Movie Recommendation System

Jhansi Saketa B V  
 MS in Computer Science  
University of Colorado  
 Boulder, CO  
 jhbh9073@colorado.edu

Danielle Aras  
 BS in Computer Science  
 University of Colorado  
 Boulder, CO  
 danielle.aras@colorado.edu

Soumyadeb Maity  
 MS in Computer Science  
 University of Colorado  
 Boulder, CO  
 soma5722@colorado.edu

ABSTRACT

Movies are all time favorite entertainment media for everyone. Movie Recommendation systems are very popular these days, people like to get recommendations from other users, or based on the ratings of the movie. We built a Movie Recommendation System based on the datamining algorithms. We used different classification techniques like KNN, SVD and Clustering algorithm like K-means Clustering. Based on these algorithms we evaluated the performances and accuracy of these models, and tested them to obtain recommendations.

INTRODUCTION

Movies are one of the top all-time sources of entertainment. People enjoy watching, reviewing, and discussing them. For many people reviewing and rating movies is a passion. A movie can have fans and critics too. So, in a crowded entertainment market, movie streaming services and advertisers need to present viewers with the most relevant movies possible to maintain customer interest and loyalty.

Many companies now use data mining algorithms to create a recommendation system to match users with movies they would likely enjoy. A movie recommendation system can leverage data collected from users to identify patterns in an individual or group's viewing history, then use this insight to identify movies to recommend to a specific viewer. This project will use a database of user-submitted movie ratings to explore ways to generate movie recommendations and predict how users may rate future movies. Multiple algorithms will be used in order to compare their performance and practicality.

LITERATURE SURVEY

One of the better-known recommender algorithms is the Netflix Recommender System [1]. The algorithm originally began as a simple predictor of how many stars a user would rate a movie on a scale of one to five based on previous ratings but has grown into a multi-faceted system that generates many different types of recommendations according to other types of data Netflix now collects. For example, it can recommend titles that are popular among the entire user base, or movies that are similar to a single movie a viewer has watched based on a similarity algorithm.

Another system, MovieGEN, relies more on direct user input by asking viewers to respond to a list of questions and then generate recommendations based on their answers [2]. It uses both clustering and SVD to generate recommendations. Other systems use clustering to identify groups of users with similar tastes [3] or TF-IDF to find similar movies [4].

DATA SET

The Dataset that we are using for this project is from MovieLens Dataset. There are different sizes of the dataset provided, so we used the “Latest Full” dataset for the project which has 27 million data points. The “Latest Small” dataset can be used for testing the model which has nearly 100,000 data points. The data can be accessed in the URL: <https://grouplens.org/datasets/movielens/> with the permission of the MovieLens organization. We completed their contact form and successfully got permission to use the dataset in our project.

The dataset consists of reviews from 283,228 individual users. There are 58,098 movies and 18 genres: action, adventure, animation, children’s, comedy, crime, documentary, drama, fantasy, film-noir, horror, musical, mystery, romance, sci-fi, thriller, war, and western. Rating are on a scale from 0.5 to 5 in increments of 0.5.

The relevant csv files we used from the dataset were the movies.csv and ratings.csv files. The movies.csv file contained information about movies in the following columns:

* ‘movieId’, an integer key assigned to each movie
* ‘title’, a string of the movie’s title and release year
* ‘genres’, a pipe-separated string of genres applying to the movie

The ratings.csv file contained information about all user ratings, with columns:

* ‘userId’, an integer key assigned to each user
* ‘movieId’, an integer foreign key which corresponded to the same column in movies.csv
* ‘rating’, a numeric rating from 0.5 to 5
* ‘timestamp’, an integer representing the seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970 at the time the review was submitted

MAIN TECHNIQUES APPLIED

The MovieLens dataset was almost clean and could be used directly with some minimal pre-processing. We performed basic cleaning to check for null values (movies without genres, missing ratings, invalid ratings). We also combined the separate csv files for ratings and movies into a single table so ratings were linked to movie genres.

