.



Similar Items

Similar items for item BAG 250g SWIRLY MARBLES:

- VINTAGE SNAKES & LADDERS
- WOODEN BOX OF DOMINOES
- 4 TRADITIONAL SPINNING TOPS
- BAG 500g SWIRLY MARBLES
- BLUE HARMONICA IN BOX
- VINTAGE SNAP CARDS
- TRADITIONAL WOODEN CATCH CUP GAME
- BAG 125g SWIRLY MARBLES
- WORLD WAR 2 GLIDERS ASSTD DESIGNS
- TRADITIONAL MODELLING CLAY

Go back

5.2.3 Error Handling

Error handling for wrong inputs like missing id.

Error

Please enter a valid user ID (integer)

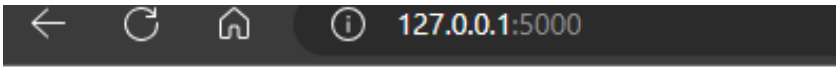
Go back

5.3 Frequently Bought Together App

Offers users suggestions for items frequently purchased alongside their selected item.

5.3.1 Data Entry

The app takes user ids as an input and utilizes the button to get recommendations based on items frequently bought together.

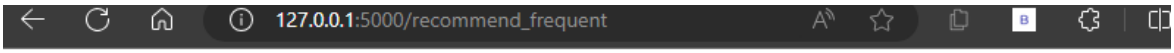


Frequently Bought Together

Enter Item ID: 22383 Recommend

5.3.2 Data Retrieval

The data is retrieved and printed on the screen. We can also see how the items are related to each other.



Recommended items for "LUNCH BAG SUKI DESIGN "

- **Recommended:** LUNCH BAG SPACEBOY DESIGN (22382)
- **Recommended:** LUNCH BAG CARS BLUE (20728)
- **Recommended:** LUNCH BAG SPACEBOY DESIGN (22382)
- **Recommended:** LUNCH BAG BLACK SKULL. (20727)
- **Recommended:** LUNCH BAG PINK POLKADOT (22384)

Go back

5.3.3 Error Handling

Error handling for wrong inputs like missing id.

Error

Please enter a valid user ID (integer)

Go back

6 Conclusion

In closing, I am enthusiastic about the journey ahead at ConvertedIn. The experience of building these recommendation systems has deepened my understanding of data science and machine learning. I am excited to bring this knowledge to the team and learn from their expertise. I am committed to contributing effectively, embracing new challenges, and applying my skills to drive innovation and success at ConvertedIn.

7 Project Links

GitHub Repository

Kaggle Notebook