Movie Recommender System

**Project Type**: Application Development (Recommender Systems)

**Project Name** : Movie recommendation and rating model

**Team Name** : Learn2Grow

**Team Members**:

1. Ravi Sharma

2. Suresh Kadhamkode

# **Introduction**

We worked on a Movie Recommender system where we have used different techniques such as

1. Classification to cross verify the accuracy of the clustering
   1. Decision Trees
   2. Random Forest Classifier
2. Recommendation
   1. Top K Popular Movies
   2. K-means Clustering to get an idea of the data and its relationship
   3. User Based Collaborative Filtering
   4. Item Based Collaborative Filtering (Truncated SVD)
   5. KNN based predictions

**Data**

The data for this project was taken from ‘<https://grouplens.org/datasets/movielens/100k/>’ and it is structured into 5 documents

u.data -- The full u data set, 100000 ratings by 943 users on 1682 items.

Each user has rated at least 20 movies.

Users and items are numbered consecutively from 1.

The data is randomly ordered.

This is a tab separated list of ( user id | item id | rating | timestamp).

The time stamps are unix seconds since 1/1/1970 UTC

u.info -- The number of users, items, and ratings in the u data set.

u.item -- Information about the items (movies); this is a tab separated list of

movie id | movie title | release date | video release date |

IMDb URL | unknown | Action | Adventure | Animation |

Children's | Comedy | Crime | Documentary | Drama | Fantasy |

Film-Noir | Horror | Musical | Mystery | Romance | Sci-Fi |

Thriller | War | Western |

The last 19 fields are the genres, a 1 indicates the movie

is of that genre, a 0 indicates it is not; movies can be in

several genres at once.

The movie ids are the ones used in the u.data data set.

u.genre -- A list of the genres.

u.user -- Demographic information about the users; this is a tab separated list of

user id | age | gender | occupation | zip code

The user ids are the ones used in the u.data data set.

u.occupation -- A list of the occupations.

## **Tools and Techniques**

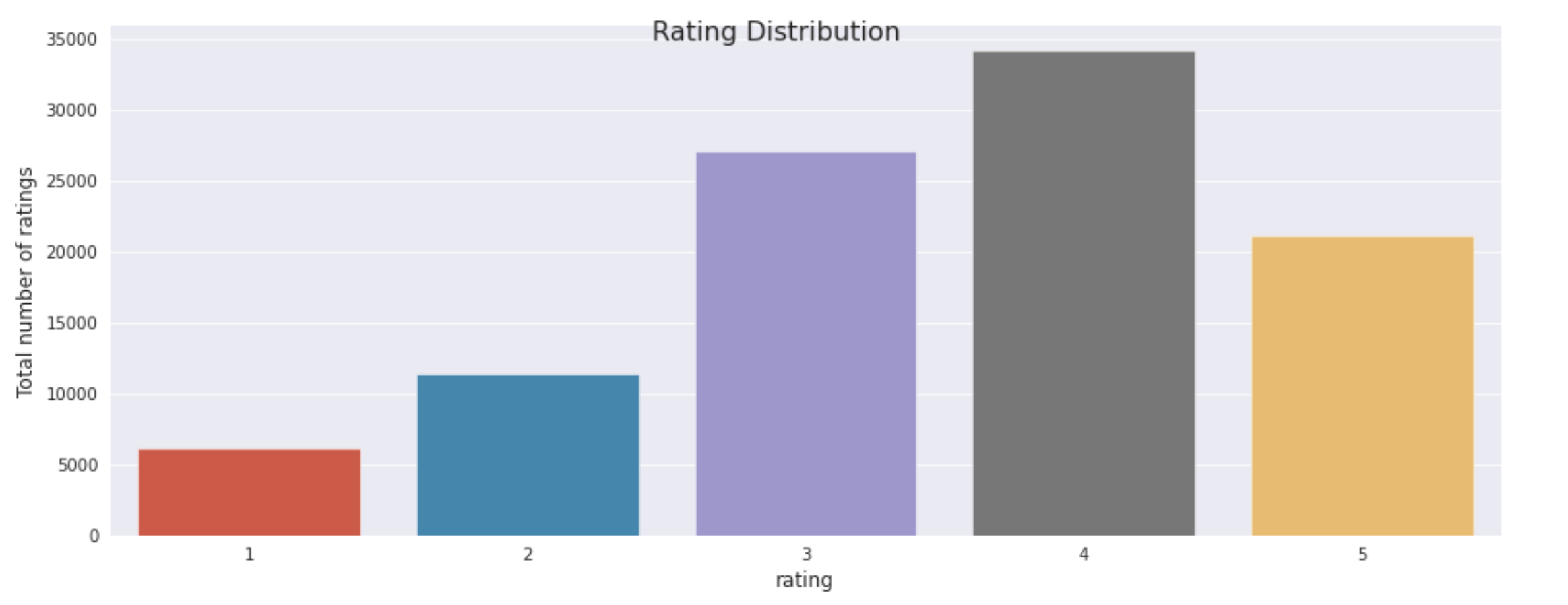
**Data Analysis**

Since ‘u.data’ includes the userid, item id and rating, we used it for recommendation implementation.