Recommendation systems generally will use two major techniques:

1. Content-based filtering uses algorithms which will identify similar items to items the user already likes i.e., items that have comparable properties independent of other users' data. A common algorithm for content-based filtering is “Term Frequency-Inverse Document Frequency" or TF-IDF. This is the most used algorithm to convert text into vectors and is widely used to extract features across various NLP applications but can also be applied to a recommender system. Another common algorithm is dot product / sum of product. For our movie recommendation system, content-based filtering will be used to identify movies that are similar to those that viewers have rated highly based on their genres.
2. Collaborative-based filtering uses data from other users to make recommendations to a given user. Methods include clustering, which groups similar users together and uses the top-rated movies from a user's cluster as recommendations, and classification algorithms like KNN and SVD.

The problem with a collaborative filtering is the cold start problem. Once the model is deployed, the main concern is that the system cannot draw any inferences for movies or users about which it has not yet gathered sufficient information.

Due to the large size and stable nature of our dataset, we have focused on collaborative-based filtering methods and sought to create an optimal movie recommendation system using techniques from previous recommender systems. While the Netflix recommendation system is highly developed and combines many different techniques, for our purposes and with the limited features in our dataset we focused on simpler algorithms. We also did not have the ability to use an interactive process like MovieGEN’s question system as we did not have direct access to the users in our dataset. Therefore, we focused on collaborative-based filtering using clustering and classification.

In Classification, we are using two main techniques.

1. SVD Algorithm: Singular Value Decomposition is an unsupervised and collaborative filtering algorithm which is one of the widely used algorithms for recommendation and Dimensionality reduction systems. The Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices. 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 factorization 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.

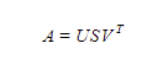


FIGURE 1

The factorization of this matrix is done by the singular value decomposition. It finds factors of matrices from the factorization of a high-level (user-item-rating) matrix. In Figure 1, The singular value decomposition is a method of decomposing a matrix into three other matrices as given below: Where A is a m x n utility matrix, U is a m x r orthogonal left singular matrix, which represents the relationship between users and latent factors, S is a r x r diagonal matrix, which describes the strength of each latent factor and V is a r x n diagonal right singular matrix, which indicates the similarity between items and latent factors. The latent factors here are the characteristics of the items, for example, the genre of the music. The SVD decreases the dimension of the utility matrix A by extracting its latent factors. It maps each user and each item into a r-dimensional latent space. This mapping facilitates a clear representation of relationships between users and items. [5]

1. K Nearest Neighbor: In previous method of collaborative filtering-based recommendation, we used SVD to recommend movie as per user ID or user’s other choices and ratings given to the other movies. In this method we tried to design an algorithm to suggest movie recommendation based on movie Id or movie title. Similar thing happens in Netflix search, when we search for a movie like “God Father”, it will show other movies as well apart from “God Father”. We need to find the similarity among movies. KNN or nearest neighbor is a very good choice in this category. We will use the movies.csv and ratings.csv for this nearest neighbor approach. While finding distances there are different approaches (ball\_tree, kd\_tree) to find the distances. However, we will be using the brute force method where the algorithm attempts to determine the best approach from the training data. Also, the distances among points are calculated using the cosine metrics. Cosine similarity determines angle between two vectors. We had the other option of Euclidean distance to measure the similarity but in high dimensional data cosine similarity will perform better.

