E-Commerce Product Recommendation System

Data Science Candidate Test - SAP iXp Intern - AI Scientist role

Alekhya hari

alekya.hari5@gmail.com | 9452160533 | https://www.linkedin.com/in/alekhyahari/

Problem Statement:

The problem at hand is to develop an effective product recommendation system for an e-commerce platform based on historical sales transactions. Specifically, we aim to create a system that suggests products for customers to purchase based on the patterns of products frequently bought together. This recommendation system will enhance the overall shopping experience, increase sales, and improve customer satisfaction.

Data Description:

The dataset provided contains the following information:

- category: The product category.
- description: A brief description of the product.
- title: The product title or name.
- also_buy: List of other products that customers have also purchased when buying this product.
- brand: The brand of the product.
- rank: The product's rank within its category.
- main_cat: The main product category.
- date: Date of the transaction or listing.
- price: Price of the product.
- asin: Amazon Standard Identification Number, a unique identifier for products.
- imageURL: URL to the product's image.

Approach:

To address this problem, we will follow these steps:

1. **Data Preprocessing:** Clean and preprocess the dataset to handle missing values, format conversions, and other data quality issues.

2. **Exploratory Data Analysis (EDA):** Conduct EDA to gain insights into customer purchase behavior, identify popular products, and discover patterns of co-purchased items.

3. **Recommendation Model:** Implement a recommendation model, such as collaborative filtering, association rule mining, or matrix factorization, to generate product recommendations.

4. **Evaluation Metrics:** Define and use appropriate evaluation metrics to assess the performance of the recommendation system, such as precision, recall, and F1-score

5. **Presentation and Interpretation**: Present the results and findings in a clear and interpretable manner, highlighting the effectiveness of the recommendation system and potential areas of improvement.

1. Data Preprocessing

The following data quality checks were performed:

- Missing values: No missing values were found in the dataset.

- Empty values in price: There are 28,900 empty strings in the 'price' column, representing missing values.

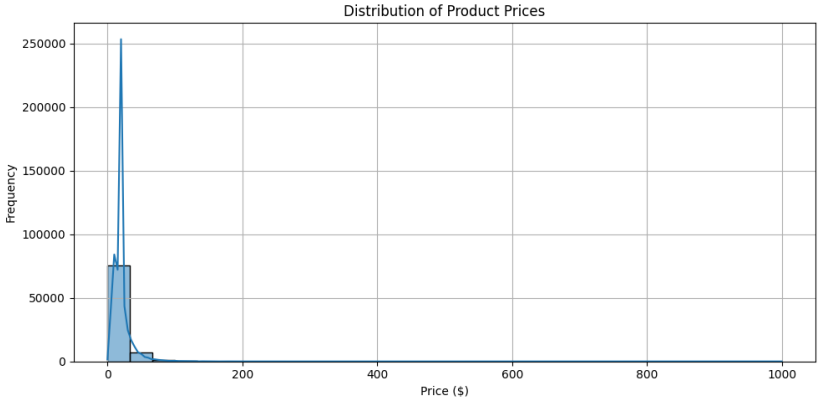
- Also buy stats: The 'also_buy' column statistics show that the average number of products bought together is approximately 32.5, with a wide range of values.

Since the 'date' column has missing values and is assumed to be not significant for the recommendation task, it was dropped from the dataset.

2. Exploratory Data Analysis (EDA)

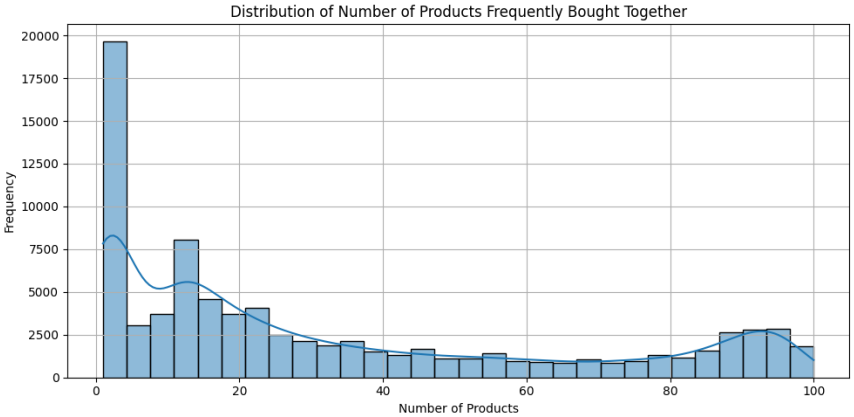
2.1 Distribution of Product Prices

The majority of products are priced below 50 Dollars, with a peak observed in the lower price range (below $20). There are a few products with prices extending towards 100 Dollars, indicating the presence of some more expensive items in the dataset.