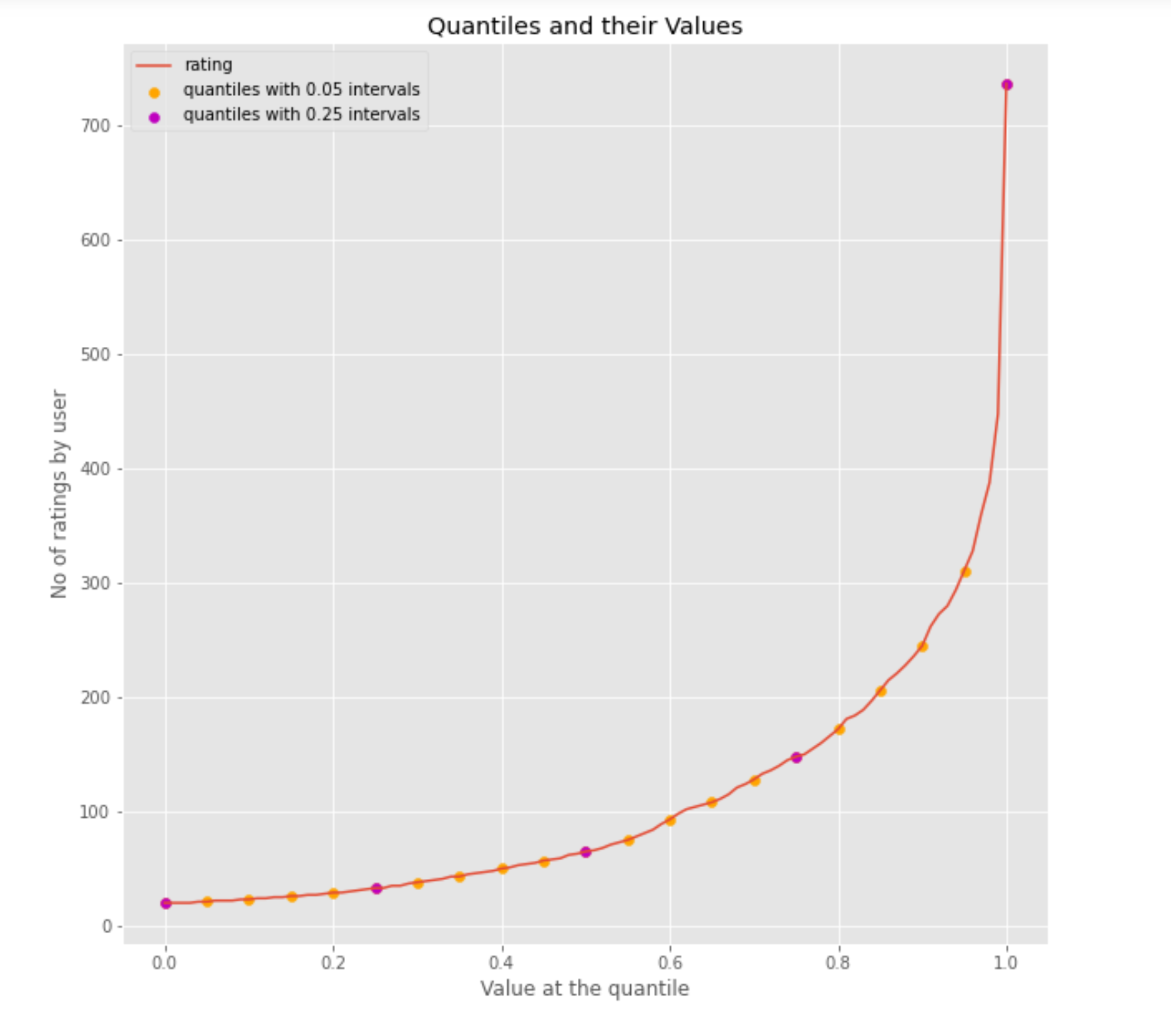
We merged the user rating, movie, user data into one data frame to perform clustering and classification to get the data relation and trends.