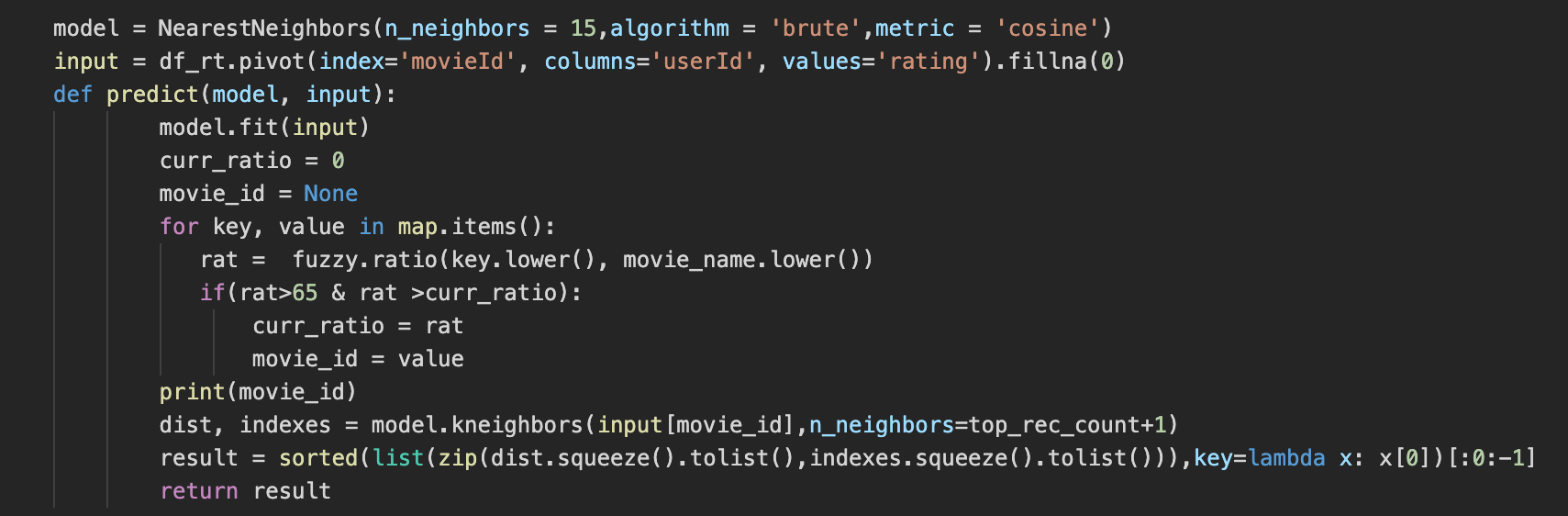


FIGURE 2

First, we are creating a sparse matrix where row will be the movieId, column will be userId and the value will be ratings. We will fit this matrix as input to the Nearest Neighbor model. Also, we will prepare a mapping for mapping movie Id to title. At the time of prediction, we will first find a movie title which is matching keywords to certain levels using a python library and this will be our test point. If we are searching for a movie which has exactly same title then it will be the starting point otherwise other movies which has similar name. Like if we search for “God Father 2” and that movie is not available then “God Father” will be found as a starting point. In the next step we will predict nearest movies which is similar to this starting point using the previously prepared Nearest Neighbor model. Once we find the nearest movies and their ids, we will use the previously prepared map to find the corresponding movie title to show the final result.

1. K-Means Clustering: We used the K-means algorithm for clustering. This generates a specified number of centroids and repeatedly assigns the data points to the cluster with the nearest centroid, updating the centroids after each iteration using the new mean of each cluster. The process terminates when it reaches the specified number of iterations, or the centroids reach convergence where the change between iterations is within a minimal bound. The output is a list of each data point’s assigned cluster.

We used K-means clustering to divide reviewers into clusters with similar taste in movies based on their average ratings of different movie genres. Clusters could then be used to predict a user’s rating for a movie they had not seen yet based on the cluster average rating. “Top” movies for each cluster could be found by identifying the highest-rated movies for each cluster.

Due to the sparseness of the dataset (most users only reviewed a small fraction of the thousands of movies) we chose to consider the user’s average ratings per each genre rather than individual movie ratings. We grouped the ratings data by genre for each user, using the mean of all ratings for movies categorized in the given genres. Instead of using the numerical average ratings for each genre, we found the using a ‘like/dislike’ Boolean resulted in the best model. The average rating across all users and genres was around 3.5, so we considered ratings of 3.5 and over as a ‘like’ denoted by 1, and ratings below 3.5 a ‘dislike’ denoted by 0.

