*E-commerce Product Recommendation System*

COEN – 240 Machine Learning



Submitted By:

Anwitha Arbi (07700005478)

Pujitha Kallu (W1653660)

**Table of Contents**

1. [***Abstract***](#_gjdgxs) *3*
2. ***Product Recommendation***4

1.[Introduction](#_1fob9te) 4

2.[Training](#_3znysh7) 5

3.[Inference](#_2et92p0) 7

4. *Data Pre-processing 7*

1. ***Data Sets and Algorithms*** *9*

1. Rank based product recommendation system.10

2. User Based Product Recommendation System..…...11

3. Model Based Collaborative Filtering………..……...15

1. [***Dataset Processing***](#_1ksv4uv)…………..………………...…….…..17
2. [***Result And Outcome***](#_1ci93xb) 18
3. **Avg\_actual\_ratings vs Avg\_predicted\_ratings.……….**24
4. **Roc & Precision vs Recall Curve………………...……**.29
5. [***Conclusion***](#_3whwml4) 29
6. [***References***](#_2bn6wsx) 30

# **Abstract**

The "Ecommerce-product-recommendation-system" project endeavors to transform the e-commerce landscape by introducing an advanced recommendation system designed to enhance user experience and drive business growth. Leveraging state-of-the-art machine learning algorithms, including rank-based recommendation, similarity-based collaborative filtering, and model-based collaborative filtering, the system aims to provide users with personalized product recommendations tailored to their preferences and browsing history. By analyzing user interactions and behavior, the system will address challenges such as the Cold Start Problem and sparsity in user-item interactions, ensuring accurate and relevant recommendations even for new users or items with limited data. The project will adopt rigorous evaluation methodologies, including the calculation of metrics like Root Mean Squared Error (RMSE), to assess the effectiveness and performance of the recommendation algorithms. Furthermore, drawing insights from recent research papers and studies in the field of recommendation systems, the project will contribute to advancing the knowledge and understanding of personalized recommendation algorithms, paving the way for future innovations in e-commerce. Through its comprehensive approach and innovative solutions, the "Ecommerce-product-recommendation-system" project aims to revolutionize the e-commerce industry, fostering user satisfaction, increasing engagement, and driving business success.

In detail, the project will begin by acquiring and preprocessing the dataset, ensuring data cleanliness and compatibility with the recommendation algorithms. We then implement various recommendation algorithms, including rank-based recommendation, similarity-based collaborative filtering, and model-based collaborative filtering, each serving different use cases and addressing specific challenges in recommendation systems. Through iterative testing and optimization, we fine-tuned the algorithms to maximize recommendation accuracy and relevance while minimizing computational complexity.

Moreover, the project will prioritize user privacy and data security by implementing robust data encryption and anonymization techniques to safeguard sensitive user information. Additionally, the system will offer transparency and control to users by providing options to adjust recommendation settings, opt out of personalized recommendations, and review their data usage preferences.

Furthermore, the project will explore innovative features such as real-time recommendation updates, personalized notifications, and adaptive recommendation interfaces to further enhance user engagement and satisfaction. By continuously monitoring user feedback and interaction patterns, the system will dynamically adjust its recommendation strategies to reflect evolving user preferences and market trends, ensuring a seamless and personalized shopping experience for users.

Overall, the "Ecommerce-product-recommendation-system" project represents a comprehensive and innovative approach to recommendation systems in e-commerce, aiming to set new standards for user experience, personalization, and business performance in the digital marketplace. Through its collaborative efforts, cutting-edge technology, and commitment to excellence, the project seeks to shape the future of e-commerce and empower businesses to thrive in the competitive online landscape.

# **Product Recommendation**

## **Introduction**

In the rapidly evolving world of e-commerce, the role of product recommendation systems has become increasingly vital. These systems are designed to enhance the shopping experience by providing personalized product suggestions based on users' browsing and purchasing histories. Our "Product Recommendation System" project is an innovative approach aimed at leveraging advanced machine learning techniques to develop a system that not only understands user preferences but also predicts future interests with high accuracy. By employing collaborative filtering, content-based filtering, and hybrid methods, this project aims to create a seamless and intuitive shopping experience that drives customer satisfaction and business growth.

The primary objective of this project is to tackle the dual challenges of product recommendation and customer targeting in the e-commerce sector. By focusing on recommending products with the highest number of ratings and targeting new customers with popular products, we aim to solve the Cold Start Problem and enhance user engagement. Our approach involves analyzing user behavior to generate relevant and personalized recommendations, thereby increasing customer satisfaction and boosting sales. The ultimate goal is to develop a scalable and efficient system that can be easily integrated into existing e-commerce platforms, providing businesses with a competitive edge in the marketplace. Through this project, we strive to demonstrate the transformative potential of machine learning in personalizing the online shopping experience and driving e-commerce success.

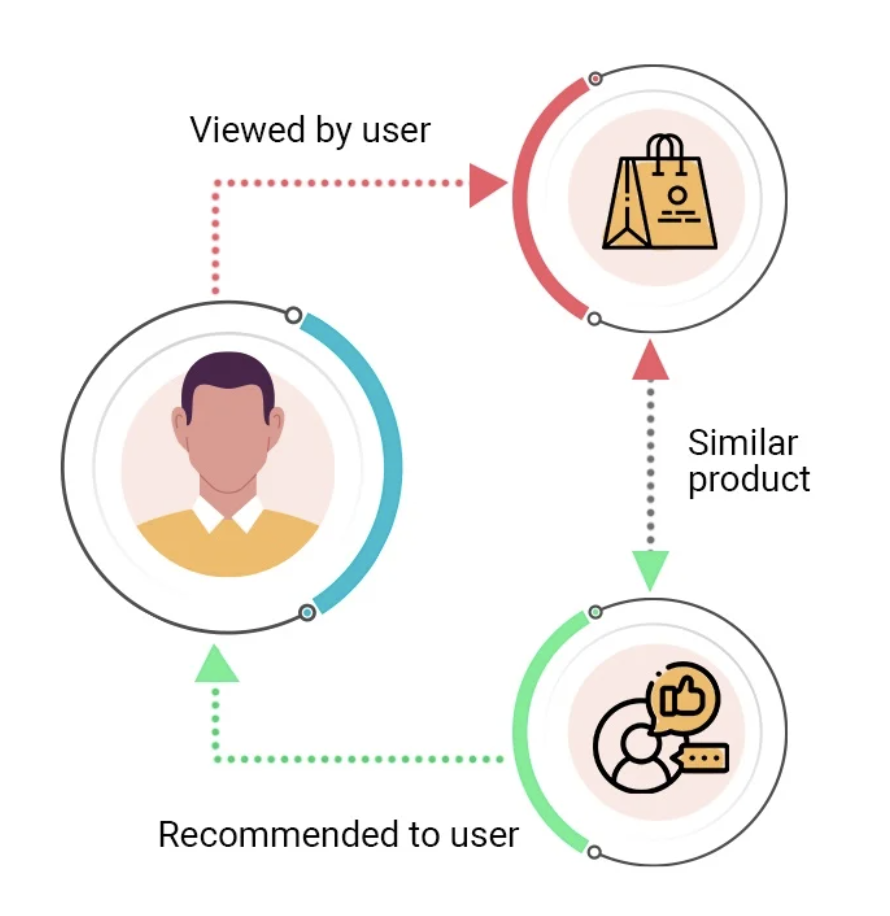


Fig – 1.0: Data Flow diagram

## **Training**

***Data Collection:***

Training data for a product recommendation system should encompass a wide range of user interactions and product details to ensure comprehensive and accurate recommendations. User interaction data includes both explicit feedback, such as star ratings and reviews, and implicit feedback, such as clicks, views, purchases, and time spent on product pages. Explicit feedback provides direct indicators of user preferences, while implicit feedback offers abundant insights into user interests and engagement patterns.

Additionally, collecting user profile data, including demographic information like age, gender, location, and occupation, helps create user segments and tailor recommendations. Detailed product data, encompassing features like category, price, brand, specifications, and descriptions, is essential for content-based filtering algorithms. This data forms the backbone of personalized recommendations by enabling the system to understand user behavior and product attributes.

Moreover, integrating contextual data, such as the time of day, day of the week, and seasonal trends, can significantly enhance the relevance of recommendations by aligning them with temporal patterns in user behavior. Social data from user interactions on social media platforms can also provide valuable insights into user preferences and influence product recommendations. Incorporating geolocation data allows for location-based recommendations, which can be particularly useful for businesses with physical stores or location-specific promotions.

In summary, a robust data collection strategy for a product recommendation system should include:

. User interaction data: Explicit feedback (ratings, reviews) and implicit feedback (clicks, views, purchases).

2. User profile data: Demographic information (age, gender, location, occupation).

3. Product data: Features like category, price, brand, specifications, and descriptions.

4. Contextual data: Time of day, day of the week, seasonal trends.

5. Social data: Interactions and preferences from social media platforms.

6. Geolocation data: Location-based information for personalized recommendations.

7. Data validation: Processes to ensure data accuracy and completeness.

8. External data sources: Market trends and competitor analysis for enriched insights.

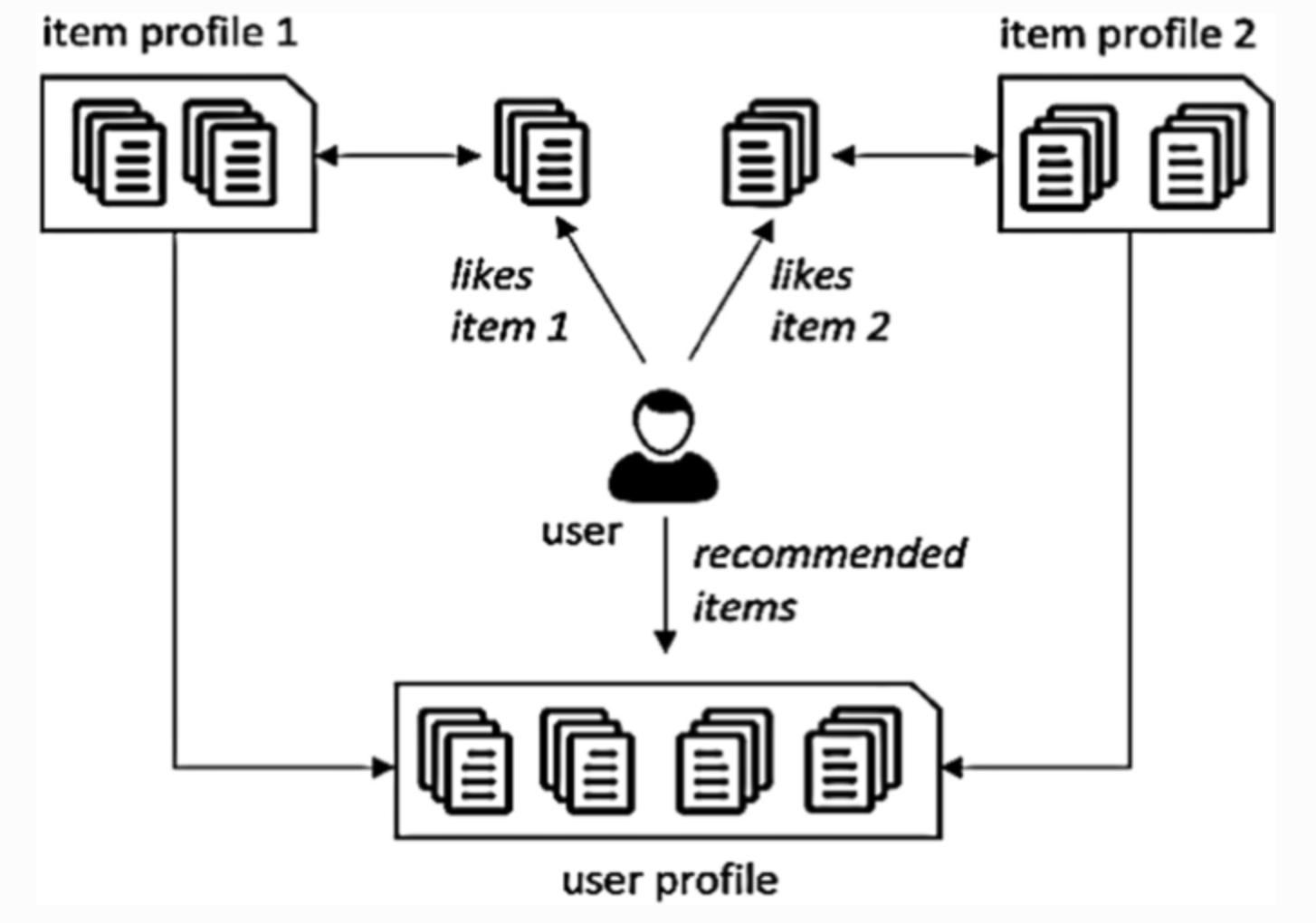


Fig – 2.0: Data collection

***Model Training:***

To achieve successful model training in a product recommendation system, a structured and comprehensive approach is essential. Initially, the dataset is divided into training, validation, and test sets, with proportions typically set at 70-80% for training, 0-20% for validation, and 0-20% for testing. This division ensures effective learning and accurate performance evaluation. Employing k-fold cross-validation further enhances model robustness by training the model 'k' times on different subsets, preventing overfitting and ensuring generalization across diverse data samples.

The selection of appropriate algorithms plays a pivotal role in model training. For rank-based recommendation algorithms, which focus on recommending popular products based on the highest number of ratings, techniques like collaborative filtering based on user-item interactions or item popularity can be employed. Conversely, model-based collaborative filtering algorithms leverage user behavior and preferences to predict ratings and recommend products tailored to individual users.

