Unsupervised machine learning - Product recommendation engine

Technical documentation

Mohammed Arifuddin Atif
arifuddinatif63@gmail.com

Introduction

Amazon.com is one of the largest electronic commerce and cloud computing companies.Amazon relies heavily on a recommendation engine that reviews customer ratings and purchase history to recommend items and improve sales.we are given a dataset which contains the customer reviews and ratings of beauty related products sold on their website.

Problem statement

We are tasked with building a recommender engine that reviews customer ratings and purchase history to recommend items and improve sales.This engine looks at the customers previous purchases and ratings and provides recommendations of products similar to the products they like.

Overview of data

We are given the following columns in our data:

1. **Unique userId -** Unique id used for customer Identification.
2. **Product ASIN -** Amazon's unique product identification code for each product.
3. **Ratings -** Ratings(1-5) given by customers based on their satisfaction.
4. **Timestamp -** Timestamp of the given rating in UNIX time.

The **Ratings** column provides valuable information which tells the satisfaction of the customer from the product or the extent of their likeness towards a particular product.

The **Unique userId** column provides information on the identification of the customer.This id is unique to a customer.

Steps involved

I. Performing EDA (exploratory data analysis)

A. Exploring head and tail of the data to get insights on the given data.
B. Getting overview of the data.
C. Looking for null values and removing them if it affects the performance of the model.
D. Getting the inter-quartile range of the **Rating** column.
E. Getting the total number of unique users from the data.
F. Extracting the total number of unique products from the given dataset.
G. Grouping the products with high ratings.Given rating is considered high when the rating is more than or equal to 4.

II. Drawing conclusions from the data

Plotting necessary graphs which provides relevant information on our data like :

A. Getting a plot on the total number of ratings for each rating category.
B. Getting plots of top 20 products based on number of sales.

III. Training the model

A. Splitting the model into train and test sets.

B. Grouping the train data with the mean of **Product ASIN** column.

C. Again grouping the train data with the count of the **Rating** column.

D. Filtering the data accordingly as the given data is huge.

E. Hyperparameter tuning of the model.

F. Getting the predicted ratings from the model.

IV. Evaluating metrics of our model

A. Calculating the RMSE value of all the models used.

B. Building a function which recommends 5 products and checking the accuracy of the recommendations.

Models used

Popularity based recommender:

It is a type of recommendation system which works on the principle of popularity and or anything which is in trend. These systems check about the product or movie which are in trend or are most popular among the users and directly recommend those.

For example, if a product is often purchased by most people then the system will get to know that that product is most popular so for every new user who just signed it, the system will recommend that product to that user also and chances become high that the new user will also purchase that.

Merits of popularity based recommendation system:

- It does not suffer from cold start problems which means on day 1 of the business also it can recommend products on various different filters.
- There is no need for the user's historical data.

Demerits of popularity based recommendation system:

- Not personalized
- The system would recommend the same sort of products/movies which are solely based upon popularity to every other user.

Collaborative filtering:

Collaborative does not need the features of the items to be given. Every user and item is described by a feature vector or embedding.It creates embedding for both users and items on its own. It embeds both users and items in the same embedding space.

It considers other users' reactions while recommending a particular user. It notes which items a particular user likes and also the items that the users with behavior and likings like him/her likes, to recommend items to that user.It collects user feedbacks on different items and uses them for recommendations.

Types of collaborative Recommender Systems:

Memory-based collaborative filtering: Done mainly remembering the user-item interaction matrix, and how a user reacts to it, i.e, the rating that a user gives to an item. There is no dimensionality reduction or model fitting as such.

User-User filtering: In this kind, if a user A's characteristics are similar to some other user B then, the products that B liked are recommended to A. As a statement, we can say, "the users who like products similar to you also liked those products". So here we recommend using the similarities between two users.