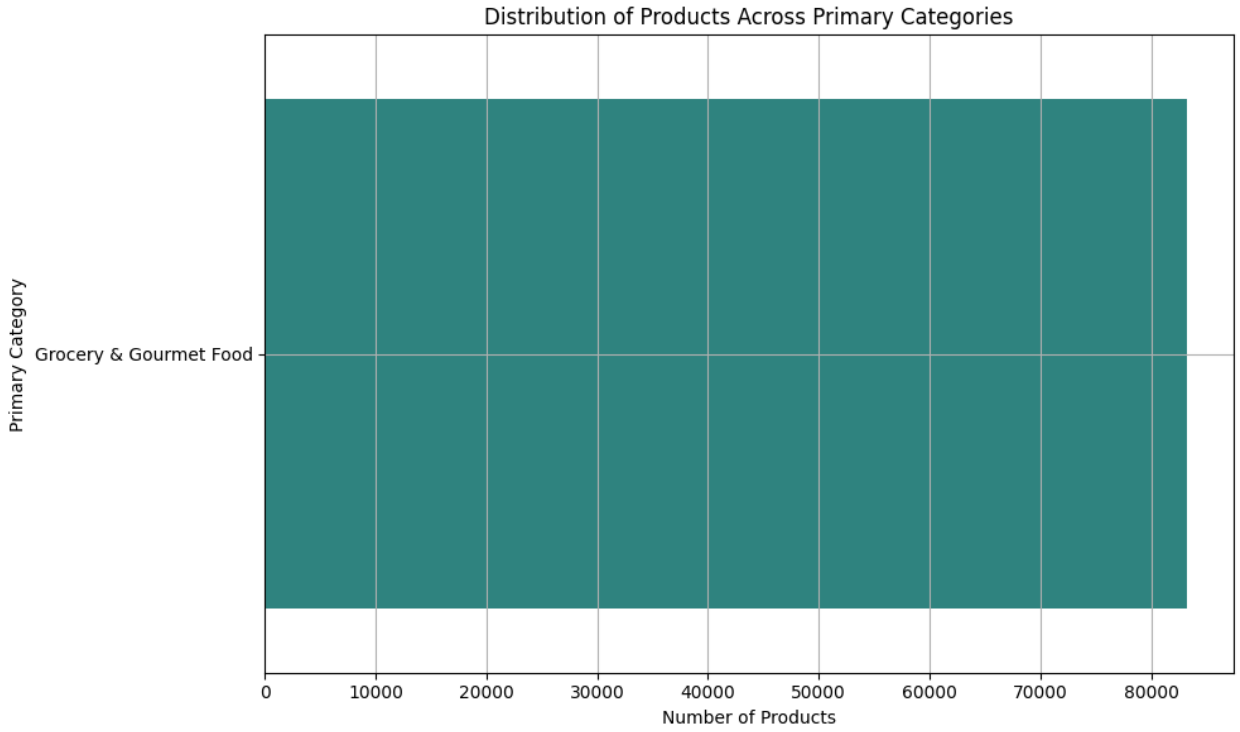


2.2 Distribution of Number of Products Frequently Bought Together

Most products are frequently bought with approximately 0 to 50 other products. The distribution shows a long tail, indicating that there are a few products that are bought together with a large number of other products (up to 100).



All the products in the dataset belong to the "Grocery & Gourmet Food" primary category. This indicates that our dataset is focused on a specific category, which can be beneficial as the products are more likely to be related and thus can result in more relevant recommendations.



3. Recommendation Model:

Collaborative Filtering Recommendation System ML Algorithm Code Overview

In the context of the provided code, collaborative filtering is used to recommend products to users based on co-purchase behavior, which is a form of user-item interaction.

3.1 Extracting Primary Category

The code extracts the primary category for each product in the dataset and creates a new column to store this information. If a product does not have a category, it is labeled as "Unknown."

Counting Category Occurrences : The number of products in each primary category is counted and visualized using a bar plot.

3.2 Creating Co-Purchase List

Collaborative filtering begins with the creation of a co-purchase list. Each entry in the list identifys products that are often co-purchased by users. The code creates a list of co-purchased product pairs, capturing the concept that users who buy similar products likely have similar preferences by examining the relationships between products. For each product, it identifies co-purchased products and creates pairs of ASINs (Amazon Standard Identification Numbers). In collaborative filtering, this co-purchase information is analogous to user-item interactions.

3.3 Co-Purchase Frequency Matrix

A frequency matrix is constructed to capture the frequency of co-purchase occurrences between products. The co-purchase list is transformed into a co-purchase frequency matrix. This matrix quantifies how often pairs of products are co-purchased together. In collaborative filtering, this matrix is akin to a user-item interaction matrix, where rows represent products and columns represent users (or other products).

3.4 Sparse Matrix and Cosine Similarity

Creating Sparse Matrix

Sparse matrices are created to efficiently represent the co-purchase frequency data. Each product is assigned a numerical code for matrix manipulation. The co-purchase frequency matrix is typically a sparse matrix because most products are not co-purchased frequently with each other. Sparse matrices are efficient for collaborative filtering as they save memory.