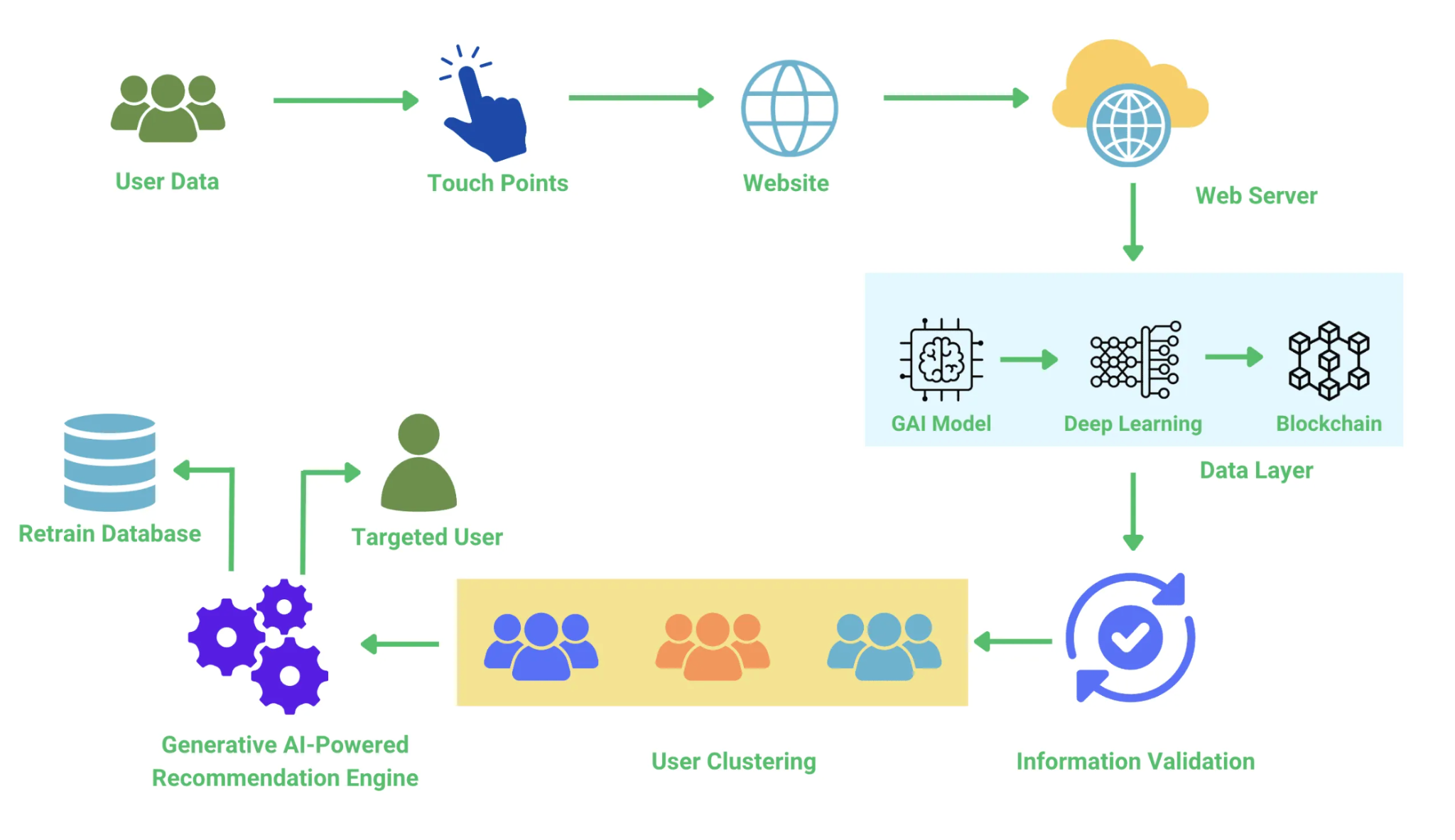


Fig: 2.1 Model based Training Hierarchy

## **Inference**

By following this structured approach to model training, a product recommendation system can be developed to deliver accurate, efficient, and personalized recommendations, ultimately enhancing user satisfaction and driving business growth.

## **Data preprocessing**

Data preprocessing is a crucial step in preparing the dataset for machine learning models. Initially, we load the dataset using pandas' read\_csv() function, allowing us to examine its structure and identify any missing values or inconsistencies. For missing values, techniques like imputation can be employed to replace them with suitable estimates such as the mean, median, or mode of the respective feature. Categorical variables are often encoded using methods like one-hot encoding to convert them into a numerical format that machine learning algorithms can process effectively.

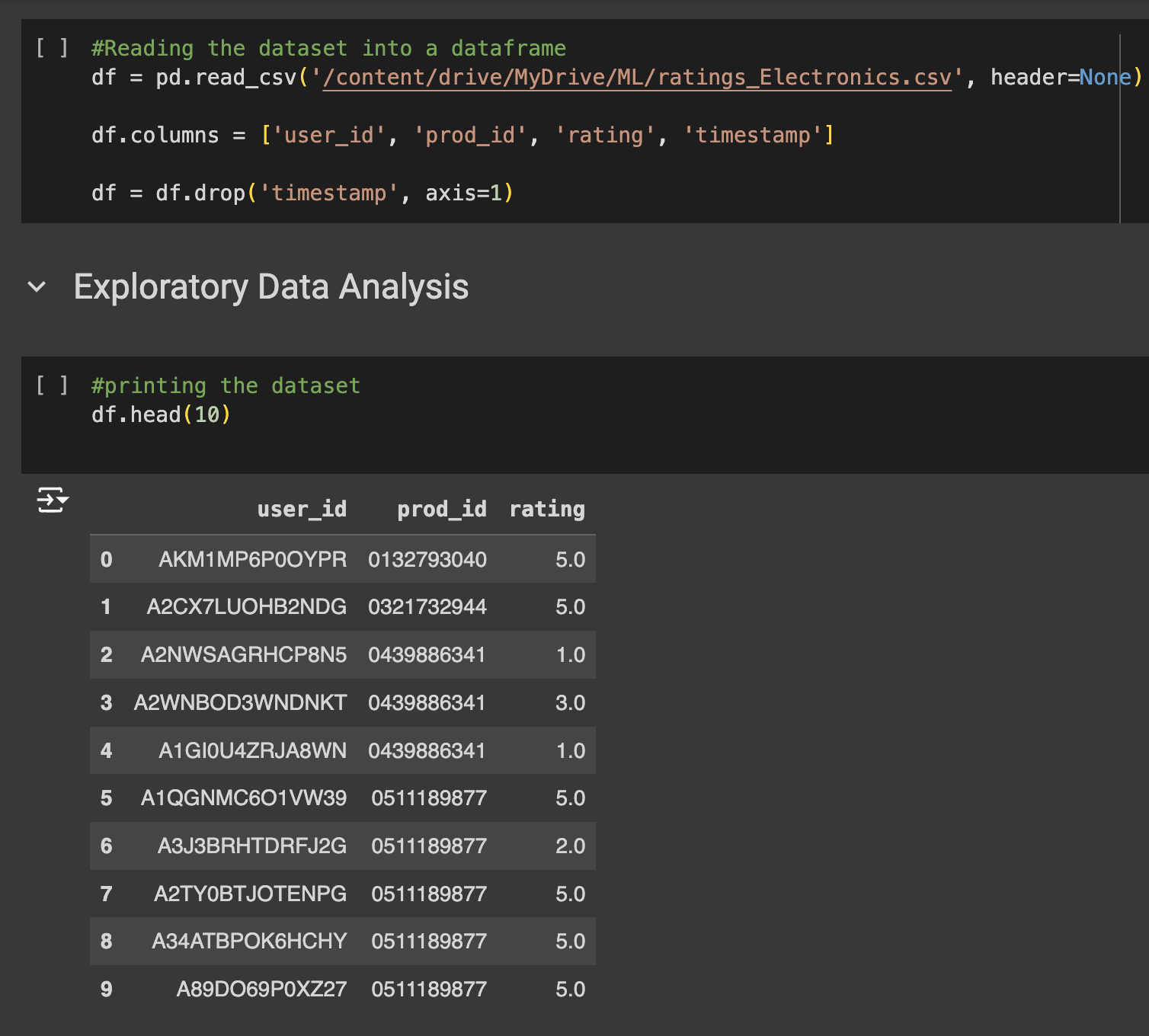


Fig 3.0 Preprocessing data

Moreover, feature scaling is applied to standardize the range of features, ensuring they contribute equally to the model's training process. Techniques such as z-score normalization or min-max scaling transform numerical features to fall within a specified range.

Additionally, feature engineering techniques may be used to create new features or modify existing ones to enhance the model's predictive performance. This can involve generating interaction terms, polynomial features, or extracting relevant information from existing features.

After preprocessing, the dataset is typically split into training and testing sets using methods like stratified sampling to ensure both sets are representative of the overall data distribution. Finally, the preprocessed data is ready for training and evaluation by machine learning models. This systematic approach to data preprocessing ensures that resulting models are robust, accurate, and capable of making reliable predictions on new data.

# 

Fig: 3.1 : Preprocessing data for Training

The preprocessing step is crucial in data preparation for recommendation systems for several reasons:

**Quality of Recommendations:** Users with only a few interactions may not provide enough data for the recommendation algorithm to generate accurate and meaningful recommendations. By filtering out users with fewer interactions, we focus on those who have provided sufficient feedback, leading to better quality recommendations.

**Reducing Sparsity:** In recommendation systems, the user-item interaction matrix can be extremely sparse, especially in datasets with a large number of users and items. Removing users with few interactions helps reduce sparsity, making the dataset more manageable and improving the performance of recommendation algorithms.

**Computational Efficiency:** Processing large datasets with many users and items can be computationally expensive. By eliminating users with insufficient interactions, we reduce the size of the dataset, leading to faster computation times during training and recommendation generation.

**Improved Generalization:** Users with very few interactions may exhibit erratic behavior or preferences, making it challenging for the recommendation algorithm to generalize effectively. Focusing on users with a sufficient number of interactions allows the algorithm to learn more robust patterns and preferences, leading to more accurate recommendations for similar users.

**Rank Based Product Recommendation System**

A rank-based product recommendation system relies on the popularity or ranking of items to make recommendations to users. Unlike personalized recommendation systems that tailor suggestions to individual user preferences, rank-based systems prioritize items based on their overall popularity or ratings across all users.

The process involves identifying the top-ranked or most popular items in the dataset and recommending them to users. This can be particularly useful for new users or when there is limited information about user preferences, as it ensures that users are exposed to the most popular and highly-rated items.

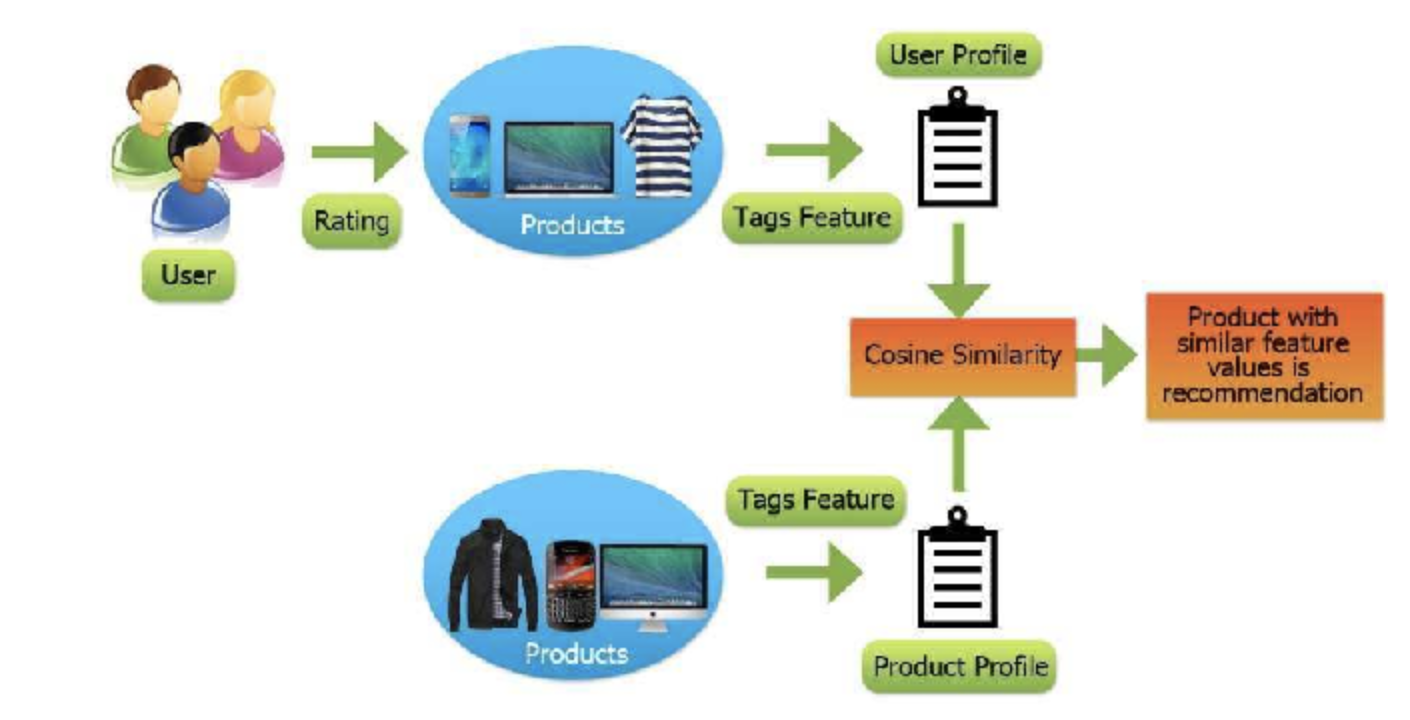
****

Fig 4.0 Rank based recommendation

There are many approaches, one way is to calculate metrics such as average ratings or total number of ratings for each item, and then rank the items accordingly. The top-ranked items are then recommended to users based on their popularity or ranking.

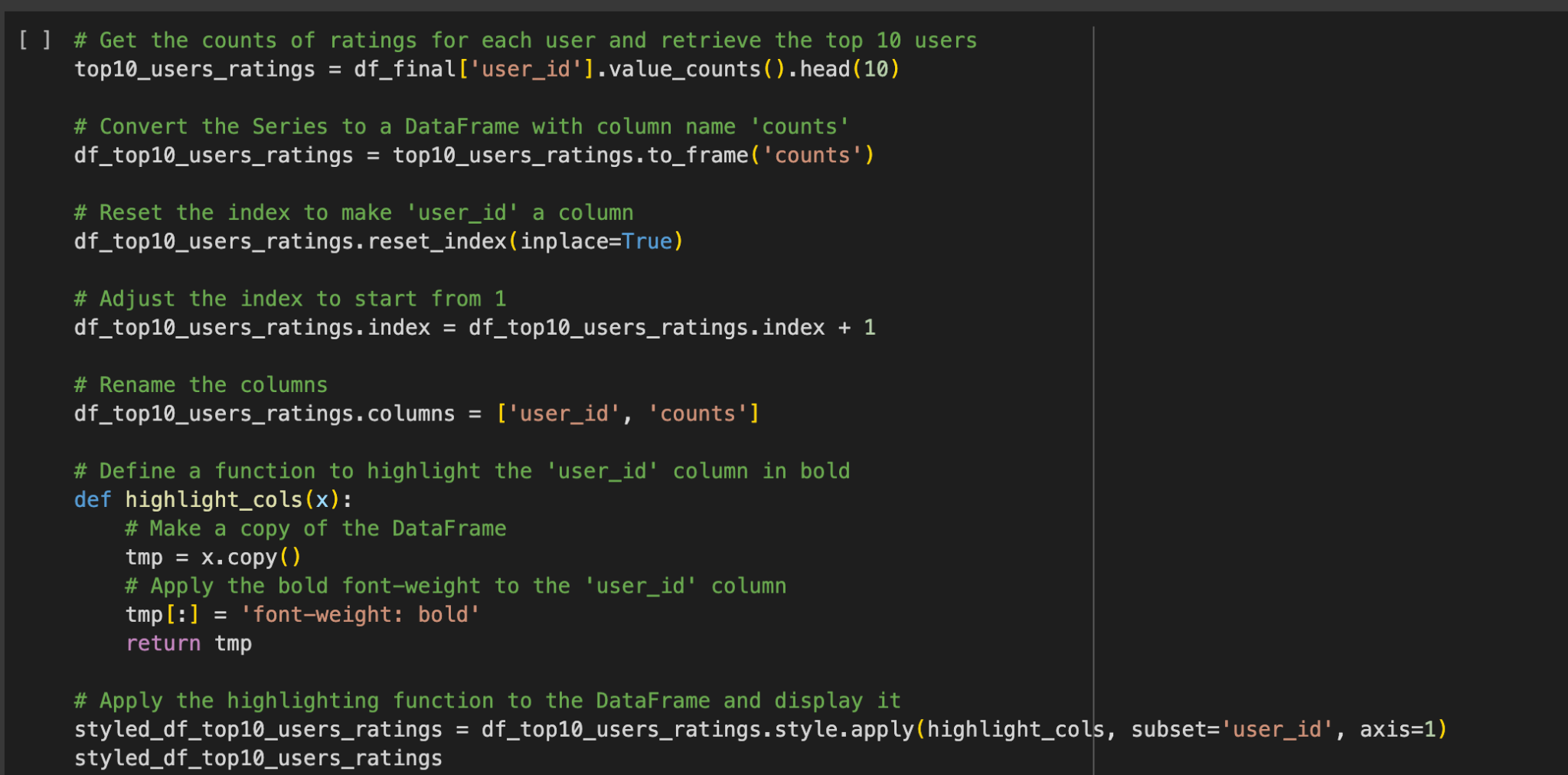


Fig : 4.1 Code snippet for Rank based w.r.t User and Product

Rank-based recommendation systems are simple to implement and computationally efficient, making them suitable for scenarios where personalized recommendations are not feasible or necessary. However, they may not always provide the most relevant recommendations for individual users, especially in cases where user preferences vary widely.