Now, if one user A behaves like other users B, C, and D, then for a product x, A's rating is given by:

$$R_{xu}=(\Sigma_{i=0}{}^{n}R_{i}) / n$$

Where Rxu is the rating given to x by user u and i=0 to n are the users who have shown behavior similar to u. Now, all the n users are not an equal amount similar to the user u. So, we find a weighted sum to provide the rank.

$$R_{xu}=(\Sigma_{i=0}^{n}R_iW_i\ ) \ / \ \Sigma_{i=0}^{n}W_i$$

The weights here are the similarity metrics used.

Now, users show some differences in behaviors while rating. Some are generous raters, others are not, i.e, maybe one user rates in range 3 to 5, while other user rates 1 to 3. So, we calculate the average of all the ratings that the user has provided, and subtract the value from Ri in order to normalize the ratings by each user.

Item-Item filtering: Here, if user A likes an item x, then, the items y and z which are similar to x in property, then y and z are recommended to the user. As a statement, it can be said, “Because you liked this, you may also like those”.

The same equations are used here also

$$R_{xu}=(\Sigma_{i=0}^{n}R_i) \ / \ n$$

Where R is the rating user u gives to the product x, and it is the average of the ratings u gave to products like x. Here also, we take a weighted average

$$R_{xu}=(\Sigma_{i=0}^{n}R_iW_i\ ) \ / \ \Sigma_{i=0}^{n}W_i$$

Where the Weight is the similarity between the products.

Similarity Metrics:

They are mathematical measures which are used to determine how similar is a vector to a given vector.

Similarity metrics used mostly:

1. Cosine Similarity: The Cosine angle between the vectors.
2. Dot Product: The cosine angle and magnitude of the vectors also matters.
3. Euclidean Distance: The element wise squared distance between two vectors
4. Pearson Similarity: It is a coefficient given by:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Model-based collaborative filtering: Remembering the matrix is not required here. From the matrix, we try to learn how a specific user or an item behaves. We compress the large interaction matrix using dimensional Reduction or using clustering algorithms. In this type, We fit machine learning models and try to predict how many ratings a user will give a product. There are several methods:

1. **Clustering algorithms**
2. **Matrix Factorization based algorithm**
3. **Deep Learning methods**

Approach used

Surprise library:

Surprise is a Python scikit for building and analyzing recommender systems that deal with explicit rating data.

Purpose of surprise library:

- Give users perfect control over their experiments. To this end, a strong emphasis is laid on documentation, which we have tried to make as clear and precise as possible by pointing out every detail of the algorithms.
- Alleviate the pain of Dataset handling. Users can use both built-in datasets (Movielens, Jester), and their own custom datasets.
- Provide various ready-to-use prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based (SVD, PMF, SVD++, NMF), and many others. Also, various similarity measures (cosine, MSD, pearson…) are built-in.
- Make it easy to implement new algorithm ideas.
- Provide tools to evaluate, analyse and compare the algorithms' performance. Cross-validation procedures can be run very easily using powerful CV

iterators (inspired by scikit-learn excellent tools), as well as exhaustive search over a set of parameters.

The name *SurPRISE (roughly :)) stands for Simple Python RecommendatIon System Engine.*

Please note that surprise does not support implicit ratings or content-based information.

Singular Value Decomposition:

The Singular Value Decomposition (SVD), a method from linear algebra that has been generally used as a dimensionality reduction technique in machine learning. SVD is a matrix factorisation technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where K<N). In the context of the recommender system, the SVD is used as a collaborative filtering technique. It uses a matrix structure where each row represents a user, and each column represents an item. The elements of this matrix are the ratings that are given to items by users.

K-Nearest Neighbor(KNN):

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Challenges faced

1. The given data contains approximately 20 lakh data points which made it difficult to work on the data.So we had to filter the data accordingly.
2. Tuning the model appropriately was one of the challenges we faced as the metrics of our model before tuning were not acceptable.

Conclusion

We are finally at the conclusion of our project!

Coming from the beginning we did EDA on the dataset and also cleaned the data according to our needs.After that we were able to draw relevant conclusions from the given data and then we trained our model on recommender systems.

For our model we were able to get an RMSE value of 0.875 after hyper parameter tuning.

Given the size of data and the amount of irrelevance in the data , the above score is good.