PRODUCT RECOMMENDATION USING MACHINE LEARNING

Submitted by:

ASTHALOCHAN MOHANTA,

SWAGAT MOHANTY,

BISWANATH SAHOO

Under the guidance of:

MR.PRADIPTA KUMAR PATTANAYAK

M.SC. DATASCIENCE

Silicon Institute of Technology,

Bhubaneswar

ABSTRACT

In the modern era, e-commerce has become the go-to platform for people's daily shopping needs. With the abundance of options available, it can often be challenging for consumers to make the right choices. This is where recommendation systems step in to alleviate these difficulties. Acting as powerful marketing tools, recommendation systems enable the promotion of various products, while simultaneously providing customers with a seamless and personalized shopping experience.

These systems rely on customer behavior data, including search history, purchase history, and clickstream data, to gain deep insights into individual preferences and interests. By analyzing this wealth of data, recommendation systems can accurately understand customers' needs and make tailored product suggestions. Whether it's suggesting similar items based on previous purchases or providing personalized recommendations based on browsing patterns, these systems empower customers to find what they truly need and desire.

INTRODUCTION

A product recommendation system in e-commerce utilizes machine learning techniques to understand customer preferences, predict their future buying behaviors, and offer tailored suggestions accordingly. These systems leverage a variety of data sources, including customer profiles, purchase histories, browsing patterns, and product attributes. By analyzing this wealth of information, machine learning models can uncover intricate patterns and relationships that may not be readily apparent to human analysts.

One of the primary methods in product recommendation systems is collaborative filtering. This technique compares the preferences

and behaviors of different users to find similarities and make recommendations based on those similarities. Collaborative filtering can be implemented using either user-based or item-based approaches. User-based collaborative filtering recommends items based on the preferences of similar users, while item-based collaborative filtering suggests items based on their similarity to items previously favored by the user. Here we used item-based collaborative filtering.

Another widely used approach is content-based filtering, which focuses on the characteristics and attributes of products themselves. By analyzing features such as product descriptions, metadata, and customer reviews, machine learning algorithms can identify patterns and correlations between these attributes and customer preferences. Content-based filtering enables the system to recommend products based on specific attributes that match the customer's preferences, such as brand, price range, or product category.

The benefits of utilizing machine learning-based recommendation systems in e-commerce are substantial. By providing customers with personalized and relevant suggestions, these systems can significantly improve the overall shopping experience. Customers are more likely to find products that align with their tastes and needs, saving time and effort that would otherwise be spent on searching through countless options. This personalized experience fosters customer satisfaction, loyalty, and ultimately drives increased sales and customer retention.

ARCHITECTURE

Recommender systems can be built using various architectures, depending on the specific requirements and goals of the system. Here are a few commonly used architectures:

0.1 Candidate Generation

Candidate generation involves creating a limited set of product options for the user from a vast pool of available products. This selection process takes into account the user's past activity as input to determine the subset of products.

To obtain a smaller subset of products, we analyze the user's previous interactions, such as their purchase history, viewed products, search queries, and other behavioral patterns. By considering these actions, we can identify products that align with the user's preferences and interests, thus narrowing down the available options to a more personalized and relevant set.

0.2 Scoring

During the scoring phase of the system, each candidate generated is assigned a rank on a scale of 1 to 10. This ranking reflects the model's assessment of the candidates' suitability for the user. The candidates with higher scores are prioritized and recommended to the user.

In the scoring phase, the model evaluates the characteristics and relevance of each candidate based on various factors such as user preferences, item attributes, and historical data. The assigned score indicates the model's confidence in the suitability of the candidate for the user's needs and preferences.

By considering these scores, the system can effectively identify the candidates that have a higher likelihood of being well-received by the user. Recommendations are then made by prioritizing and presenting the candidates with the highest scores, as they are deemed to be the most relevant and desirable options for the user.

0.3 Re-ranking

After the scoring process, the system goes through a re-ranking stage to accommodate additional constraints, such as ensuring product diversity and freshness. In order to achieve this, certain guidelines are followed.

Firstly, the system avoids recommending products that the user has previously indicated a dislike for. By considering the user's feedback, such as explicit dislikes or negative interactions, the system aims to avoid suggesting items that are likely to be rejected or disliked by the user.

Secondly, the system gives priority to newly available products. This means that recently introduced or updated items are more likely to be included in the recommendations. By considering the freshness of products, the system ensures that users are exposed to a variety of options, including recently added or updated items that may align with their evolving preferences.

By incorporating these additional constraints into the re-ranking process, the system strives to provide a diverse and up-to-date set of recommendations that align with the user's preferences, while also taking into account their specific dislikes and the availability of fresh options.