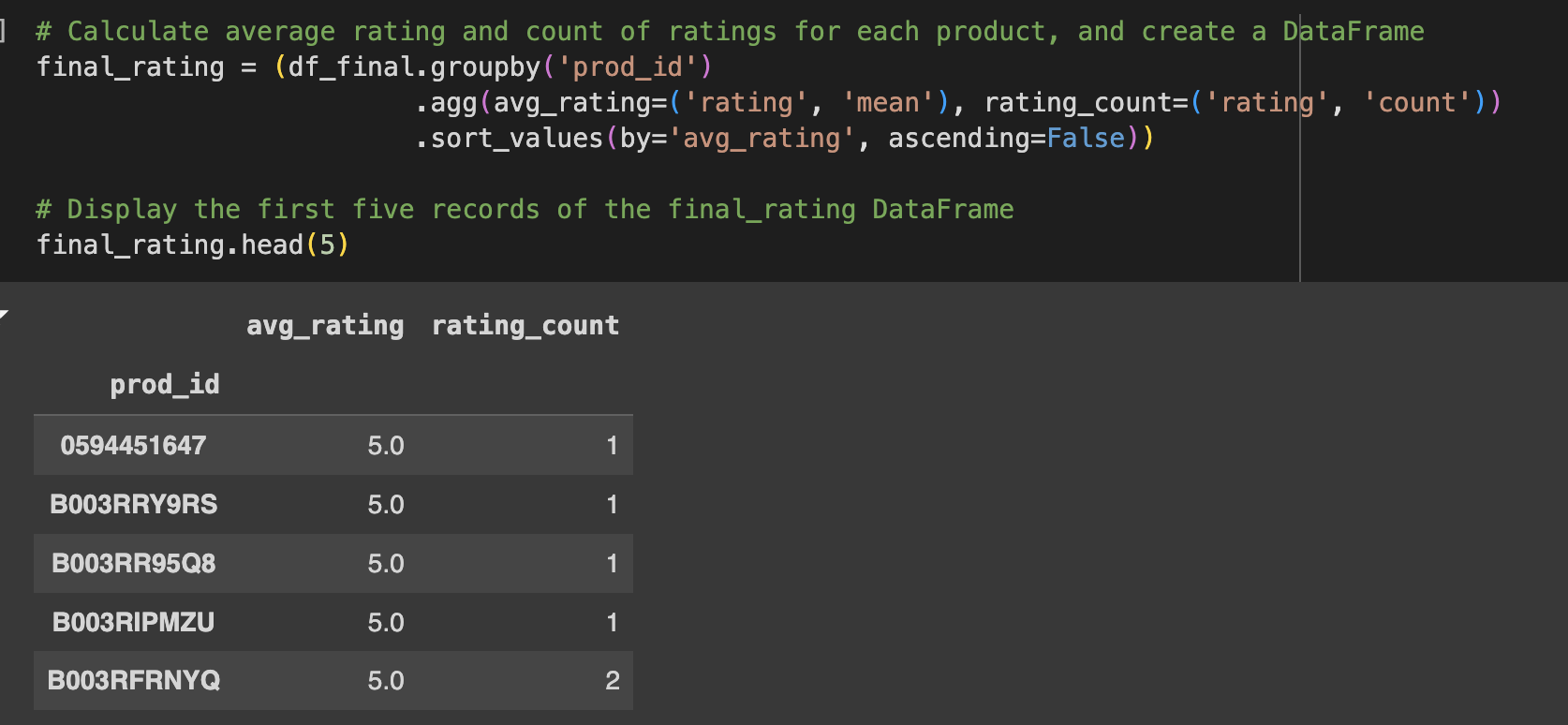


Fig : 4.2 Code snippet for Rank based

Overall, rank-based recommendation systems offer a straightforward way to make recommendations based on the popularity or ranking of items, providing users with a curated list of top-rated products or items.

**User based Collaborative Filtering Recommendation System:**

A user-based product recommendation system focuses on identifying and recommending products to users by analyzing the preferences and behaviors of similar users. The system operates on the principle of collaborative filtering, where it assumes that users who have agreed on past products will likely agree on future ones as well. The recommendation process begins by collecting interaction data, such as purchase history, ratings, or browsing patterns, from a group of users. This data is then used to calculate the similarity between users, often using metrics like cosine similarity or Pearson correlation.

Once similarities are determined, the system identifies a target user's nearest neighbors—those users with the highest similarity scores. The preferences of these neighbors are aggregated to generate a list of recommended products. If the target user has not rated a particular product but their neighbors have, the system can infer that the target user might also like that product. This method is effective because it leverages the collective experience and preferences of a community of users to make personalized recommendations.

However, user-based collaborative filtering faces challenges such as scalability and sparsity. As the number of users increases, the computation of similarities can become intensive. Additionally, if users have interacted with only a few products, the system might struggle to find significant overlaps in preferences. Despite these challenges, user-based recommendation systems are widely used in e-commerce and streaming services, where they help enhance user experience by suggesting relevant products based on the behavior of similar users.

****

Fig:4.3 User based Recommendation System

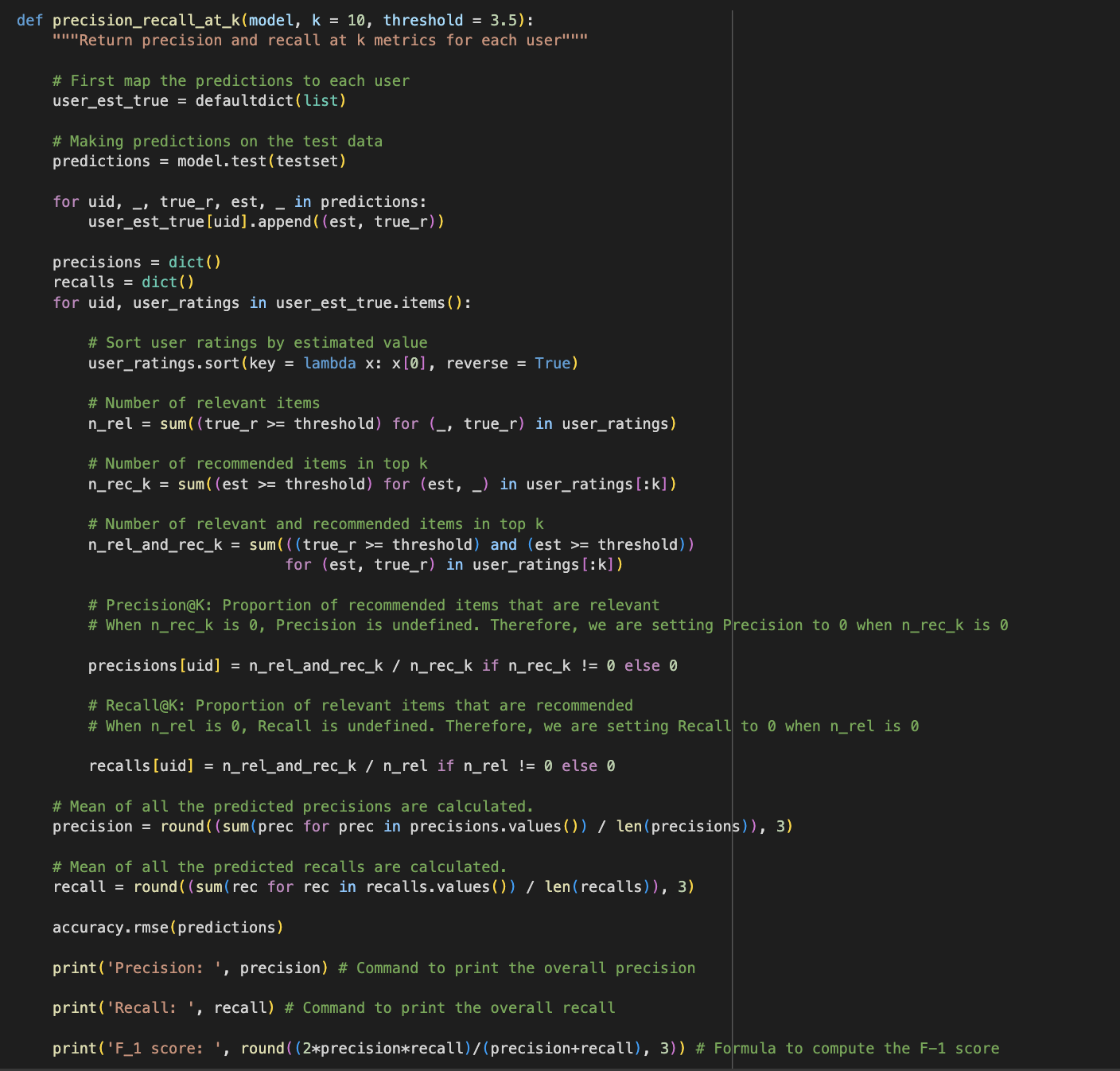
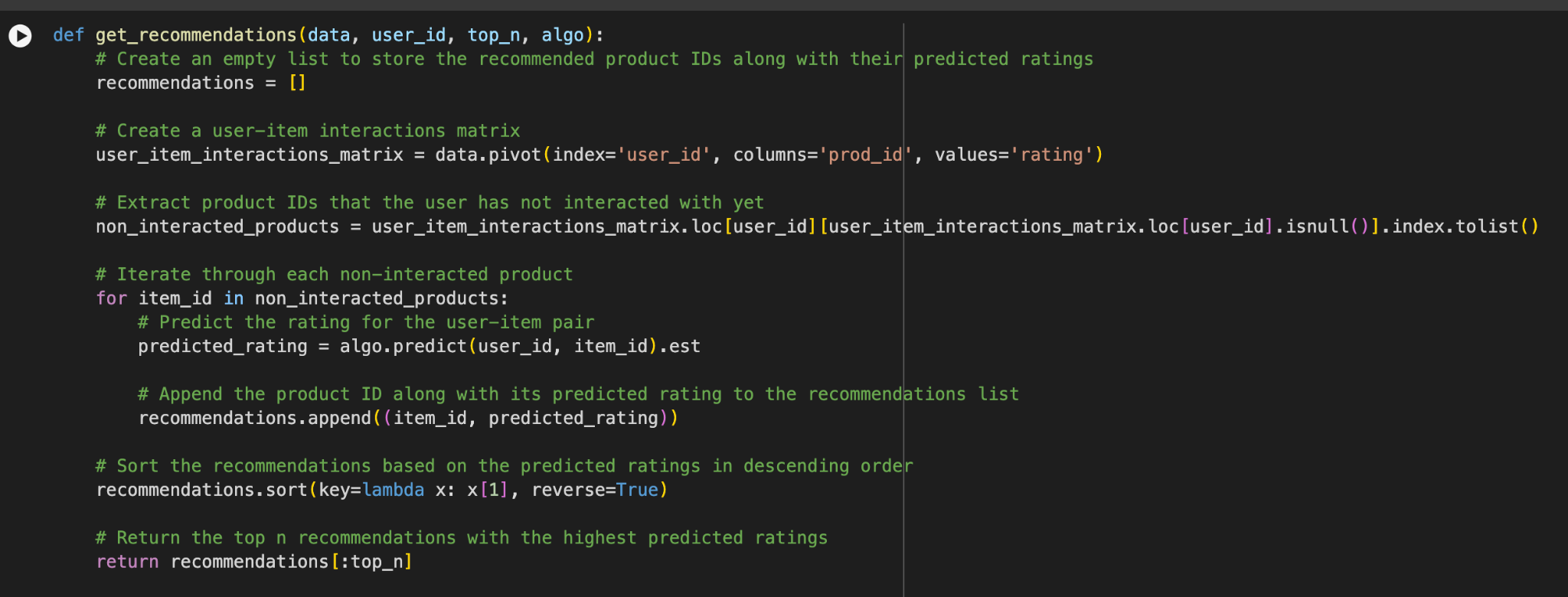


Fig: 4.4 Code snippet for User Based Recommendation

To enhance the performance of our similarity-based recommendation system, we'll fine-tune the hyperparameters of the KNNBasic algorithm. Key hyperparameters include(k), which specifies the maximum number of neighbors used for recommendations, with a default of 40, and (min\_k), setting the minimum number of neighbors required for making a prediction, defaulting to 1. If there aren't enough neighbors, the prediction defaults to the global mean of all ratings. Additionally, we'll explore the `sim\_options` dictionary, which contains options for the similarity measure: cosine similarity, mean squared difference (msd), Pearson correlation, and Pearson baseline. By experimenting with different values for these parameters, we aim to optimize the recommendation system's performance, ensuring it provides more accurate and relevant suggestions tailored to our specific data set and use case.

 Fig:4.6 Code snippet for the KNN user-user similarity model

**Model-Based Collaborative Filtering:**

A model-based product recommendation system that utilizes a sigma-based matrix approach involves constructing a user-item interaction matrix where each cell represents the strength of the interaction between a user and an item. The sigma-based matrix approach incorporates additional factors or features beyond simple user-item interactions to enhance recommendation accuracy.

To implement this approach, we start by defining a feature space that captures relevant information about users and items. This feature space could include attributes such as user demographics, item attributes, contextual information, and past behavior. These features are then used to construct a feature matrix for both users and items.

Next, we compute a similarity matrix between users and items based on the feature representations. This similarity matrix quantifies the similarity between users or items in the feature space. Techniques such as cosine similarity or Pearson correlation coefficient can be used to compute similarities between feature vectors.