Here is some snapshot of the data analysis we did

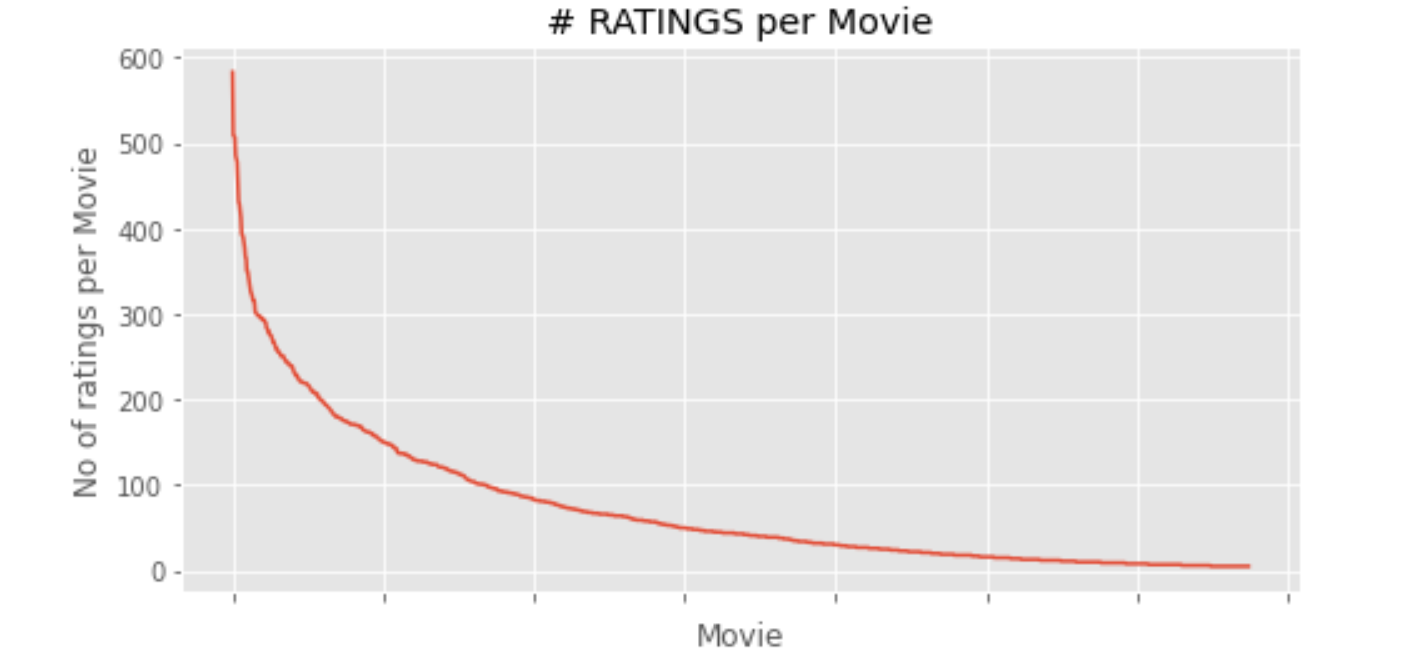
* **Rating Distribution**



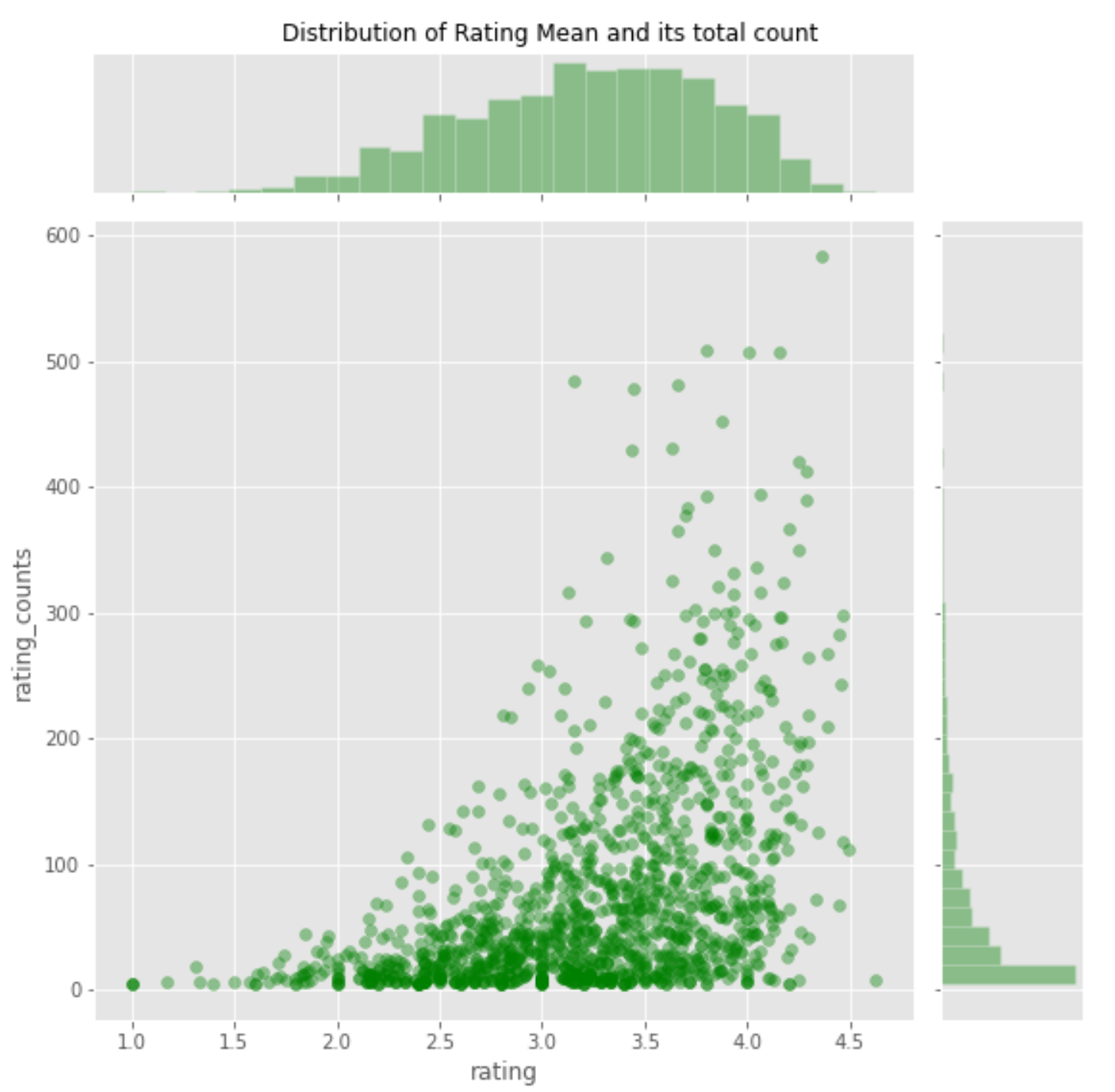
* **Quantile Plot for rating distribution by user**



* **Rating Distribution Per movie**



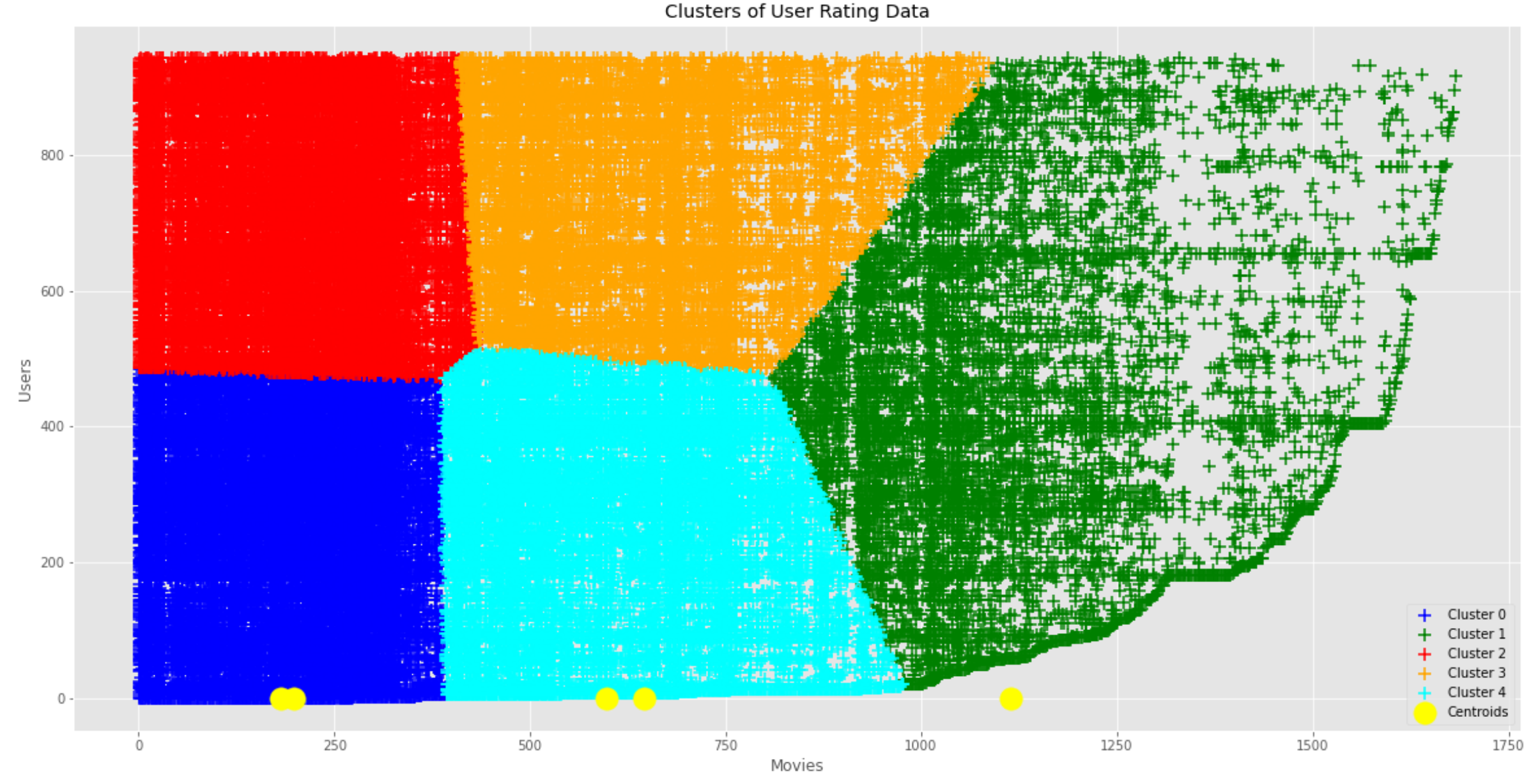
* **Joint plot for rating mean and its total count distribution**