The pipeline of a recommendation system has the following five phases

1) Pre-Processing
2) Model Training
3) Hyper Parameter Optimizations
4) Post Processing
5) Evaluations

1) Pre-Processing

Normalization:

In the dataset there are users who are over reactive and provide high ratings or least ratings. We need to normalize the ratings from such users. For this we find the average rating of the product and average rating of the user and add them to the global average.

Formula for normalization is

User-item rating bias= global average + item's average rating + user's average rating.

2) Model Training

Following the pre-processing stage, the subsequent step involves training the model. In model building, a common technique employed is matrix factorization. Specifically, Single Value Decomposition (SVD) is often utilized to decompose the original matrix and extract latent factors that contribute to generating recommendations.

During the model building phase, the original matrix, representing the user-item interactions or ratings, is decomposed using SVD. This decomposition separates the matrix into three components: the user matrix, the item matrix, and the singular value matrix. These components capture the underlying latent factors that influence user preferences and item

characteristics.

By decomposing the matrix, the model identifies the latent factors that play a significant role in the user-item interactions. These factors could represent attributes such as genre, director, or other relevant characteristics depending on the application domain. The extracted latent factors enable the model to understand the underlying patterns and relationships between users and items.

The obtained latent factors are then leveraged to generate recommendations. The model utilizes these factors to calculate similarities or predictions, allowing it to recommend items that align with a user's preferences and exhibit similar characteristics to those they have previously shown interest in.

By employing matrix factorization, specifically SVD, in the model-building process, the system can effectively capture and utilize latent factors to provide accurate and personalized recommendations based on user-item interactions.

3) Hyper Parameter Optimizations

Before tuning the parameters, an evaluation metric needs to be selected. In our case, we have chosen the value of K, which represents the number of recommendations to be generated for the user. In our research, we have specifically set the value of K to 10, meaning that the system recommends the top 10 products to the user.

By selecting K as 10, we aim to provide a substantial number of recommendations to the user, ensuring a diverse set of options to explore. This value allows for a reasonably extensive range of choices while balancing the need for relevance and personalization.

The decision to set K as 10 is based on our evaluation of the system's performance and the desired user experience. It is crucial to strike a

balance between providing a sufficient number of recommendations and avoiding overwhelming the user with an excessive amount of options.

With K set as 10, the system can present a substantial number of high-quality recommendations to the user, reflecting their preferences and maximizing the chances of finding products that align with their interests.

4) Post Processing

As part of the post-processing stage, we implement a step to identify any items that the user has previously purchased. These purchased items are then removed from the generated recommendations, ensuring that there are no duplicates in the final list of recommended products.

In the post-processing phase, the system compares the list of recommended items with the user's purchase history. Any items that have been previously purchased by the user are filtered out, ensuring that they are not included in the final recommendations presented to the user.

By removing duplicates, we enhance the relevance and diversity of the recommended products. This step prevents the system from suggesting items that the user has already acquired, enabling them to explore new options and discover fresh products that align with their preferences.

The post-processing step, which involves identifying and excluding previously purchased items, plays a crucial role in refining the recommendation list, and providing a personalized and unique set of products to the user.

5) Evaluations

To assess the performance of the model in this project, we have selected Root Mean Square Error (RMSE) as the evaluation metric. RMSE measures the average difference between the predicted ratings or scores and the actual ratings in the dataset. By calculating RMSE, we can gauge

how accurately the model predicts user preferences.

For the implementation of this project, we utilize several Python libraries, including NumPy, pandas, matplotlib, Seaborn, and Scikit-learn. These libraries play a crucial role in the system design as they enable efficient data manipulation, visualization, and provide necessary functionalities to achieve accurate results.

NumPy provides essential numerical operations and data structures, while pandas offers powerful data manipulation tools, making it easier to preprocess and analyze the dataset. Matplotlib and Seaborn are used for data visualization, allowing us to gain insights from the data and present the results effectively. Scikit-learn offers machine learning algorithms and evaluation metrics, which assist in building and evaluating the recommendation model.

By incorporating these Python libraries into the project's system design, we ensure efficient data handling, insightful visualizations, and access to machine learning algorithms, ultimately aiding in the accurate evaluation of the model's performance.

IMPLEMENTATION

The choice of Python as the programming language for building the system was primarily driven by its ease of use and the extensive range of built-in libraries. Python's user-friendly syntax and vast ecosystem of libraries make the implementation process convenient and less daunting.

In the initial stage, we focus on selecting the dataset required for the recommender system. The dataset comprises four columns. The first column represents the unique User ID, which plays a crucial role in performing Exploratory Data Analysis (EDA). EDA helps us gain insights into the data and provides context for building an effective model. By analyzing the User ID column, we can identify any potential errors or anomalies in the dataset.

Pre-processing the dataset is an essential step to ensure the accuracy and completeness of the data. We thoroughly examine each field in all four columns, ensuring that there are no missing or null values. Additionally, we validate the integrity of the second column, which contains unique Product IDs for each item in the dataset. The third column contains ratings assigned by users to the respective products, while the fourth column provides timestamp information, indicating the time of each rating.