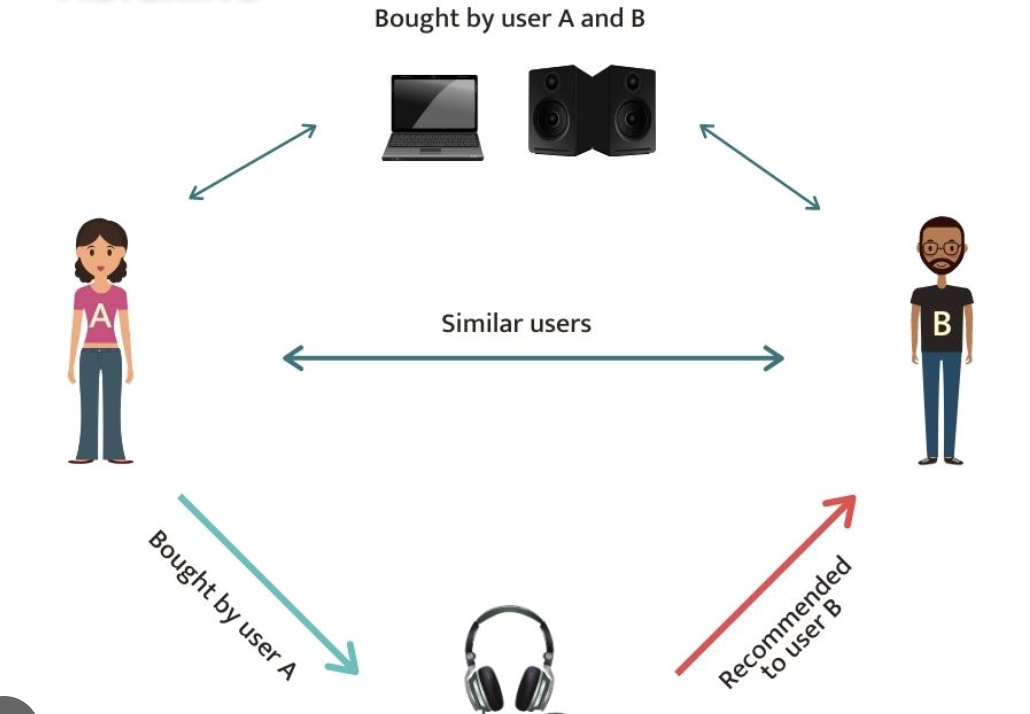


Fig 4.8 Model based Product Recommendation

Once the similarity matrices are computed, we can apply a sigma-based matrix factorization technique to decompose the user-item interaction matrix into lower-dimensional matrices while preserving important features. The sigma-based factorization method incorporates the similarity matrices into the factorization process, allowing the model to capture more nuanced relationships between users and items.

Finally, we use the factorized matrices to make predictions for user-item interactions. By reconstructing the user-item interaction matrix using the factorized matrices, we can predict the likelihood of user-item interactions for unseen data points. These predictions form the basis for generating personalized recommendations for users.

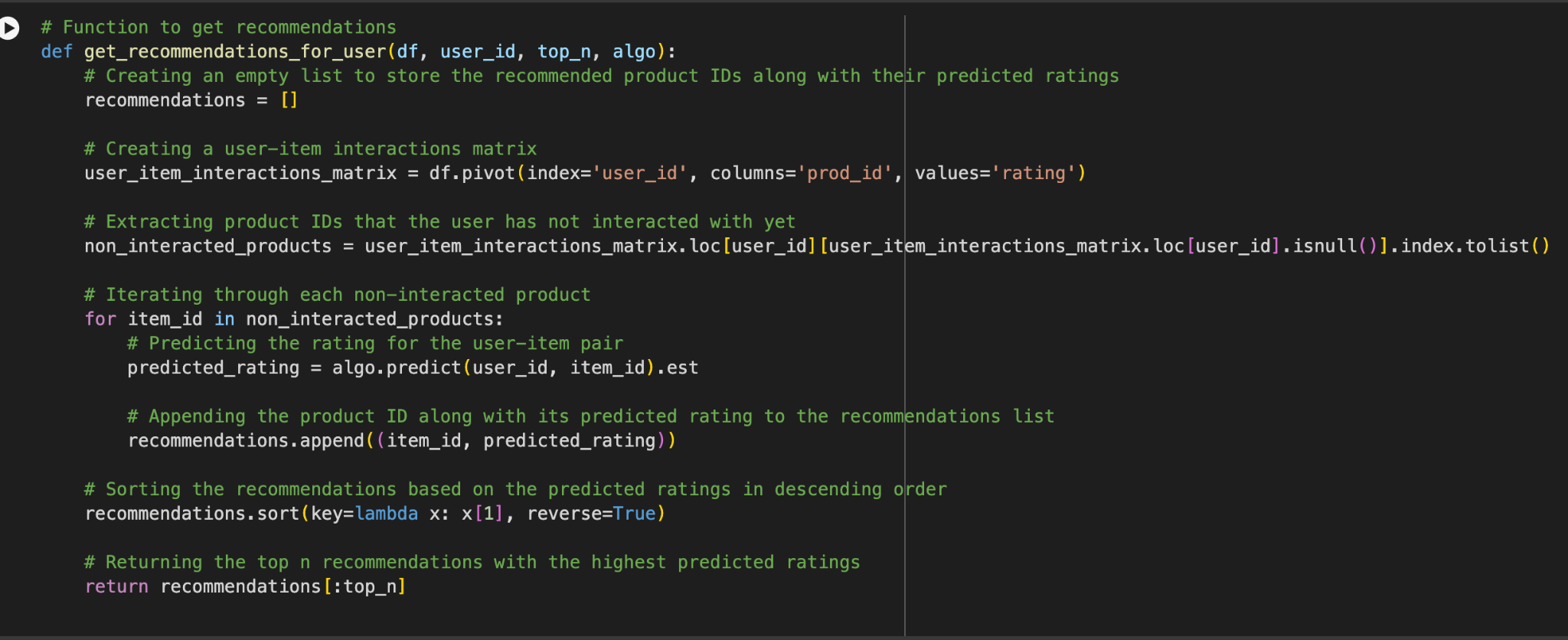


Fig 4.9 Code Snippet Model based Product Recommendation

Overall, the sigma-based matrix approach enhances the recommendation accuracy by incorporating additional features and leveraging matrix factorization techniques. This approach allows the model to capture complex relationships between users and items, leading to more accurate and personalized recommendations.To improve a matrix factorization-based recommendation system by tuning its hyperparameters, we focus on optimizing three key parameters: n\_epochs (the number of iterations for the stochastic gradient descent algorithm), lr\_all (the learning rate for all parameters), and reg\_all (the regularization term for all parameters). First, we load the dataset and split it into training and test sets. We define a grid of hyperparameters, specifying ranges for n\_epochs, lr\_all, and reg\_all. Using GridSearchCV, we perform a grid search with cross-validation to find the best combination of hyperparameters based on RMSE and MAE metrics. After identifying the optimal parameters, we train the SVD model with these settings. We then evaluate the model on the test set by making predictions and calculating the RMSE to measure its performance. This process ensures that our recommendation system is fine-tuned for better accuracy and generalization, leveraging the optimized hyperparameters to enhance prediction quality and user satisfaction.

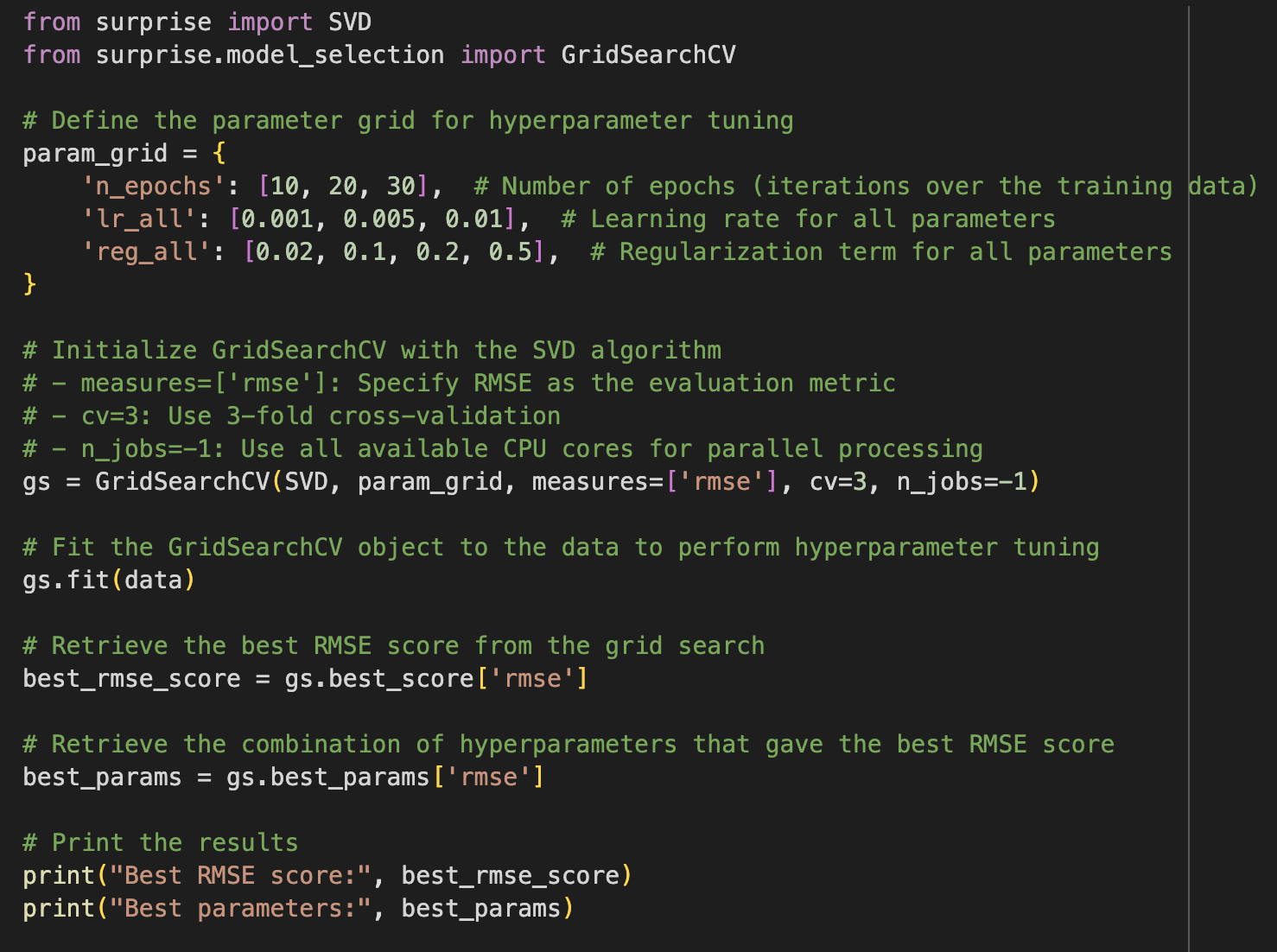
****

Fig 4.9 Precision, Recall, F\_1 score Values for Model

**Data Processing**

* Data processing in machine learning involves a series of essential steps to prepare raw data for effective model training. Initially, data is collected from diverse sources such as databases, files, APIs, or web scraping. Following this, thorough data cleaning procedures are applied to handle missing values, remove duplicates, and rectify any errors, ensuring the dataset's reliability.
* Subsequently, feature selection or extraction techniques are employed to identify and retain relevant features that strongly correlate with the target variable, either through domain expertise or automated methods.
* Once features are selected, categorical data is converted into numerical representations using methods like one-hot encoding or label encoding, while numerical features are scaled to maintain consistent ranges across the dataset, often through normalization or standardization techniques.
* Additionally, the dataset is partitioned into training, validation, and test sets to facilitate model evaluation and tuning.
* Optional steps such as data augmentation, preprocessing specific to the chosen algorithm, normalization, imputation of missing values, encoding of categorical variables, and balancing of class distributions may also be performed, depending on the nature of the dataset and the machine learning task at hand.
* Finally, data visualization techniques can be employed to gain insights into the dataset's distribution, feature relationships, and potential patterns, aiding in informed decision-making throughout the preprocessing phase.
* Through meticulous execution of these data processing steps, raw data is transformed into a clean, structured, and informative dataset, laying the groundwork for robust machine learning model training and performance evaluation.

