**Product Recommendation Engine**

**Sandeep R**

**Data science trainee,**

**AlmaBetter, Bangalore**

**Abstract:**

Online stores have millions of products available in their catalogs. Finding the right product becomes difficult because of this ‘Information overload’. Users get confused and this puts a cognitive overload on the user in choosing a product.

Recommender systems help customers by suggesting probable list of products from which they can easily select the right one. They make customers aware of new and/or similar products available for purchase by providing comparable costs, features, delivery times etc.

Recommender systems have become an integral part of e-commerce sites and other businesses like social networking, movie/music rendering sites. They have a huge impact on the revenue earned by these businesses and also benefit users by reducing the cognitive load of searching and sifting through an overload of data. Recommender systems personalize customer experience by understanding their usage of the system and recommending items they would find useful.

**1.Problem Statement**

To Build a product recommender system which helps the customers by suggesting probable list of products from which they can easily select the right one.

**2. Introduction**

Amazon.com is one of the largest electronic commerce and cloud computing companies.

Amazon is well-known for personalization and recommendations, which help customers discover items they might otherwise not have found. Amazon. com has been building a store for every customer. Each person who comes to Amazon.com sees it differently, because it’s individually personalized based on their interests. It’s as if you walked into a store and the shelves started rearranging themselves, with what you might want moving to the front, and what you’re unlikely to be interested in shuffling further away. From a catalogue of hundreds of millions of items, Amazon.com’s recommendations pick a small number of items you might enjoy based on your current context and your past behaviours. The algorithms aren’t magic; they simply share with you what other people have already discovered. The algorithm does all the work. It’s computers helping people help other people, implicitly and anonymously.

Here I will be Building a model which Recommends products to customer based on Popularity and Collaborative Filtering Model. These Models will be trained on a

The Dataset which contains over 2 million customer reviews and trained Models will be used to Recommend Products to the customers.

## **3. Methods of Recommendation systems**

## Recommender systems can be built with two different methods: Content Based Filtering, and Collaborative Filtering.

## **Content Based Filtering**

In content-based filtering, the similarity between different products is calculated based on the attributes of the products. For instance, in a content-based book recommender system, the similarity between the books is calculated based on genres, the author of the book, the publisher of the book, title of the book etc.

## **Collaborative filtering**

Collaborative filtering is commonly used for recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix..CF is based on the idea that the best recommendations come from people who have similar tastes. In other words, it uses historical item ratings of like-minded people to predict how someone would rate an item. Collaborative filtering has two sub-categories that are generally called memory based and model-based approaches.

**4. Steps involved:**

* **Exploratory Data Analysis**

After loading the dataset we performed this method It’s good practice to know the features and their data types and to take a look at the data distribution. Plotting the data can provide insights into the patterns that the data follows.

* **Null values Treatment**

Our dataset doesn’t contain any null values hence there is no requirement to trat Null values I can proceed further exploring the Dataset.

* **Feature Engineering**

Dataset contains a Timestamp in UNIX time which is converted to Individual Day, Month and Year with help of python datetime library. All These features are Further used for better Visualization Tasks

* **Feature Selection**

In This step only the features which are required for model training are used Features which are selected for Model Building are:

UserId, ProductId and Ratings

* **Fitting different models**

For modelling I have used Two approaches

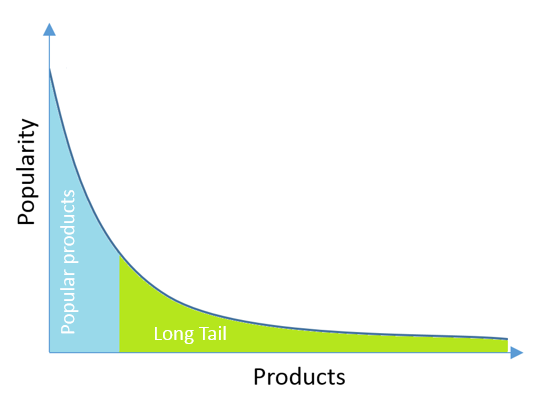
1. **Popularity Recommender model:**
2. **Collaborative Filtering Model**
3. **KNNWithMeans**
4. **SVD[Singular Value Decomposition].**

* **Tuning the hyperparameters for better accuracy**

Tuning the hyperparameters of respective algorithms is necessary for getting better accuracy and to avoid overfitting.