After carefully considering and preparing the dataset, we proceed with the subsequent steps of the recommender system's development.

After reading the dataset from the CSV file we want to ensure that all the columns are read into the system correctly without any missing values

By using the .info() and .dtypes we can get a information about index dtype,column dtype and usage of the memory.

To see the structure of the dataset we have used the .describe() function which gives information about the count, mean, standard deviation and other quantile values like min,50 percent and maximum value.

After reading the dataset and finding all the details, we move to the ratings column in the dataset. Since it uses the ratings from different users, we plot the overall ratings to see if they are well distributed.The plot shows that five star ratings are given the most by the users to different products and lowest number of users rated the products with two star ratings .

From this dataset we drop the unnecessary columns that we may not need to make it more usable. Next we find the popular products among all the products and average rating for the products . In the next step we have to choose a classification algorithm to generate the recommendations. In this project we decide to use PCA algorithm where specially we used Truncated Singular value Decomposition method.

We used Truncated SVD (Singular Value Decomposition) in product recommendation systems for several reasons:

1.Dimensionality Reduction: Product recommendation systems often deal with high-dimensional data, where each product can have numerous features. Truncated SVD helps in reducing the dimensionality of the data while preserving important information. By retaining only the most significant singular values and corresponding singular vectors, Truncated SVD allows us to represent the data in a lower-dimensional space, making computations more efficient.

2. Noise Reduction: Recommender systems frequently encounter noisy data, such as sparse ratings or irrelevant features. Truncated SVD can effectively filter out noise by reducing the influence of less informative dimensions. It captures the latent factors underlying the data, which are often the essential components for generating accurate recommendations.

3. Improved Scalability: In large-scale recommendation systems, computational efficiency is crucial. Truncated SVD enables faster computations by reducing the complexity associated with high-dimensional data. It allows us to work with a smaller, more manageable matrix, resulting in faster training and prediction times.

4. Implicit Feature Discovery: Truncated SVD can uncover latent features that may not be explicitly present in the original data. These latent features capture underlying patterns and relationships among products and users, helping to improve recommendation accuracy. By identifying these hidden factors, Truncated SVD enables the system to capture more nuanced and subtle connections between products and users.

Overall, Truncated SVD is a valuable technique in product recommendation systems as it addresses the challenges of high dimensionality, noise reduction, scalability, and implicit feature discovery. It helps improve the accuracy and efficiency of the recommender system by representing the data in a lower-dimensional space while preserving the most relevant information.

To evaluate the performance of the model we have used the Root Mean Square Error (RMSE) which gives the accuracy of the recommendations that are generated. The results of the RMSE can gives a value of 1.3436.To eliminate the products that were already bought by the users, we decompose the outputs. This removes the products that were already bought by the user. Therefore, the final recommendation will not have any duplicates in it.

CONCLUSION

The main objective of this project is to leverage machine learning algorithms to provide personalized recommendations to users on an e-commerce website. We have implemented the system using collaborative filtering and the Pearson correlation coefficient. The dataset we considered contains ratings provided by other users for specific products. Based on the similarity between the rated products, we aim to recommend relevant items to the current user.

Our future work involves enhancing the system's efficiency and expanding its capability to cater to users with no previous purchase history or new users. To achieve this, we plan to explore the application of recurrent neural networks (RNNs) and deep learning techniques. Deep learning can address some limitations of traditional matrix factorization methods. By incorporating RNNs, we can introduce the temporal aspect into the recommender system, which is not feasible with matrix factorization alone.

Furthermore, we intend to explore the concept of providing sub-optimal recommendations to users and analyzing their reactions. This feedback from users can be valuable for the system in improving its future recommendations. By recording and analyzing user reactions, we can refine the recommendation process and enhance the overall user experience.

In summary, our future work includes incorporating deep learning techniques, specifically RNNs, to improve the recommender system's performance and address its limitations. We also aim to leverage user feedback to optimize recommendations and continually enhance the system's capabilities.

REFERENCES

Here are some references for machine learning-based product recommendation systems in e-commerce:

1. Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

2. Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web (WWW 2001) (pp. 285-295).

3. Ricci, F., Rokach, L., Shapira, B. (2011). Introduction to recommender systems handbook. In Recommender Systems Handbook (pp. 1-35). Springer.

4. Zhang, Y., Koren, Y. (2014). Collaborative filtering for implicit feedback datasets. In Proceedings of the 8th ACM Conference on Recommender Systems (pp. 263-270).

5. Covington, P., Adams, J., Sargin, E. (2016). Deep neural networks for YouTube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 191-198).

6. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T. S. (2017). Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web (pp. 173-182).

7. Li, L., Chu, W., Langford, J., Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th International Conference on World Wide Web (pp. 661-670).