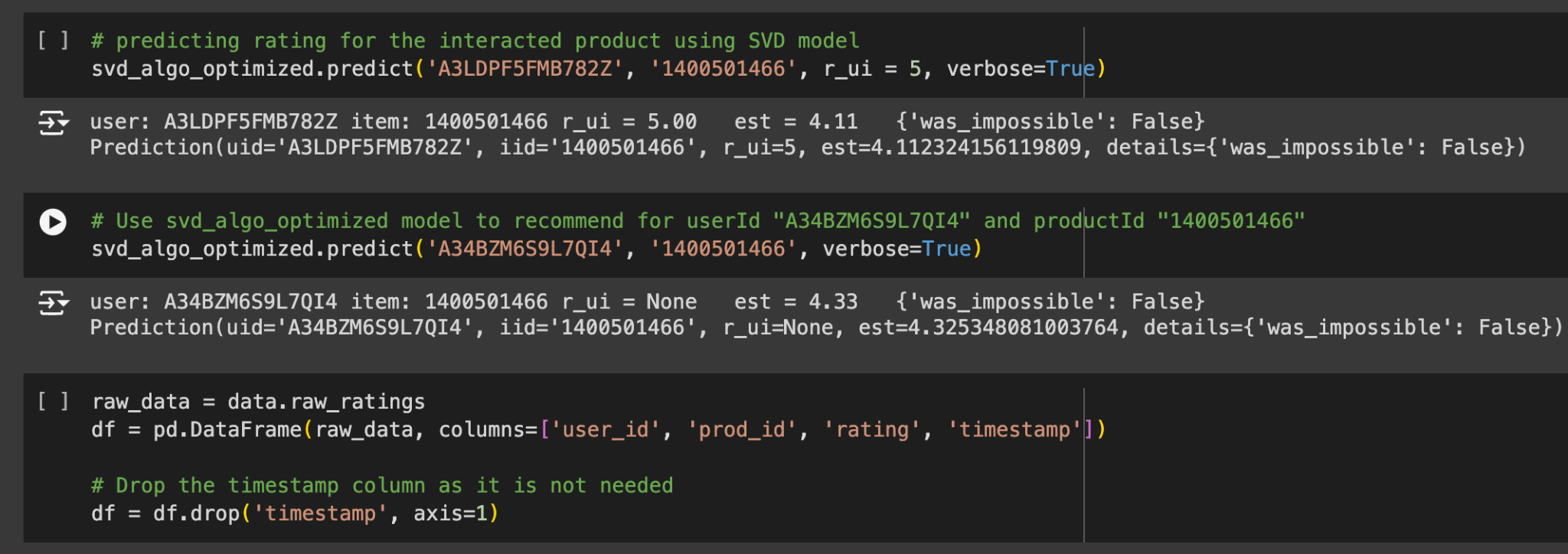


Fig 5.0 Data processing

## **Advanced Data Processing Techniques**

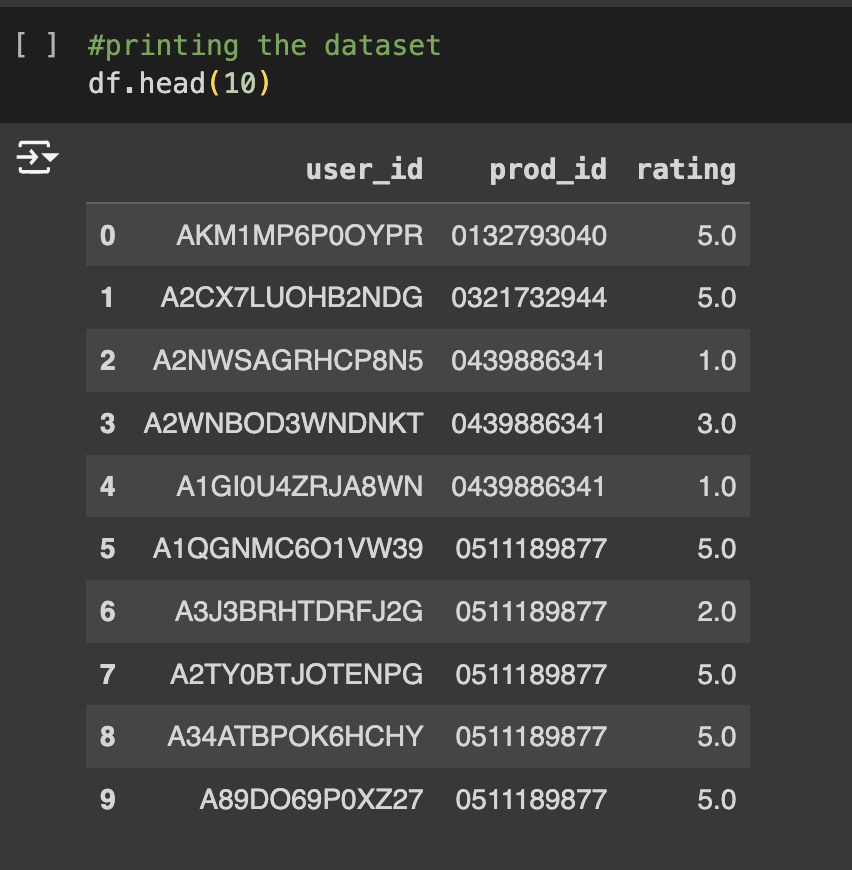
1. The process of converting a Python generator into a tf.data.Dataset involves using the Dataset.from\_generator constructor. This method requires a callable function that can restart the generator after each cycle. It's crucial to specify output\_types to ensure proper construction of the internal TensorFlow graph, as each graph edge requires a defined tf.dtype.
2. For iterative data processing, the Dataset.repeat() function is employed. When used without parameters, this function enables indefinite repetition of the input dataset, allowing for multiple epochs of iteration.
3. To introduce randomness into the dataset, the Dataset.shuffle() method is utilized. This function maintains a fixed-size buffer and randomly selects subsequent elements, thereby enhancing the model's exposure to diverse data patterns. However, it's important to note that shuffling is typically applied to the training dataset only.
4. Batch processing and epoch management are essential aspects of training a machine learning model. To maintain distinct epoch boundaries, Dataset.batch is placed strategically before Dataset.repeat(). This ensures that data is batched together before being repeated, preventing intermingling of data across epoch boundaries.
5. Prefetching is a technique used to streamline data processing and model execution. The tf.data.Dataset.prefetch transformation overlaps the data preprocessing phase with model execution. This means that while the model processes a specific step, the pipeline concurrently prepares data for the subsequent step. Prefetching reduces overall step time and enhances processing efficiency. The number of elements to prefetch should ideally match or exceed the batch count per training step. Additionally, the tf.data.AUTOTUNE option can be used for dynamic runtime adjustment, optimizing prefetching for the specific hardware and workload.

# **Execute**

* The recommend\_items function is a Python utility designed to provide personalized recommendations for a specific user within a collaborative filtering framework. It operates by first retrieving the user's actual ratings from an interactions matrix, which captures the historical interactions between users and items.
* Subsequently, it accesses the predicted ratings for the same user from a predictions matrix, generated by a recommendation model. These ratings are then consolidated into a DataFrame, allowing for easy manipulation and analysis. Following this, the function assigns unique indices to each product in the DataFrame, facilitating subsequent operations. It filters out products that the user has already rated to ensure that only unrated items are considered for recommendation.
* The remaining products are sorted based on their predicted ratings in descending order, prioritizing items with higher predicted ratings.
* Finally, the function outputs the top-N recommended products for the user, along with their corresponding predicted ratings. This approach enables the generation of tailored recommendations that reflect the user's preferences and interests, contributing to an enhanced user experience in recommendation systems.

# **Result And Outcome**

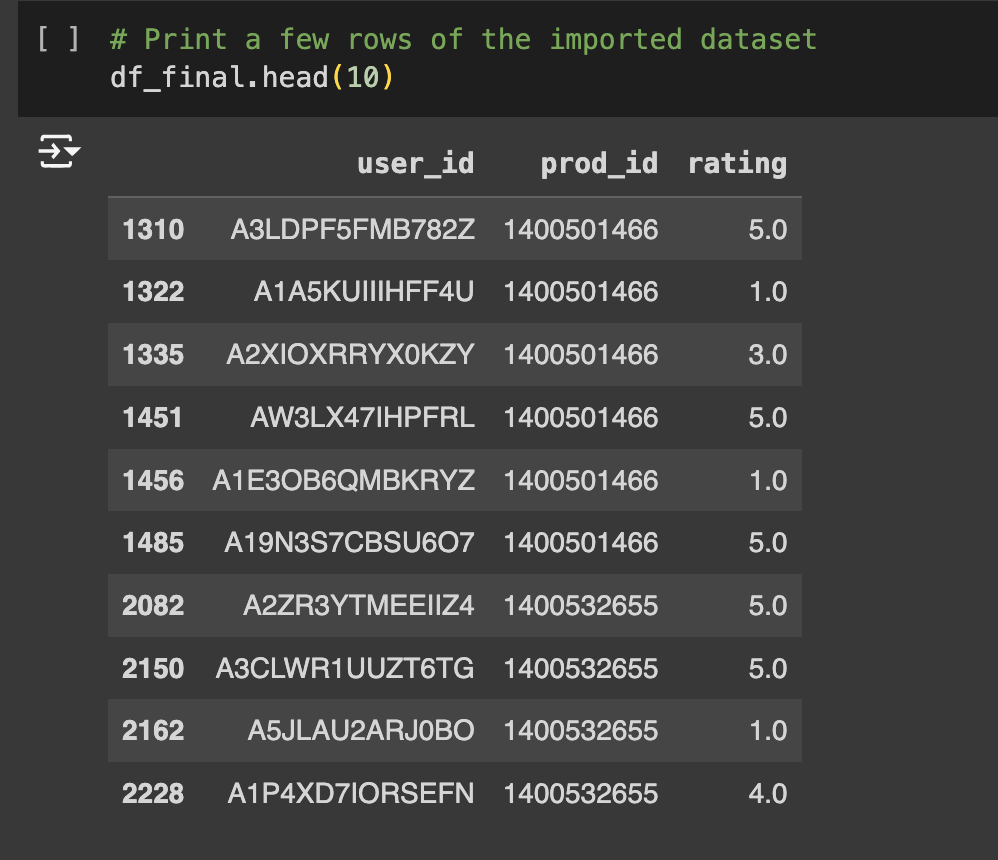
**Raw Data set:**

The provided data represents a user-item interaction dataset commonly used in recommendation systems. Each row corresponds to a user's rating for a particular product, identified by unique user and product IDs. The 'user\_id' column identifies the user who provided the rating, while the 'prod\_id' column specifies the product being rated. The 'rating' column indicates the user's rating for the respective product, typically on a scale from 1 to 5, where 1 denotes the lowest rating and 5 represents the highest. This dataset captures user preferences and interactions, forming the basis for building recommendation models that predict user preferences for unrated items and generate personalized recommendations accordingly.The data before processing the sets. ****

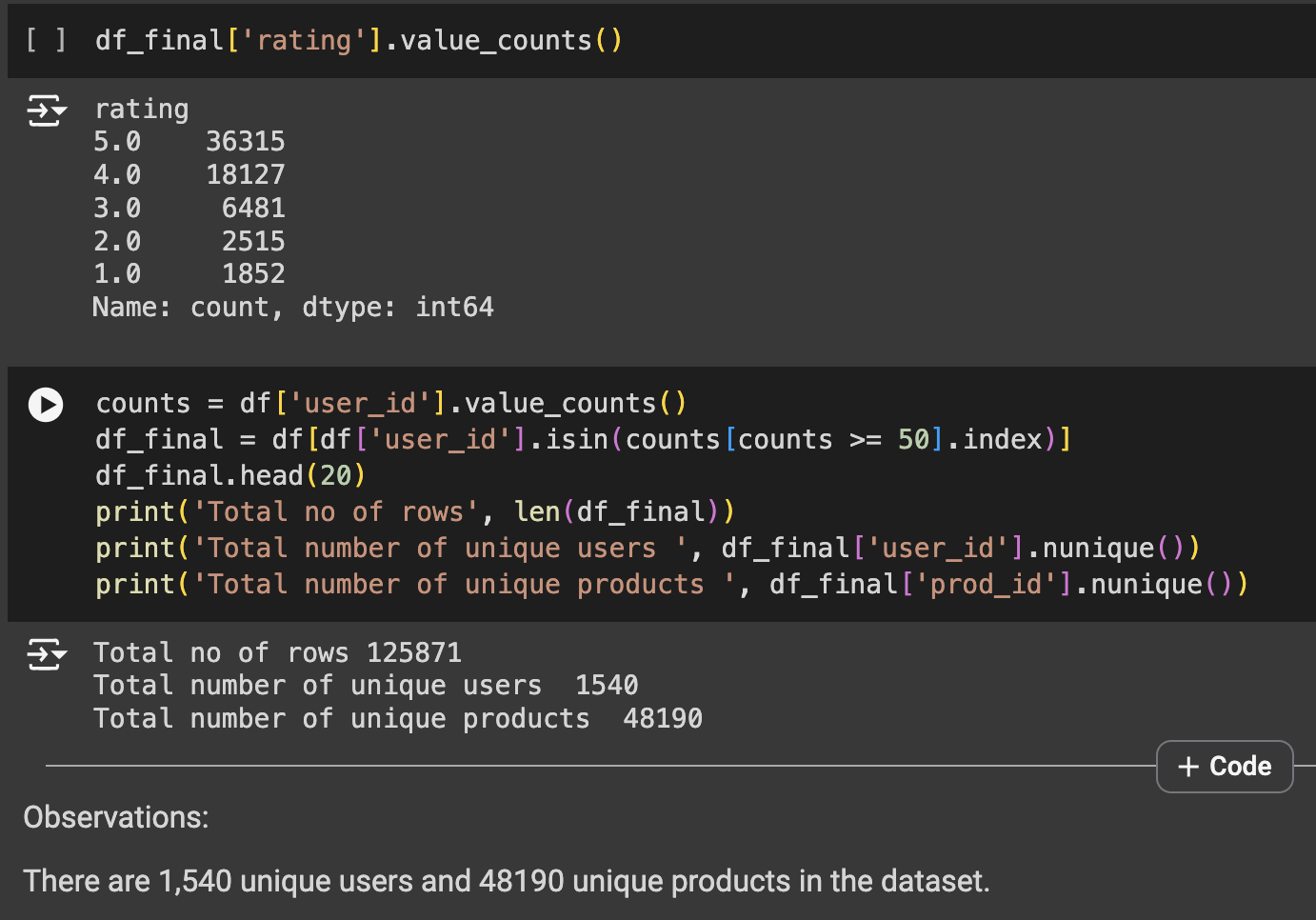
**Fig:6.0 Raw dataset**

**Training Data set:**

The interaction matrix provided has dimensions of 1540 rows by 48190 columns, representing users and products, respectively. Among the 1540 users, a total of 125871 ratings have been given, indicating the number of explicit interactions captured within the dataset. Despite the substantial number of given ratings, the total possible number of ratings stands at 74212600, suggesting a sparse matrix where many potential user-item interactions remain unobserved. With a density of 0.17%, the matrix exhibits a relatively low percentage of filled entries compared to its total capacity, highlighting the sparsity inherent in recommendation datasets and the prevalence of missing ratings. Each cell in the interaction matrix corresponds to a user's rating for a specific product, with a rating of '0.0' indicating that the user has not rated the corresponding product. This structured representation facilitates the analysis and modeling of user preferences and item recommendations.The Training data of the sets are as below:



**Fig:6.1 Trained Data set**

****

**Fig:6.2 Final predictions of users and products**

**Rank Based Product Recommendation System:**

Rank-based recommendations output is typically generated by sorting the items based on a predetermined criterion, such as the number of ratings or average rating score. For example, in a simple rank-based recommendation system, the output might consist of the top N items with the highest number of ratings. Alternatively, it could include the top N items with the highest average rating scores. These recommendations are presented to the user in descending order, with the most highly-ranked items displayed first. The output format may vary depending on the application, but it commonly includes the product ID, name, image, and other relevant details to assist users in making informed decisions.This data provides valuable insights into the distribution of ratings across different products, aiding in understanding user preferences and identifying popular or highly-rated items within the dataset. The top 5 output are as follows:

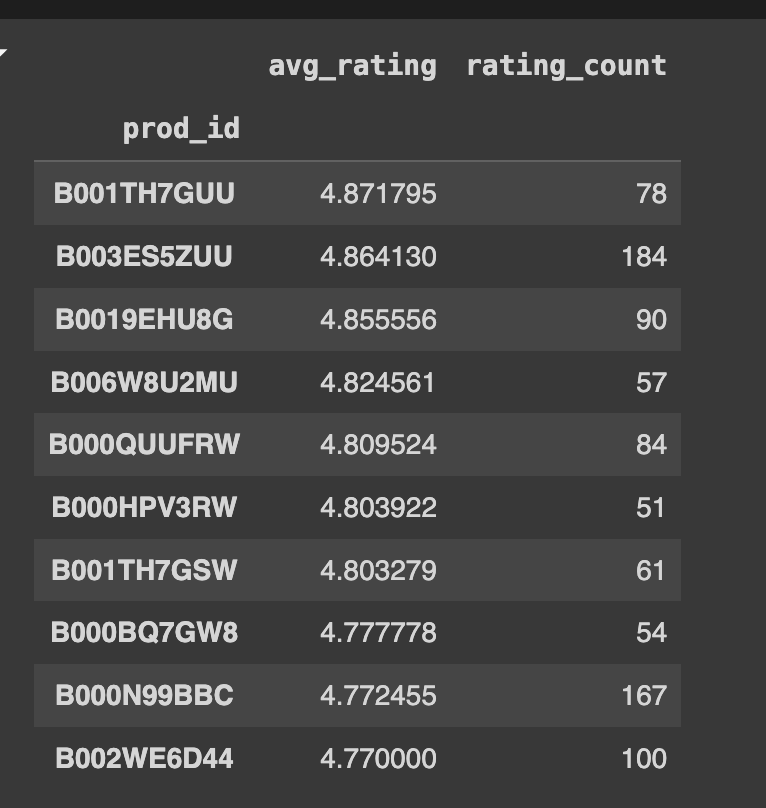


Fig : 6.3 Top 10 recommendations based on rank

**User based Product Recommendations System:**

The output of the `recommend\_items` function is a list of recommended products for a given user, along with their predicted ratings. The function takes several inputs, including the user's index, the interaction matrix capturing user-product interactions, the matrix of predicted ratings, and the number of recommendations to generate (`num\_recommendations`).