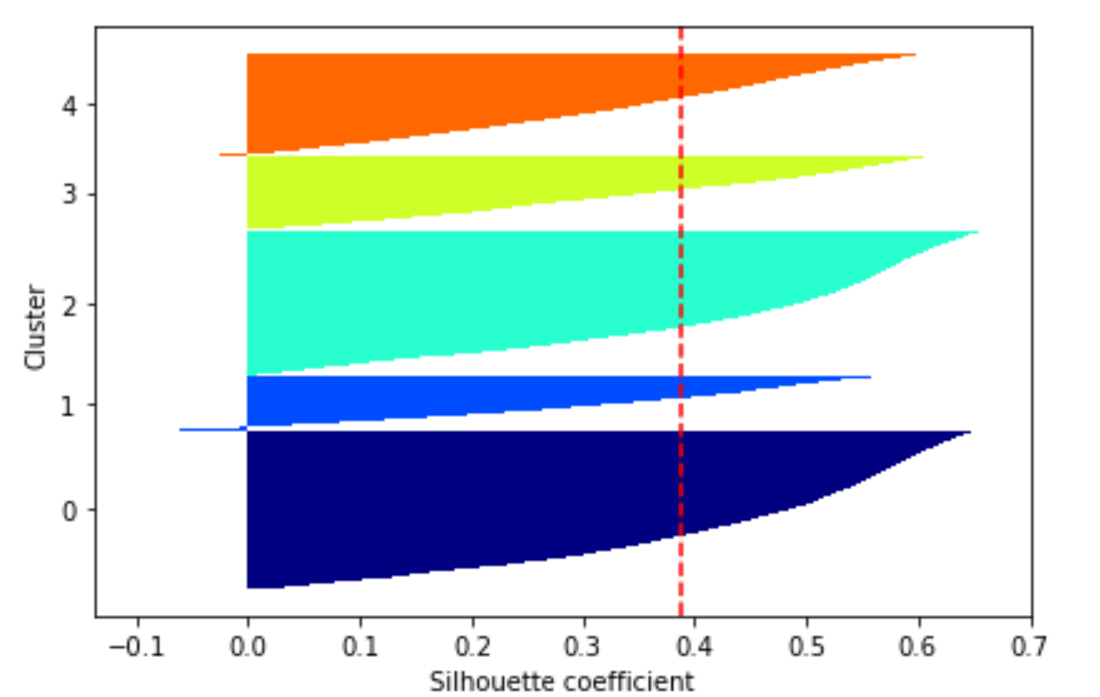
**KMeans Clustering**

We merged movie, movie rating and the user info data and then split it into train-test dataset (80-20 split). We used ratings as the class and used ‘5’ as the number of clusters.