**5. Algorithms:**

1. **Popularity Recommender model:**



Long-tail graph shows the distribution of ratings or popularity among items or products in marketplace. On the X-column items are ordered by their popularity or rating frequencies, whereas y-column shows the popularity in terms of ratings, demand etc. This graph basically points 3 important facts for recommender systems;

**Popularity**  
Products on left side (or in blue area) are called as popular because their popularity is higher then those in green or long-tail area. Moreover popular products are generally competitive products. On the other hand, products in green long-tail area are thought to be unpopular or new products in market. The threshold which discriminates the popular and unpopular items in market is an hyper-parameter for merchant. We will talk about it in later sentences and articles.

**Diversity**Recommender algorithms are generally designed to give recommendations for popular items because they are popular :) However, a good recommendation system should provide diversity. Same and known items can make the customers bored. Therefore adjusting the threshold, starting point of long-tail, in recommendation system is an important research to take into account. Moving it right in the graph can increase the diversity in recommendations made.

**Sparsity**Items in the right side of graph are less rated than the those in left side. This means that there are much more sparsity or unobserved areas for those items in ratings matrix. This can cause a recommender system which relies on neighbourhood algorithms produce bad results. The more we move the threshold to right side, The worse recommendation system results.

.

1. **Collaborative Filtering Model:**

Collaborative filtering is commonly used for recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix..CF is based on the idea that the best recommendations come from people who have similar tastes. In other words, it uses historical item ratings of like-minded people to predict how someone would rate an item. Collaborative filtering has two sub-categories that are generally called memory based and model-based approaches.

**Memory-based**There are two approaches: the first one identifies clusters of users and utilizes the interactions of one specific user to predict the interactions of other similar users. The second approach identifies clusters of items that have been rated by user A and utilizes them to predict the interaction of user A with a different but similar item B. These methods usually encounter major problems with large sparse matrices, since the number of user-item interactions can be too low for generating high quality clusters.

**Model-based**  
These methods are based on machine learning and data mining techniques. The goal is to train models to be able to make predictions. For example, we could use existing user-item interactions to train a model to predict the top-5 items that a user might like the most. One advantage of these methods is that they are able to recommend a larger number of items to a larger number of users, compared to other methods like memory-based approach. They have large coverage, even when working with large sparse matrices

**6. Model performance:**

* **Root Means Square Error**

The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model’s predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable.



Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.

**7.3. Hyper parameter tuning:**

Hyperparameters are sets of information that are used to control the way of learning an algorithm. Their definitions impact parameters of the models, seen as a way of learning, change from the new hyperparameters. This set of values affects performance, stability and interpretation of a model. Each algorithm requires a specific hyperparameters grid that can be adjusted according to the business problem. Hyperparameters alter the way a model learns to trigger this training algorithm after parameters to generate outputs.

I used Grid Search CV, for hyperparameter tuning. This also results in cross validation and in my case I divided the dataset into different folds.

1. **Grid Search CV-**Grid Search combines a selection of hyperparameters established by the scientist and runs through all of them to evaluate the model’s performance. Its advantage is that it is a simple technique that will go through all the programmed combinations. The biggest disadvantage is that it traverses a specific region of the parameter space and cannot understand which movement or which region of the space is important to optimize the model.

**8. Conclusion:**

Built a Product Recommendation system using Surprise library with an Single value Decomposition Algorithm which recommends products to user with an RMSE Score of 0.9615. By using the Algorithm recommended five different products to Each customer depending on the user’s previous purchased products and Rating preference of the Customers.

**References-**

1. Two Decades of Recommender Systems at Amazon.com- Brent Smith