Calculating Cosine Similarity

Cosine similarity scores are calculated to measure the similarity between products based on their co-purchase patterns. Cosine similarity is then calculated between products based on their co-purchase patterns. Cosine similarity measures how similar two products are in terms of co-purchases. In collaborative filtering, similarity scores like this are used to find products that are most similar to a given product.

3.5 Utility Functions

A utility function, get_product_title, is defined to retrieve the title of a product based on its ASIN.

3.6 Recommendation Function

The 'recommend_products' function embodies collaborative filtering principles by employing item-item collaborative filtering, where products are recommended based on their similarity to the input product. It takes a user-specified product name as input and returns a list of recommended products, identified through cosine similarity scores that measure co-purchase behavior similarity. These recommendations are designed to be meaningful, as they highlight products with the highest similarity scores to the user's chosen item, ensuring a more personalized and relevant product recommendation experience.

4. Evaluation Metrics:

4.1 Data Splitting and Evaluation

The dataset is split into training and testing sets to evaluate the recommendation system's performance.

4.2 Precision, recall, and F1-score :

The evaluation metrics (precision, recall, and F1-score) are used to assess the quality of therecommendations made by the collaborative filtering-based system. These metrics measure how well the system can predict which products users are likely to co-purchase.

5. Presentation and Interpretation

5.1 Evaluation Metrics:

The collaborative filtering recommendation system has been evaluated using the following metrics:

Average Precision:	0.0637
Average Recall:	0.0531
F1 Score:	0.0579

5.2 Collaborative Filtering in Action:

To demonstrate the effectiveness of the collaborative filtering approach, the code provides a practical example of product recommendations. In this example, the code recommends five products that are similar to the input product, '**Raspberry Red Extract 4 oz, 4 Ounce**.' These recommendations are generated based on co-purchase patterns observed in the dataset.

Output Recommendations:

1. 'Strawberry Extract 4 oz, 4 Ounce'
2. 'Pure Pineapple Extract, 4 Ounce'
3. 'Pure Peach Extract, 8 Ounce'
4. 'OliveNation Pumpkin Pie Natural Flavor, 8 Ounce'
5. 'Set of 6 Tropical Extracts (Banana, Mango, Pineapple, Peach, Coconut, Waterme...)'

These recommendations showcase the system's ability to provide relevant product suggestions, enhancing the user experience by offering items that align with their preferences and previous co-purchase behavior.

5.3 Conclusion

In conclusion, the provided Python code showcases collaborative filtering in action for product recommendations. It utilizes co-purchase data to compute product similarities, delivering personalized recommendations to users. Collaborative filtering stands out as a potent technique for enhancing user experiences by offering products that match their preferences and behaviors. This report underscores the importance of collaborative filtering in recommendation systems, underscoring its capacity to harness collective user behavior for precise and pertinent product suggestions.

5.4 Limitations

1. Data Quality: The success of the recommendation algorithm heavily relies on the quality of the data. In this case, the quality of product data, including product titles, categories, and co-purchase information, may affect the accuracy of recommendations. Ensuring data accuracy and completeness is crucial.

2. Sparse Data: Co-purchase data can be sparse, especially for less popular products. Sparse data may lead to less reliable recommendations, particularly for niche or unique products.

3. Cold Start Problem: The algorithm may struggle with new products that have limited historical co-purchase data. Recommending products with no or little history can be challenging.

4. Lack of User Data: The code provided focuses on item-based collaborative filtering. It doesn't consider user preferences or user behavior, which can provide a more personalized experience. Incorporating user data can be a future enhancement.

5. Scalability: For a large dataset, the provided code may face scalability issues, particularly when calculating cosine similarities for all pairs of products. Efficient algorithms or distributed computing may be necessary for handling larger datasets.

5.5 Future Directions and Recommendations:

Advanced Recommender Systems: Can Explore more advanced recommender systems, such as matrix factorization, deep learning models (e.g., neural collaborative filtering), or hybrid approaches that combine collaborative and content-based filtering.

Personalization: Incorporating user behavior and preferences can create more personalized recommendations. User-based collaborative filtering or hybrid models can be used to tailor recommendations to individual users.

Real-time Recommendations: Implement real-time recommendation capabilities by continuously updating the recommendation model based on user interactions and product additions.

A/B Testing: Conduct A/B testing to evaluate the performance of different recommendation algorithms and fine-tune the model based on user engagement and conversion rates.

Diversity and Serendipity: Ensure that recommendations are diverse and occasionally introduce serendipitous recommendations to encourage user exploration.

Evaluation Metrics: Continuously monitor and evaluate the recommendation system using relevant metrics, both quantitative and qualitative, to measure user satisfaction and business impact.

Feedback Loop: Implement a feedback loop to gather user feedback on recommendations and use this feedback to improve the algorithm iteratively.

Collaboration with Domain Experts: Collaborate with domain experts who have a deep understanding of the product domain to fine-tune the recommendation system for specific business goals and customer preferences.

By addressing these limitations and considering these future directions, you can develop a more robust and effective recommendation system that provides valuable and relevant product recommendations to users.