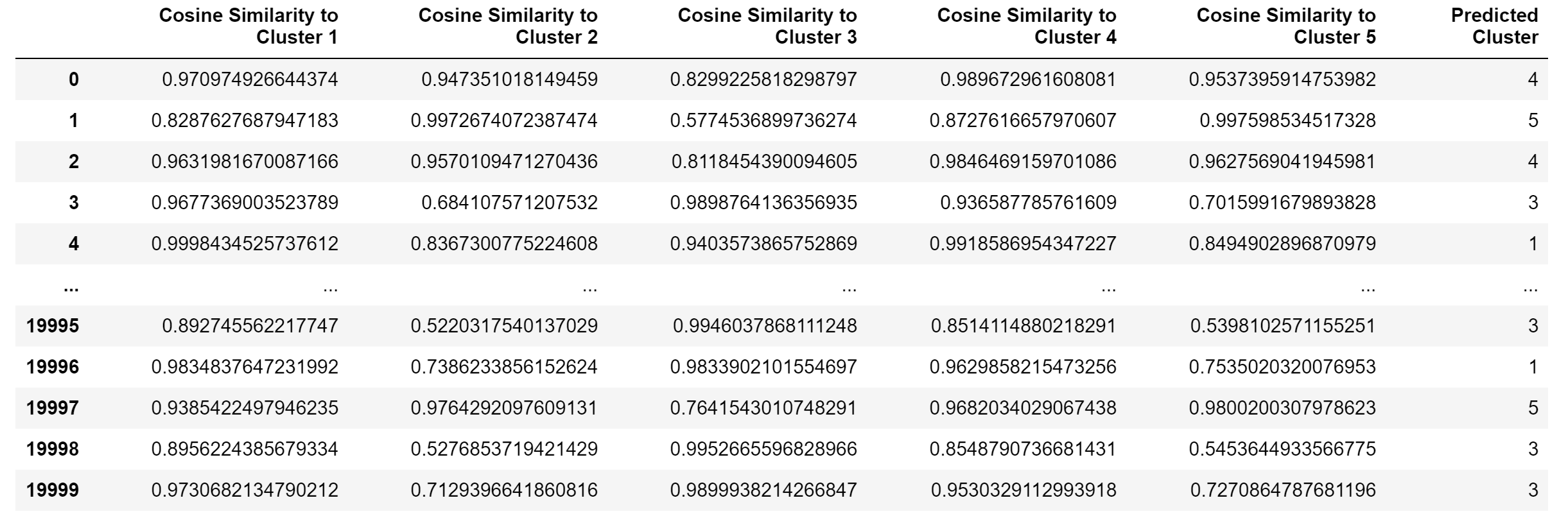
Here is the cluster map



Here is the silhouette plot for the cluster



We also predicted the cluster for the test data



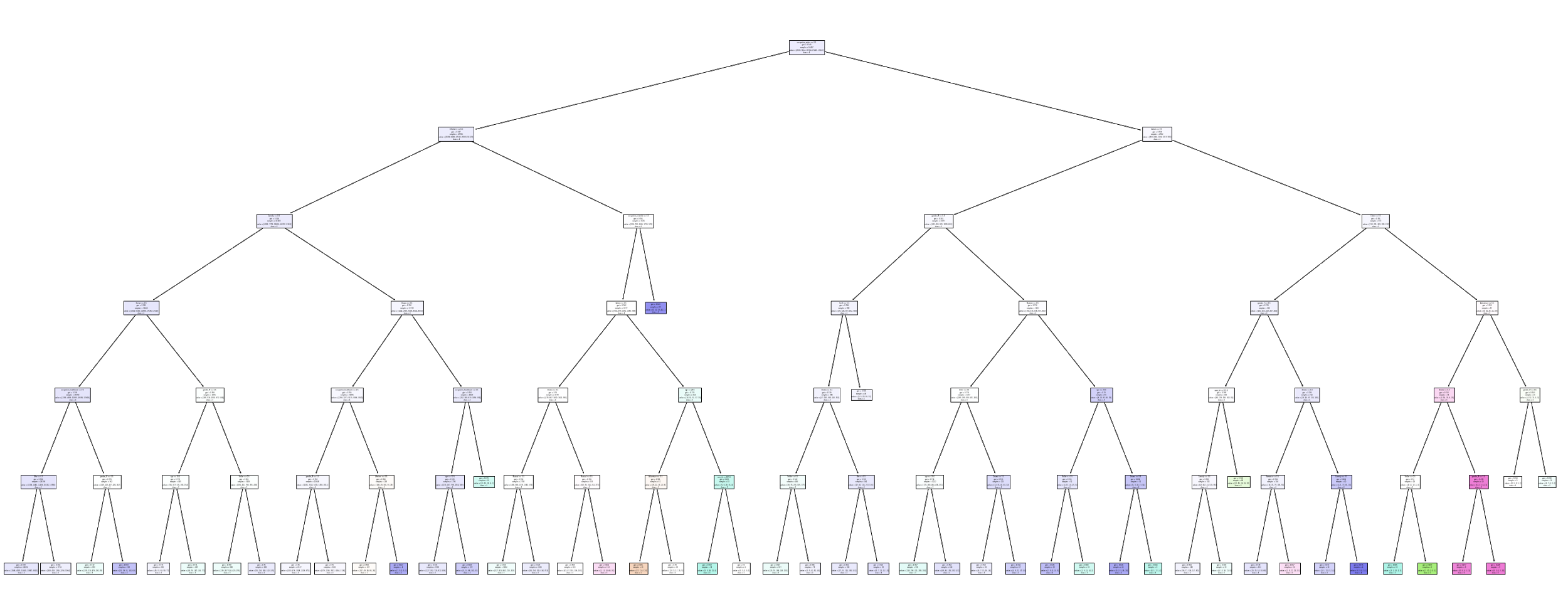
**Classification**

It is the process of finding a model or function that helps in separating the data into multiple categorical classes i.e. discrete values. In classification, data is categorized under different labels according to some parameters given in input and then the labels are predicted for the data.

**Decision Trees**

Decision tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, one will need to trim it down. One common technique used for splitting is the Recursive Binary Splitting. In this approach all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected.

We used the same 80/20 split data that was used for clustering as input for classification tasks as well but with one significant change as we dropped item\_id (the movie\_id) from the data set as this identifier variable is practically meaningless for classifying data.