The function first extracts the user's actual ratings and predicted ratings from the interaction and prediction matrices, respectively. Then, it creates a DataFrame containing these ratings along with the corresponding product indices.

Next, the function filters the DataFrame to include only products that the user has not yet rated (i.e., products with a rating of 0).

Subsequently, it sorts the filtered DataFrame based on the predicted ratings in descending order, ensuring that the highest-rated products are recommended first.

Finally, the function prints the top-n recommended products for the user, along with their predicted ratings.

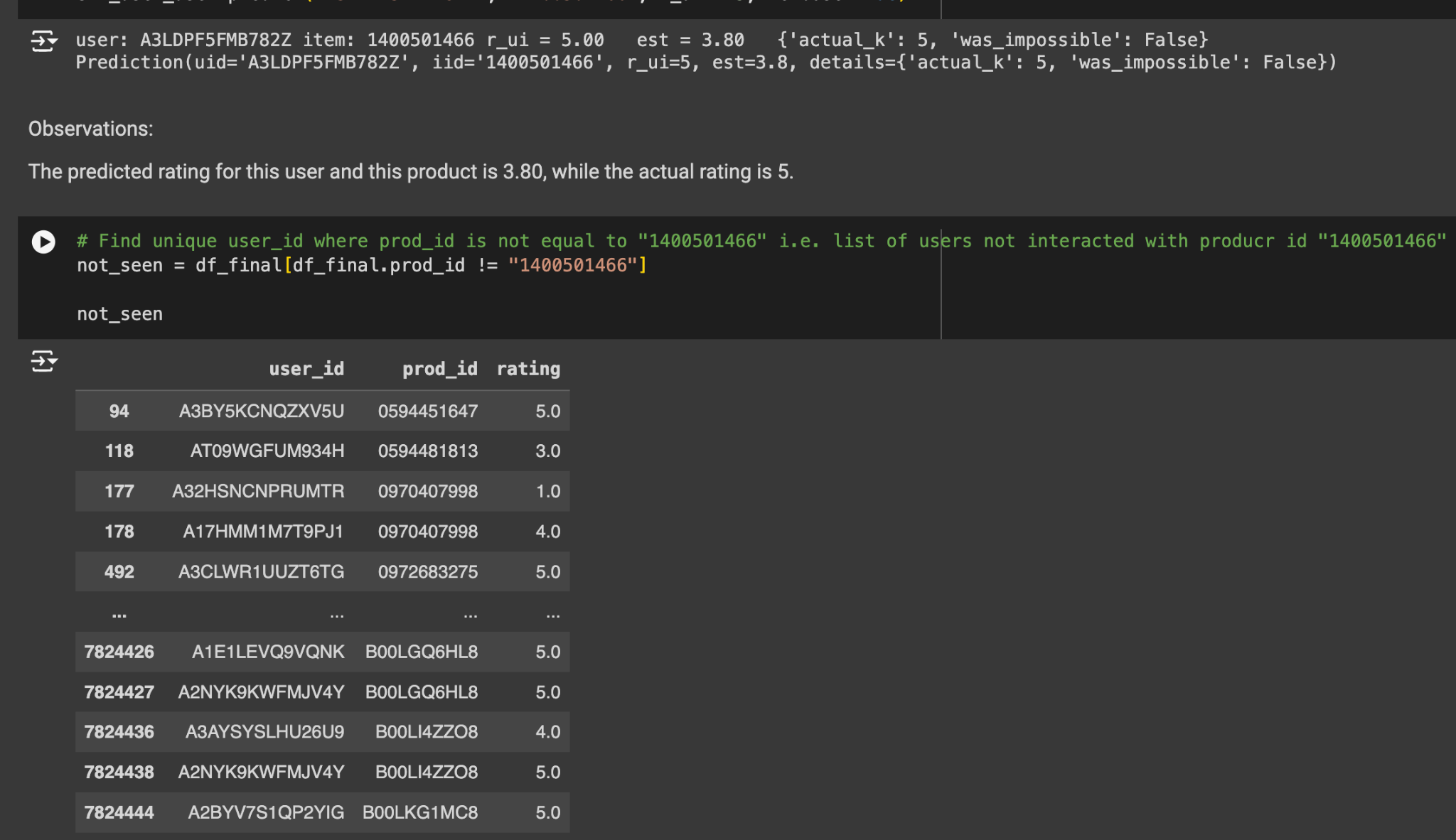


Fig:6.4 The final recommendations for specific user



Fig: 6.5 user-user similarity model based recommendations

The output provides the recommended products along with their predicted ratings, enabling the user to make informed decisions based on their preferences and the system's predictions.

**Model Based Product Recommendation System:**

The output of a model-based product recommendation system typically consists of personalized recommendations for individual users. These recommendations are generated based on the user's historical interactions, preferences, and behavior captured in the system. The output includes a list of recommended products ranked according to their predicted relevance or likelihood of being appreciated by the user. Additionally, the output may provide insights into the reasoning behind each recommendation, such as the similarity of the recommended products to those previously interacted with by the user or the underlying features driving the recommendation. The goal is to offer tailored suggestions that align with the user's interests and needs, ultimately enhancing their shopping experience and increasing the likelihood of engagement and satisfaction. The top recommendations are as follows:

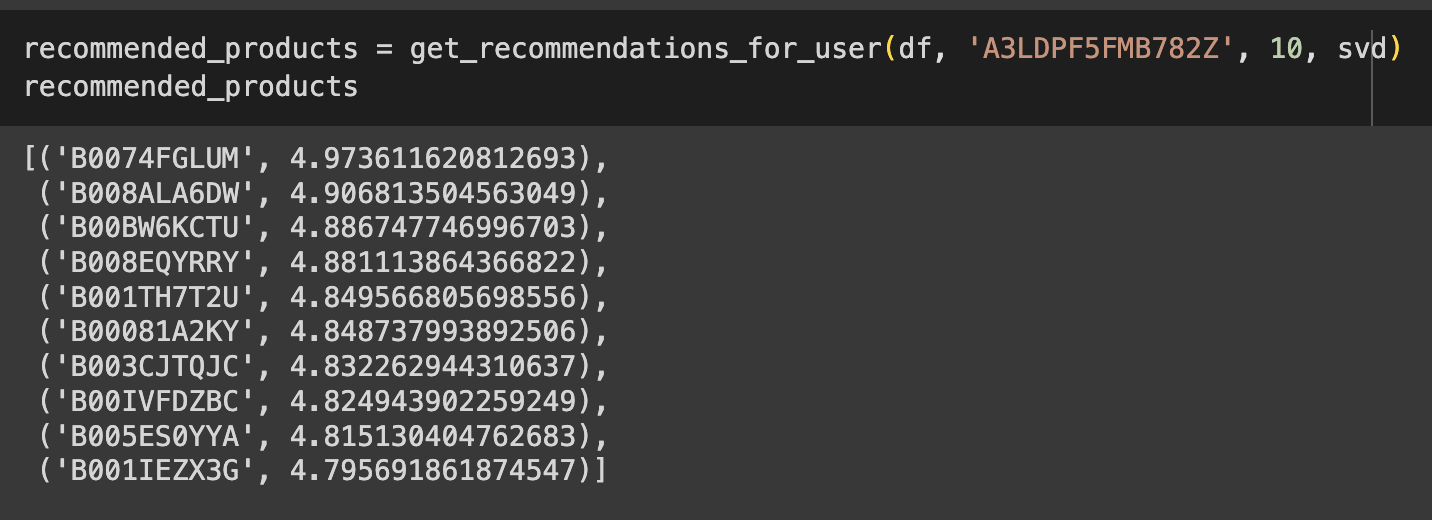


Fig 6.6: Model Based Recommendations

**Theoretical and Experimental Results:**

**Avg\_actual\_ratings vs Avg\_predicted\_ratings**

**Average Actual Ratings**: This metric represents the average ratings given by users to the products in the dataset. It reflects the overall satisfaction level of users with the items they have interacted with. A higher average actual rating suggests that users tend to rate products positively, indicating satisfaction with their purchases or interactions.

**Average Predicted Ratings:** On the other hand, average predicted ratings represent the average ratings predicted by the recommendation system for all products. These ratings are generated based on user behavior, historical data, and the recommendation algorithm used. A higher average predicted rating indicates that the recommendation system is suggesting products that it believes users will rate positively.

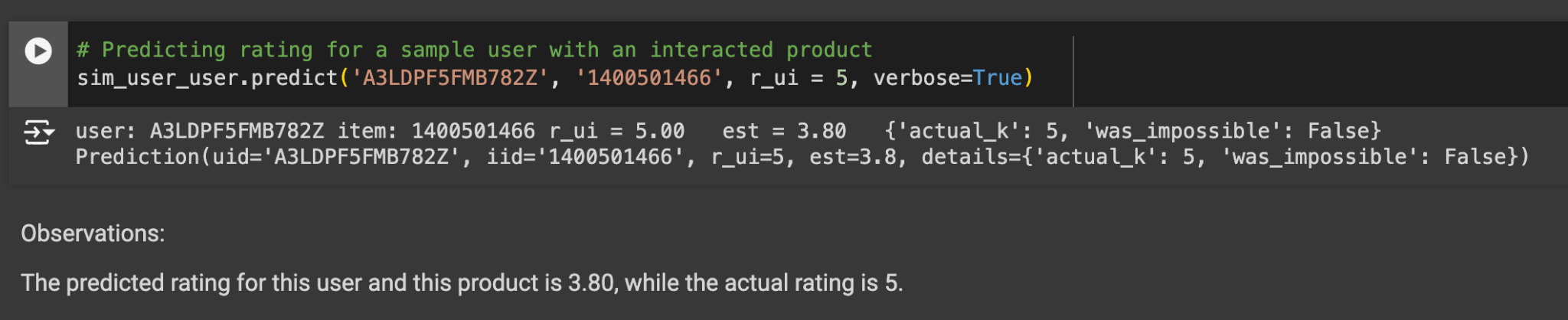


Fig :7.0 Results of actual vs predicted Ratings for user based

**The ROC & Precision-recall curve:**

The precision\_recall\_at\_k function calculates precision and recall at 𝑘 for a given recommendation model and dataset, followed by computing the F1 score. Initially, the function collects predictions for the test dataset, mapping these predictions to each user. For each user, it records the estimated and true ratings of the items they interacted with. The function then sorts these user ratings by the estimated values in descending order. It calculates the number of relevant items (where the true rating is above a specified threshold), the number of recommended items in the top 𝑘 (where the estimated rating is above the threshold), and the number of items that are both relevant and recommended in the top k. Precision@K is determined as the proportion of recommended items that are relevant, while Recall@K is the proportion of relevant items that are recommended. The function then averages the precision and recall values across all users to obtain overall precision and recall metrics. Additionally, it calculates the Root Mean Square Error (RMSE) to evaluate prediction accuracy. Finally, it computes the F1 score, a harmonic mean of precision and recall, which balances the trade-off between these two metrics, providing a single measure that accounts for both false positives and false negatives. The results are printed, giving a comprehensive overview of the model’s performance**.**