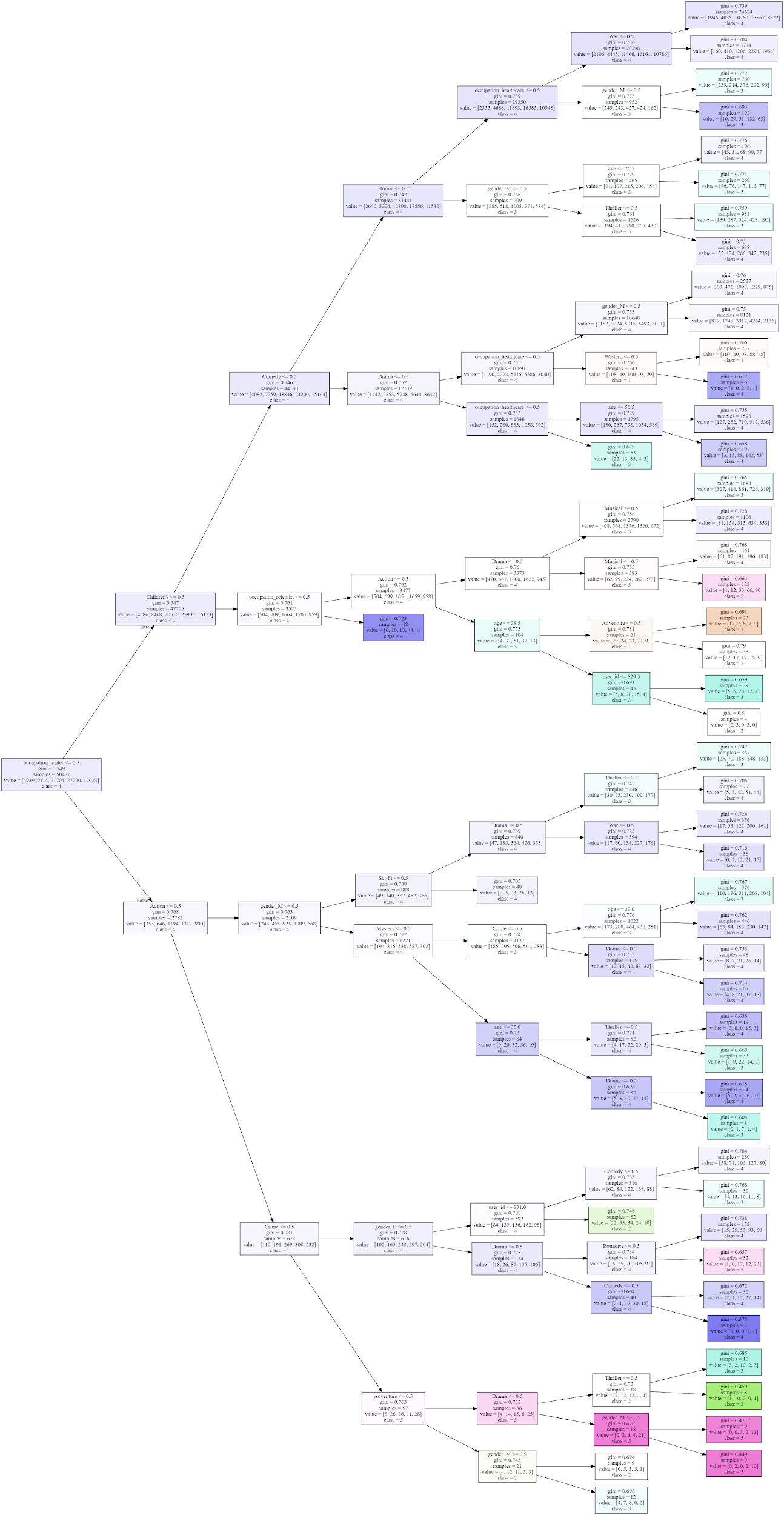


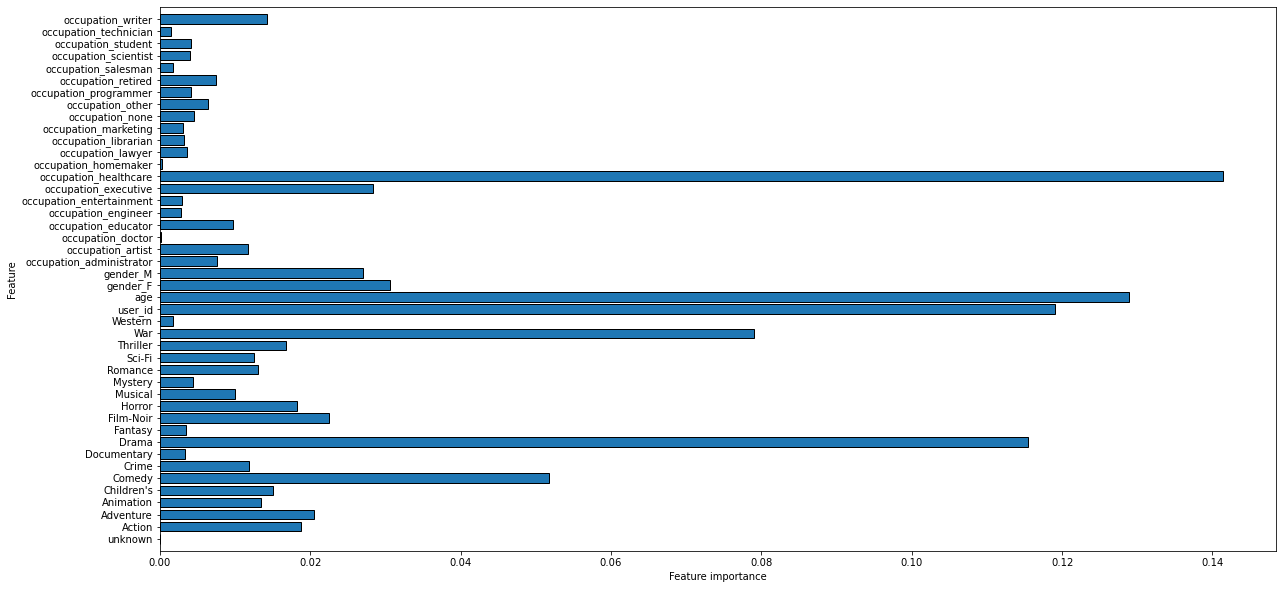
**Random Forest**

Random forest is a supervised learning algorithm. The "forest" it builds is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

The advantage of random forest is that it can be used for both classification and regression problems, which form the majority of machine learning systems.

**Feature importance**



**Cross Validation Accuracy VS min sample leaf values**

## 

**Cross Validation Accuracy VS max depth values**

## 

## **Cross Validation Accuracy VS number of estimators values**

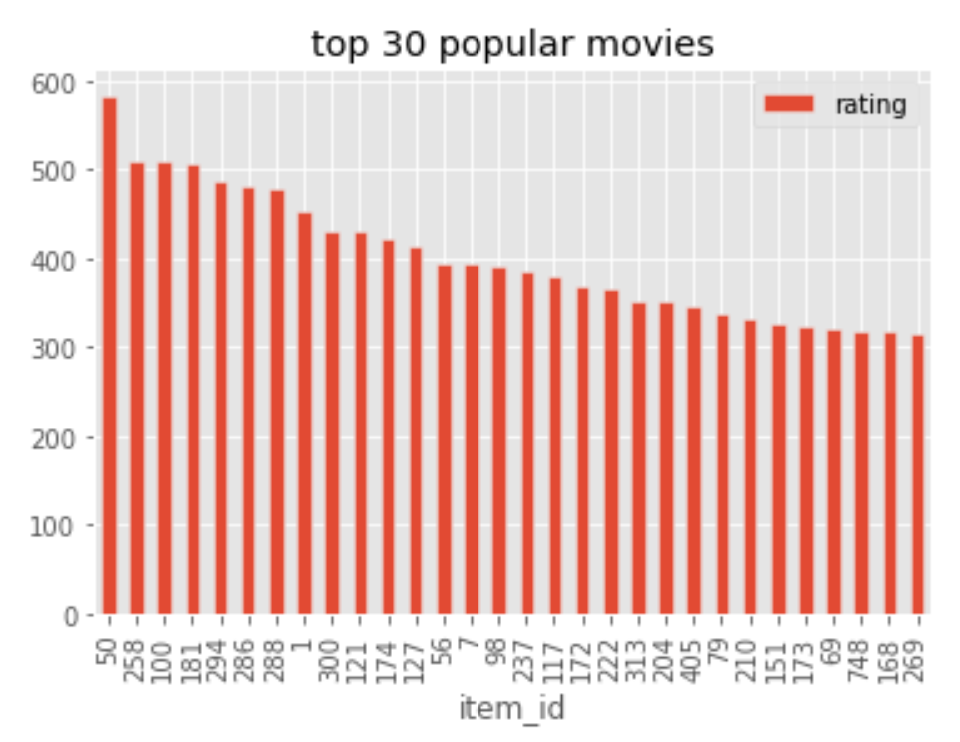
## 

## **Implementation/Recommendation**

**Popularity Based Recommendation**

* Popularity based recommendation system works with the trend. It basically uses the movies which are in trend right now. For example, if any movie which is usually watched by every new user then there are chances that it may suggest that movie to the test user.
* The problem with popularity based recommendation systems is that the personalization is not available with this method i.e. even though we know the behaviour of the user we cannot recommend movies accordingly**.**

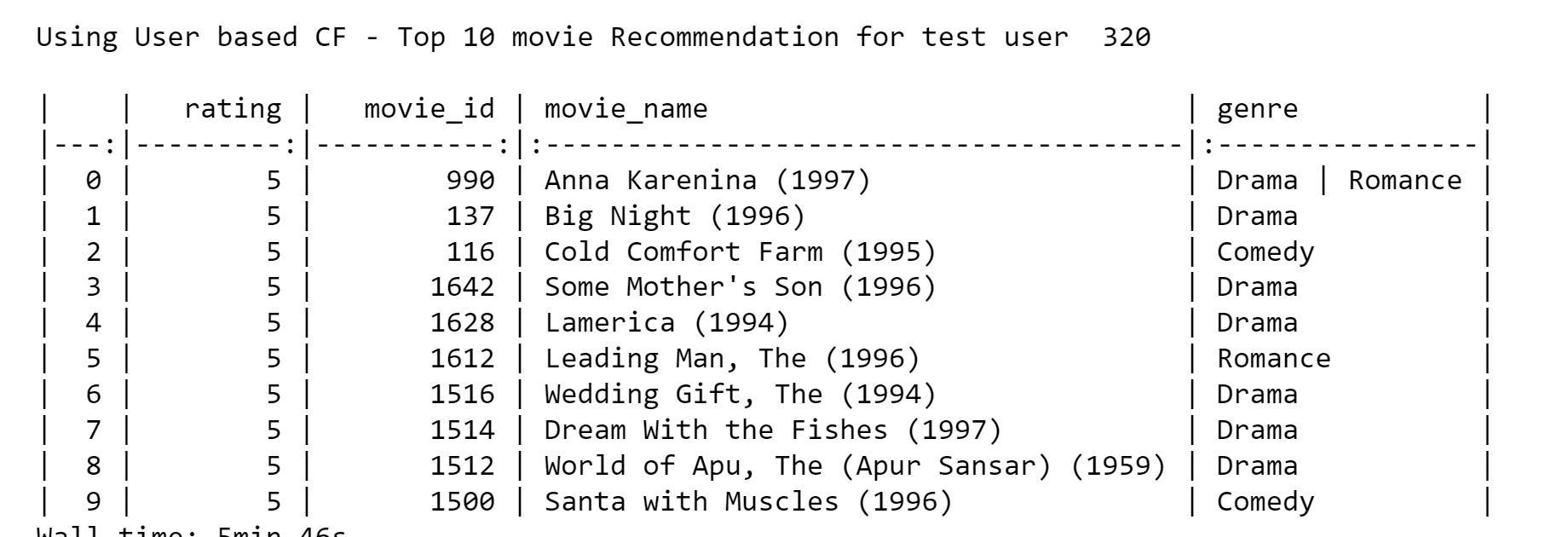
**Recommended top 30 movies**

****

**User based Collaborative Filtering**

It can be broken into 2 parts

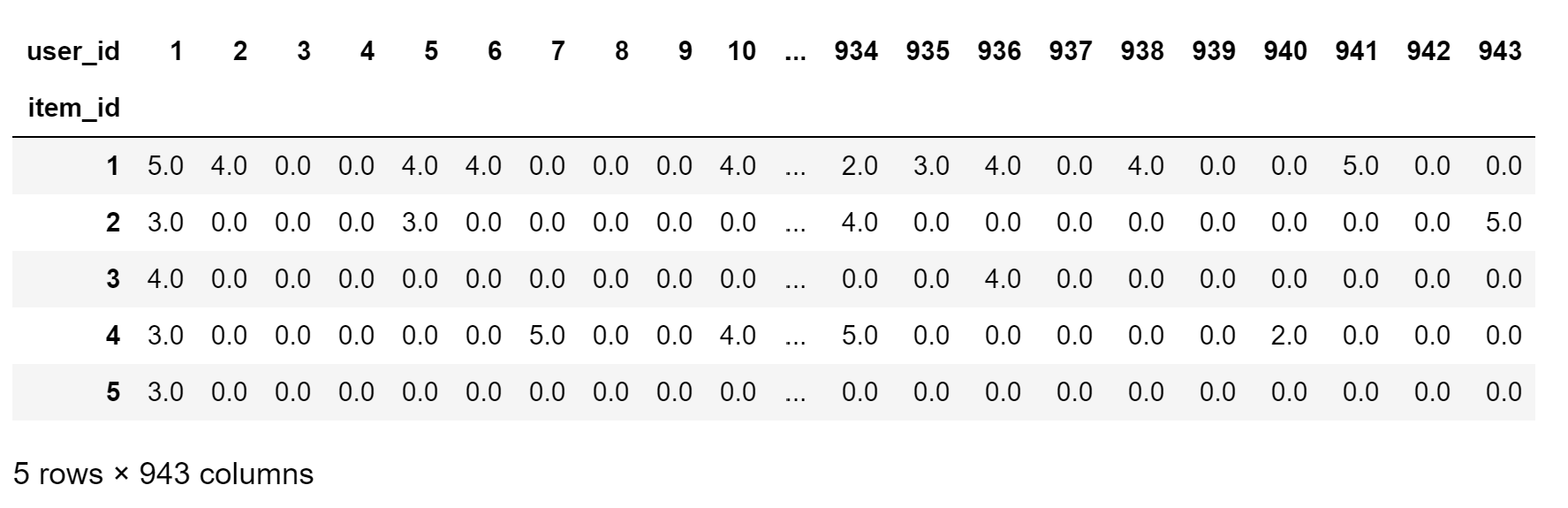
1. Get Most Similar users - here for a given user, we find the similar users based on their respective movie ratings
2. Recommend - For a given user id
   1. Get the movies not watched by the input user
   2. Get the ratings for those movies for the similar users
   3. Get the weighted similarity score
   4. Recommend Top\_K movies