**For User based:**

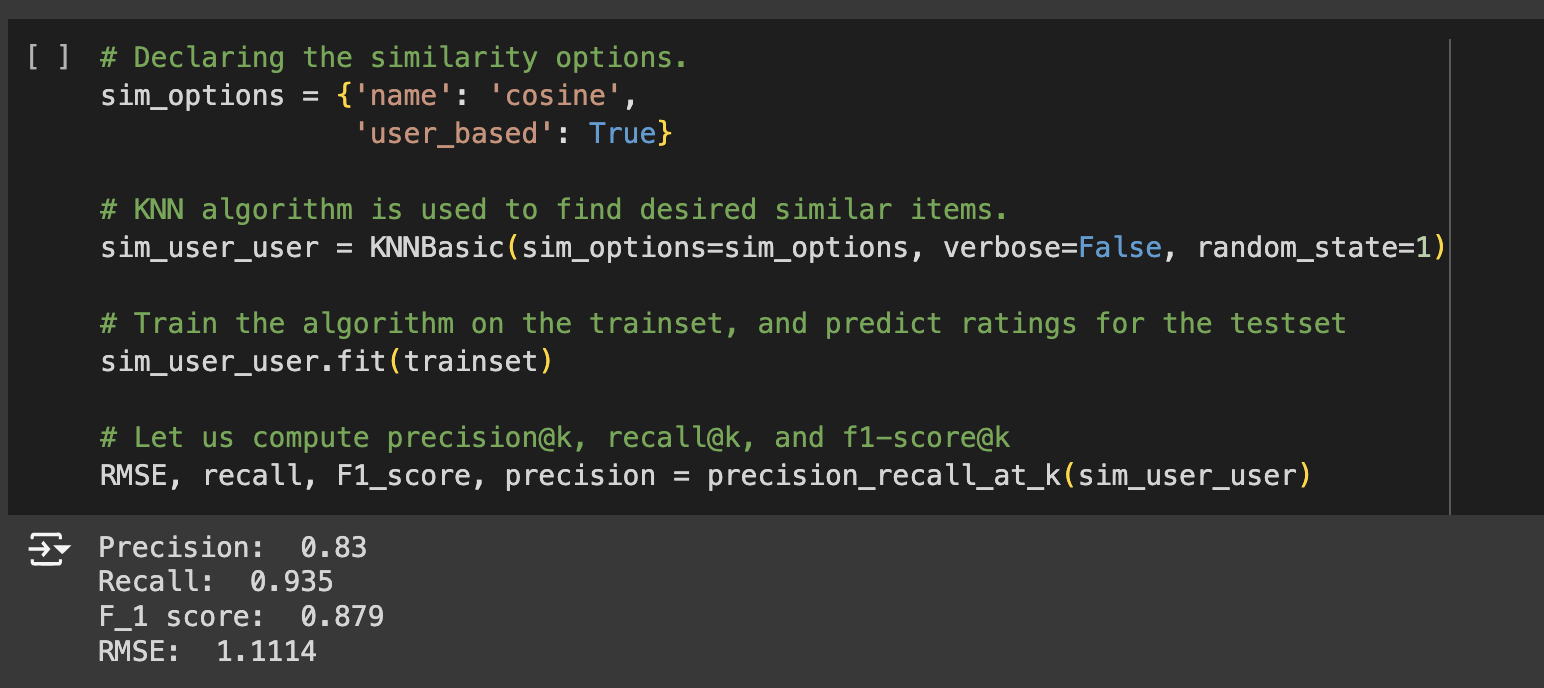
****

Fig: 7.1 Precision vs Recall Before Optimization for user based

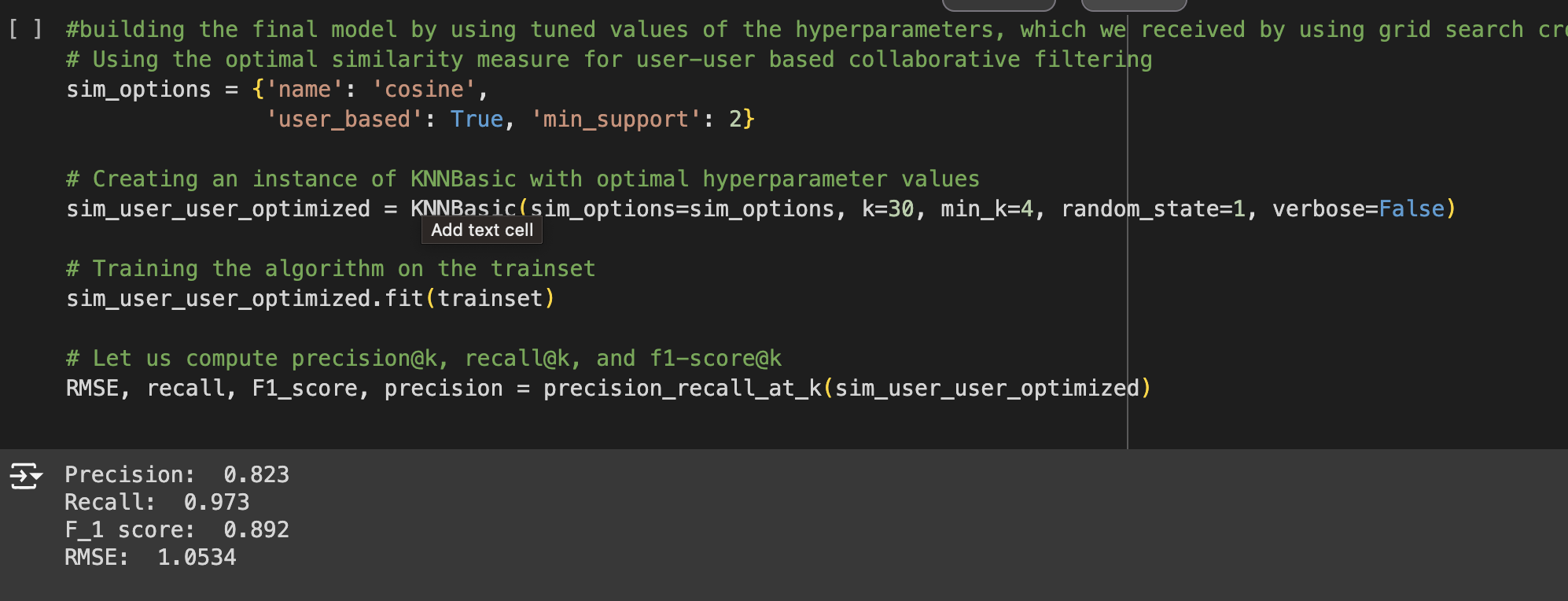
****

Fig : 7.2 Precision vs Recall After Optimization for user based

The graphical representation Precision vs Recall Before & After Optimization for User based:

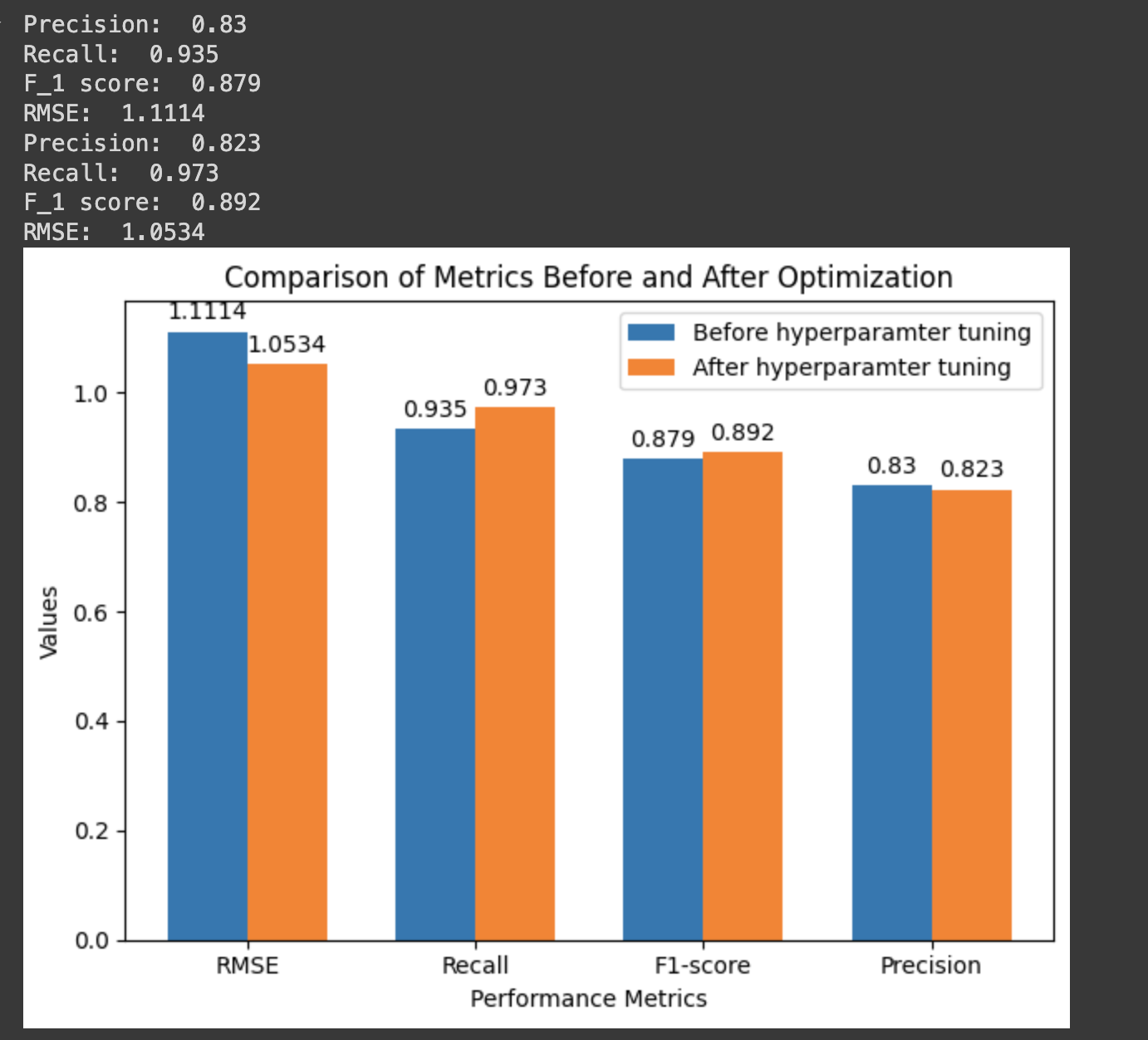


Fig:7.3 Graphical representation of performance metrics for user based

To evaluate the performance of our similarity-based recommendation system, we can use the ROC curve and the precision-recall curve. The ROC curve (Receiver Operating Characteristic) plots the true positive rate (recall) against the false positive rate, providing insights into the model's ability to distinguish between relevant and non-relevant items at various threshold settings. The area under the ROC curve (AUC-ROC) summarizes the model's performance across all thresholds, where a value of 1 represents a perfect model and 0.5 indicates random guessing. The precision-recall curve, particularly useful for imbalanced datasets, plots precision (the ratio of true positives to the total number of positive predictions) against recall. The area under the precision-recall curve (AUC-PR) provides a summary of the trade-off between precision and recall across different thresholds. High values in both AUC-ROC and AUC-PR indicate a model that makes accurate and comprehensive recommendations. Using these metrics, we can gain a deeper understanding of our recommendation system's performance and make necessary adjustments to improve it.

The graphical representation of ROC is as follows:

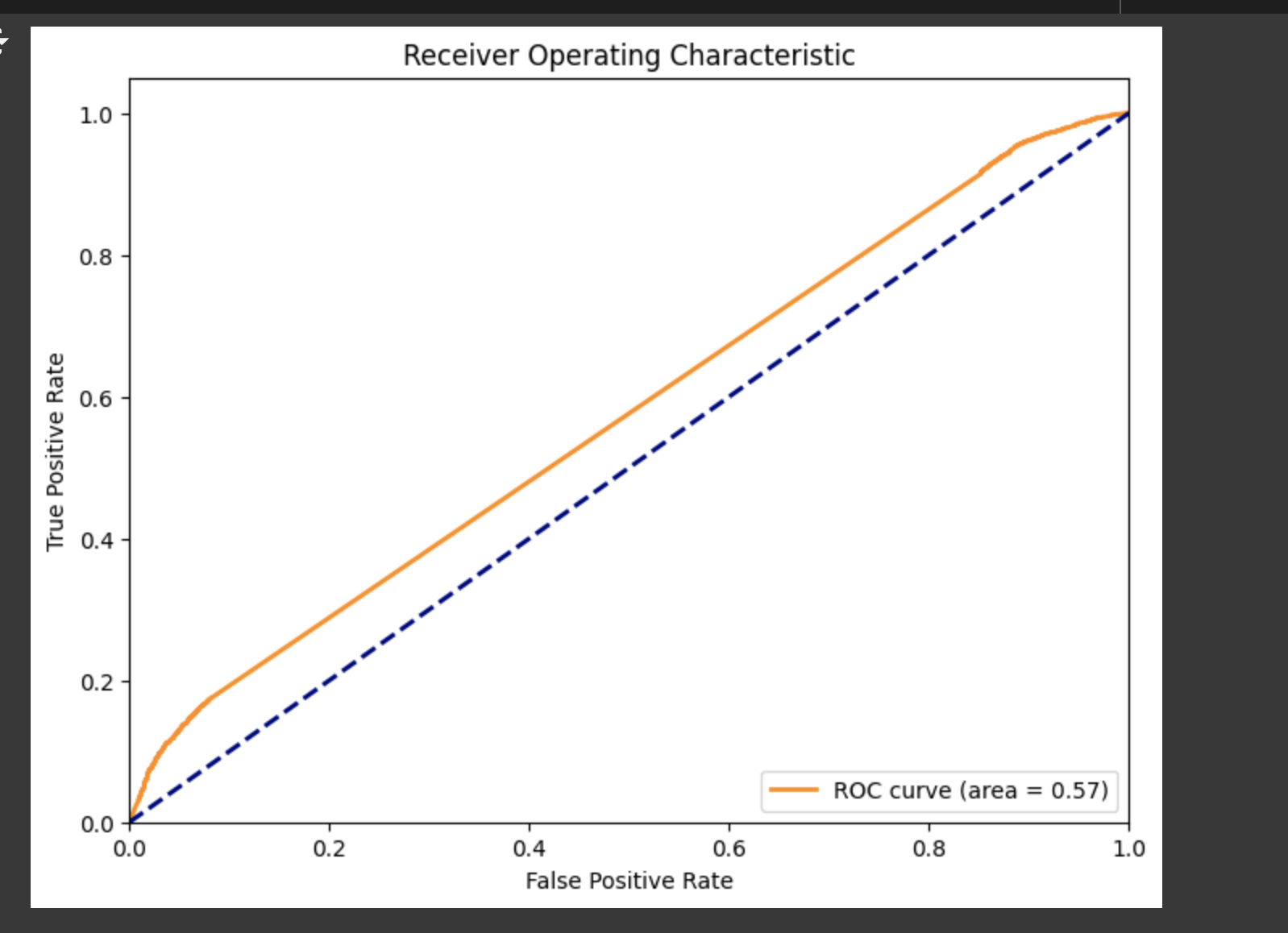


Fig:7.4 Roc Curve

**For Model based:**

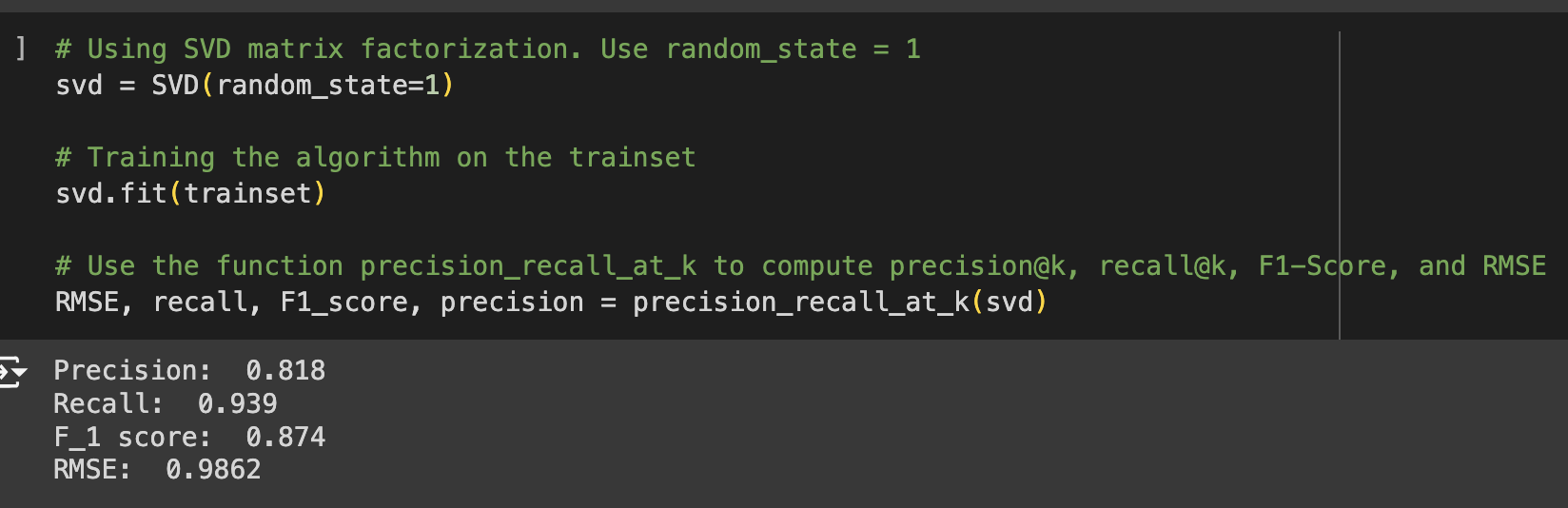


Fig: 7.5 Precision vs Recall Before Optimization for Model based

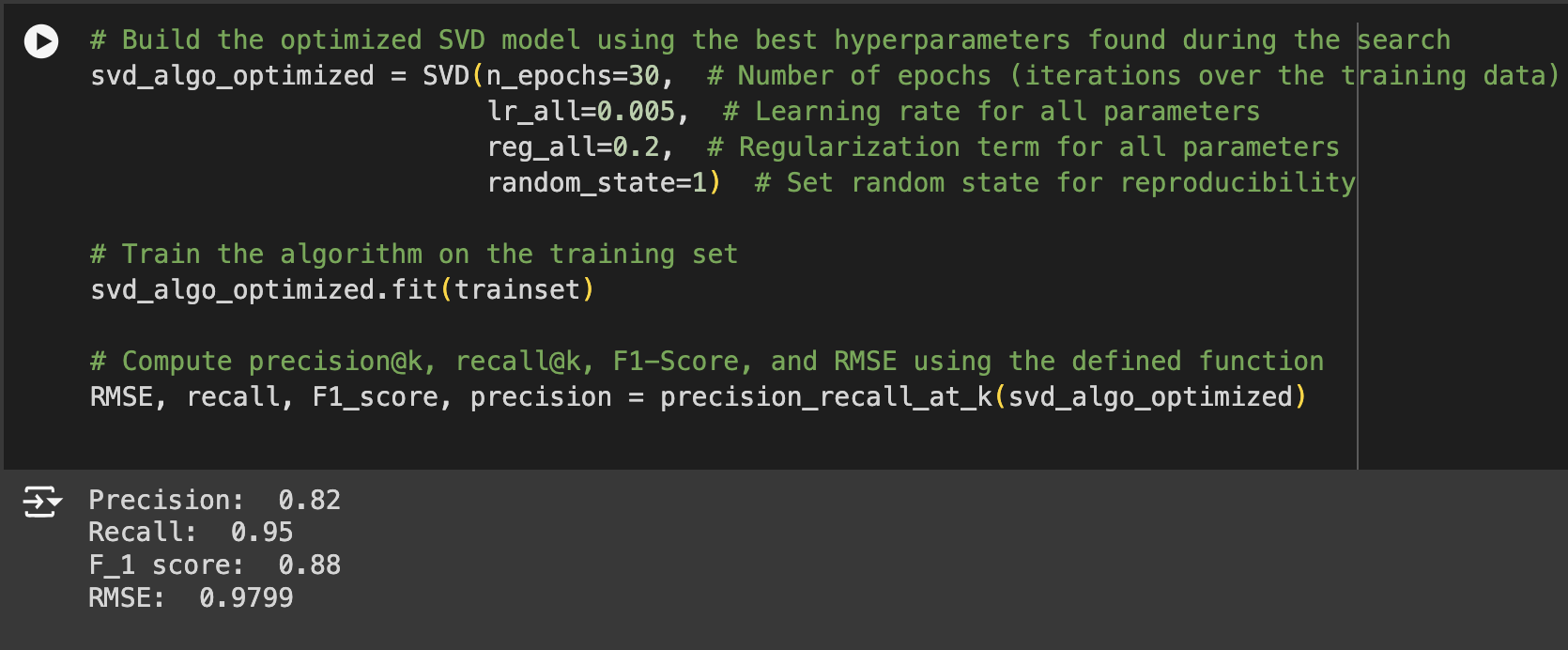


Fig: 7.6 Precision vs Recall After Optimization for Model based

The graphical representation of Precision vs Recall Before & After Optimization for Model:

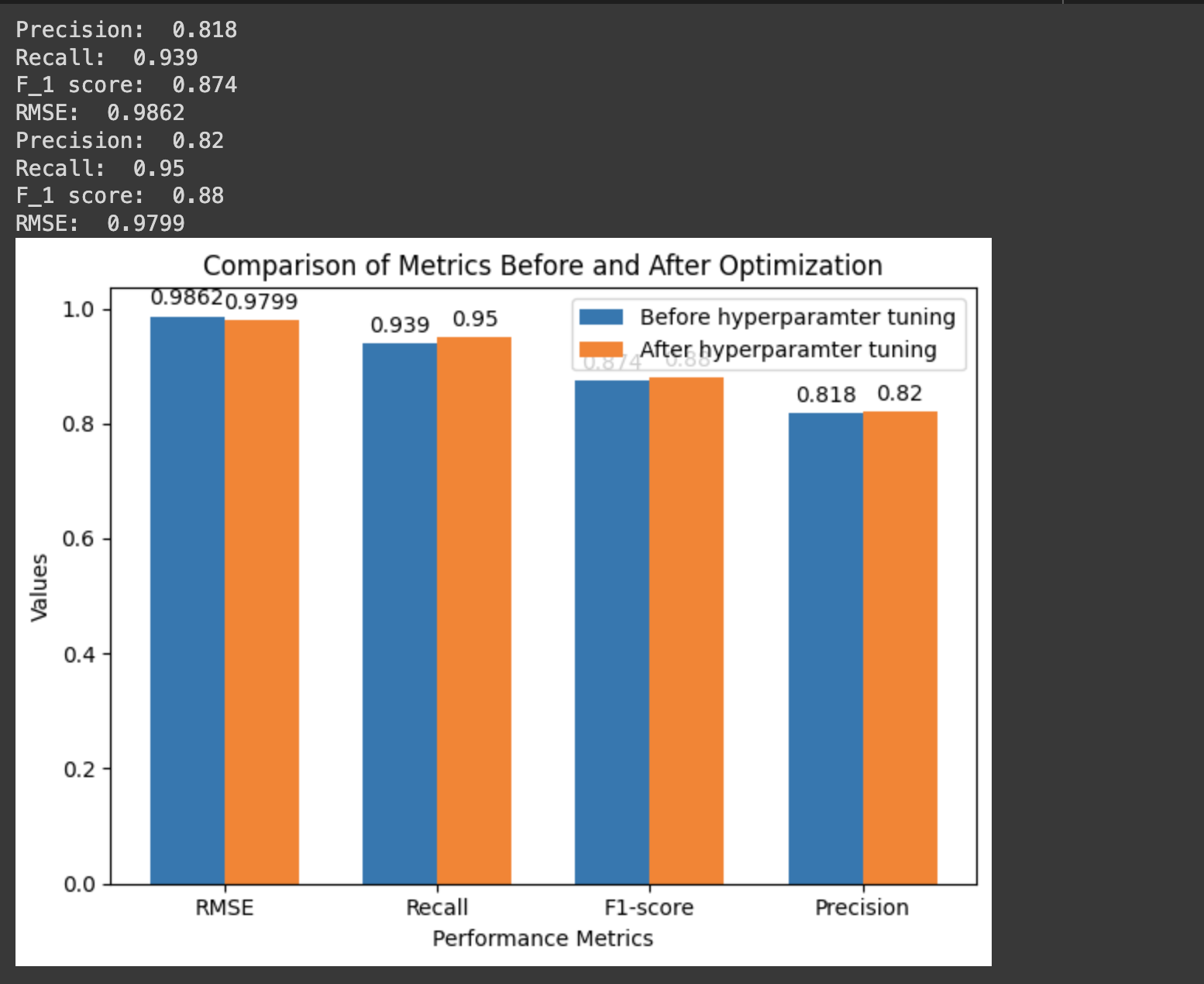
****

Fig :7.7 Graphical representation of performance metrics for Model based

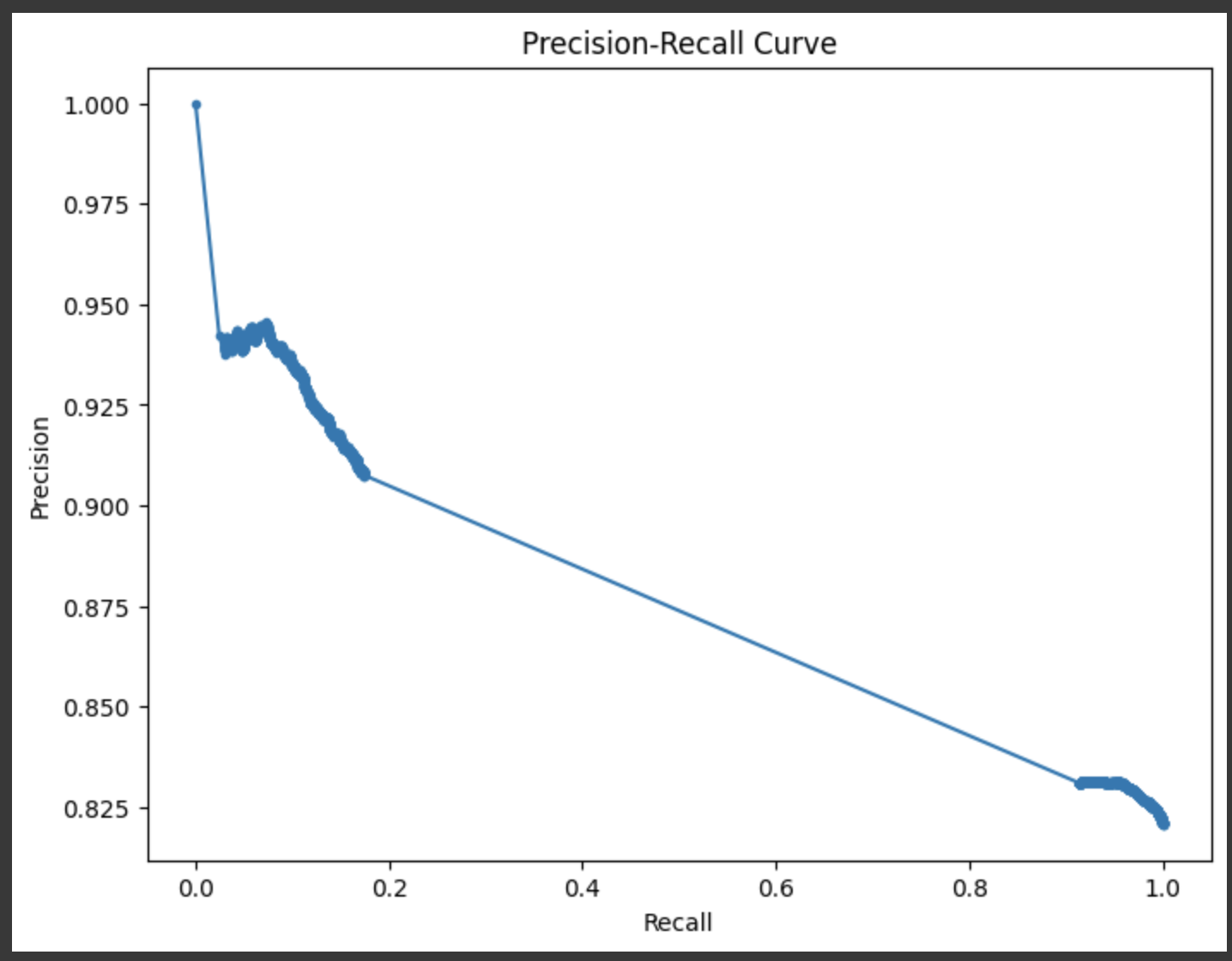


Fig:7.8 Precision vs Recall curve

By comparing these two metrics, we can assess how well the recommendation system aligns with the actual preferences of users. If the average predicted ratings closely match the average actual ratings, it suggests that the recommendation system is accurately predicting user preferences and providing relevant recommendations. On the contrary, a significant disparity between the two averages may indicate that the recommendation system needs improvement in understanding user preferences or in the accuracy of its predictions.

**Root Mean Square Error:**

RMSE provides a measure of the spread of the residuals (the differences between predicted and actual values) and gives an indication of how well the model's predictions match the observed data. A lower RMSE value indicates better accuracy, as it means the model's predictions are closer to the actual values. The Results are as shown:

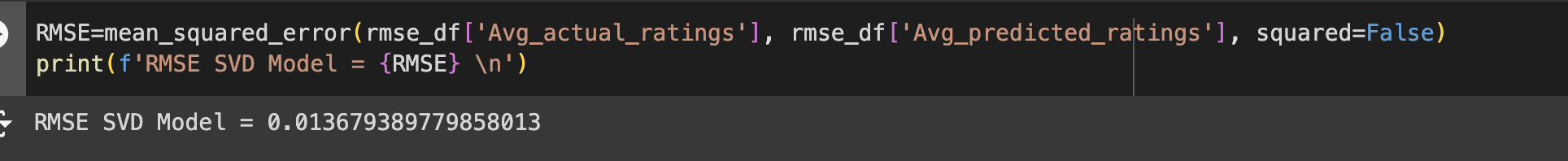


Fig:7.9 Root Mean Square Error for model based

We have Lower RMSE values which generally indicate better predictive accuracy, suggesting that the model's predictions are relatively close to the actual values.

# **Conclusion**

Recommender systems play a crucial role in both academia and e-commerce, serving as essential tools for generating revenue. This paper explores the adoption of recommendation techniques in the industry and evaluates the effectiveness of modern machine learning models in typical e-commerce scenarios where explicit ratings are unavailable and implicit information is sparse.

Our findings indicate that many e-commerce platforms utilize straightforward recommendation methods, such as best-seller lists. For personalized recommendations, traditional methods like KNN collaborative filtering and association rules mining often outperform more advanced matrix factorization techniques when applied to binary purchase data, a conclusion supported by previous studies.

In academia, accuracy is often prioritized over speed, with only 11% of research papers reporting runtimes. However, in industrial settings, real-time interaction necessitates faster algorithms, making simpler techniques more favorable. Additionally, the importance of recency and event sequence varies by merchandise type, affecting seasonality in e-commerce datasets. Our experiments suggest that training on all available data is often suboptimal. We propose a time-based splitting strategy to determine the optimal training timespan, enhancing recommendation accuracy and efficiency.

We also distinguish between rank-based and model-based recommendation approaches. Rank-based systems, which sort and recommend items based on their popularity or relevance scores, are simpler and often more efficient. Model-based systems, on the other hand, rely on sophisticated algorithms like matrix factorization and deep learning to predict user preferences. While ***model-based methods can offer more personalized recommendations***, they are generally more complex and require more computational resources.

Traditional random sampling methods, which can involve training on future events to predict past ones, often result in overly optimistic performance estimates due to erroneous shopping patterns.

# **References**

1. A top-N recommendation model Based on feedback learning to rank and online learning: <https://ieeexplore.ieee.org/document/8054042>
2. Recommendation System in Python

<https://www.geeksforgeeks.org/recommendation-system-in-python/>

1. “An overview of Machine Learning Technologies and their use in E-learning” :  
    https://ieeexplore.ieee.org/document/9151758
2. “A Survey on Machine Learning Approaches and Its Techniques”:  
   <https://ieeexplore.ieee.org/document/9087123>
3. “Multimodal Representation Learning: Advances, Trends and Challenges”:  
   <https://ieeexplore.ieee.org/document/8949228>
4. “Similarity Based Collaborative Filtering Model for Movie Recommendation Systems”:<https://ieeexplore.ieee.org/document/9432354>
5. “Product Rank Based Search Engine for E-Commerce”:<https://ieeexplore.ieee.org/document/8529503>
6. “Research on Personalized Learning Route Model Based on Improved Collaborative Filtering Algorithm”: https://ieeexplore.ieee.org/document/9696131