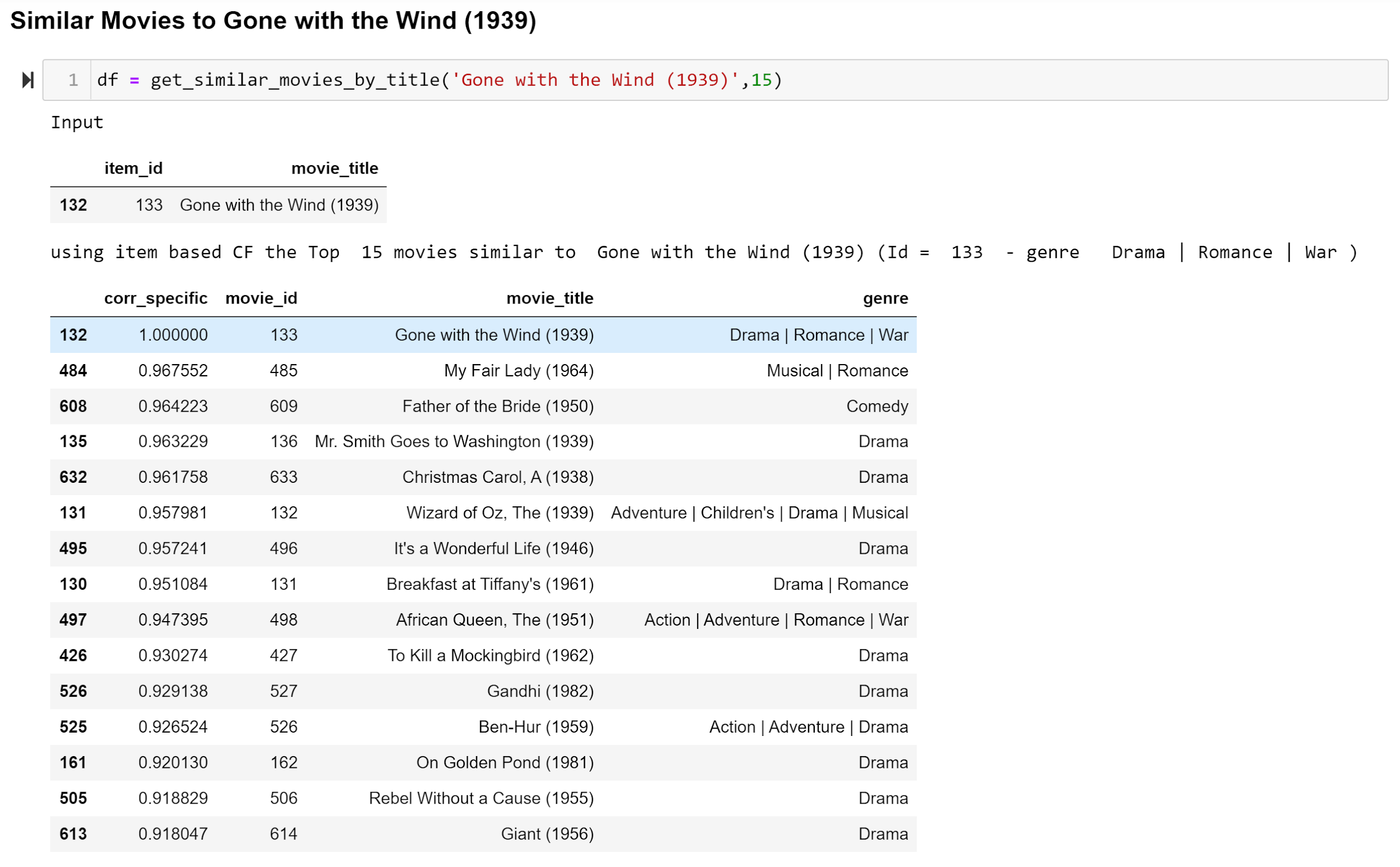
**Item Based Collaborative Filtering**

In Item based CF, the movies are compared against each other to recommend similar movies. Here we haved used **TruncatedSVD** for item based CF.

We created a pivot table with item x user and item rating data and then used **TruncatedSVD** for item based CF.



**Recommendation Result**



**KNN based recommendation**

We used cosine as the metric for KNN

We used (movie X user) and movie rating pivot table and then converted it to sparse matrix to handle the 0 raing value.

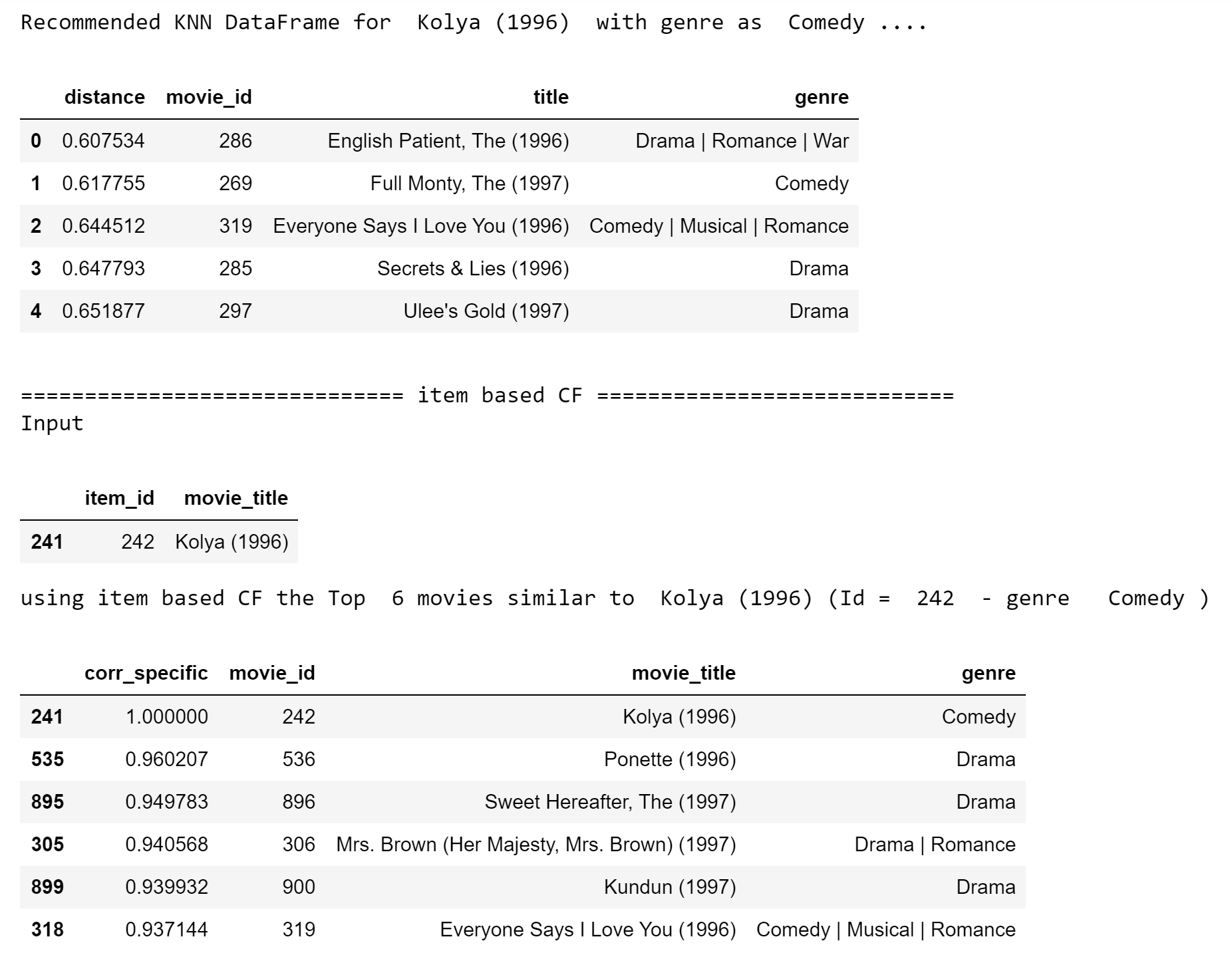


**Recommendation Result**



**Comparison of KNN v/s Item based (SVD) CF**

Here is the recommendation output for KNN and item based CF



**Future Improvements**

* Probably explore other libraries to make the predictions with reduced number of code
* Probably implement PCA to reduce dimension and improve performance
* Implement multi model based predictions