We performed feature selection to reduce the number of genres we would consider in the clustering model. We eliminated genres with too few reviews or high correlations to other genres. Different combinations of genres were tested to refine the model. We also tested different numbers of clusters.

The final k-means model divided the users into 16 clusters of similar users. We used this model and the users’ cluster assignments to create a function that would return the ‘n’ top rated movies within a given cluster. To reduce the potential for movies with few ratings to skew the recommendations, the function also took a parameter for a minimum number of ratings and would disregard movies that did not meet this threshold within the cluster.

EVALUATION METHODS

For clustering, the fit of the clusters was evaluated by the inertia, or sum of the squared error (SSE), and the silhouette coefficient. Inertia or SSE is a measure of the distance of points within each cluster from the cluster centroid. The lower the inertia, the more similar the users within each cluster. A high inertia suggests that the users in the same similar segment have a high degree of differences between them and may not be a true (or usable) cluster. We will also look at the silhouette coefficient to judge how well-separated our clusters are. The silhouette coefficient ranges from 1 to –1. Positive values mean clusters are more distinct and better defined, while negative values mean there may be confusion or incorrect assignments to clusters.

K-means clustering requires that the number of clusters be specified. The optimum number of clusters were chosen based on the ‘elbow method’ which is based on the inertia, and silhouette coefficient.

Chart, line chart

Description automatically generated

FIGURE 3

Chart

Description automatically generated

FIGURE 4

Figure 3 is an elbow plot of the inertias for a model using from 2 to 25 clusters and 9 genres ('Action', 'Adventure', 'Comedy', 'Crime', 'Drama', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller') and the reduced dataset. Figure 4 is the same type of plot with the added genres 'Animation', 'Fantasy', 'Horror', 'Musical', 'War', and 'Western'. Note there is not a clearly defined ‘elbow’ or point where the inertia begins to level off on either plot, which was true of all the elbow plots we evaluated for this dataset. The plots began to level off slowly around 10-12 clusters, but to get more insight into the number of clusters to use we turned to the silhouette coefficient.

Chart, line chart

Description automatically generated

FIGURE 4

Figure 4 shows a plot of the silhouette coefficients of k-means clustering using 3, 9, and 15 genres based on the reduced dataset and using full average data as opposed to the like/dislike Boolean system. The results of changing the number of genres were slightly unpredictable, with the highest scores coming from 3 genres, middling scores from 15 genres, and the lowest from 9 genres. The small size of the reduced dataset was likely to blame for this, as testing different genre combinations with the full dataset yielded more consistent results.

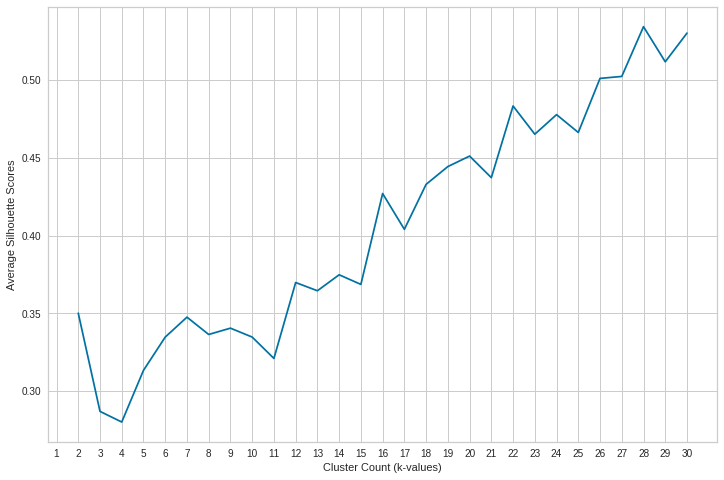


FIGURE 5

Figure 5 shows the plot of the silhouette coefficients for a model using the full dataset and 8 genres (‘Action’, ‘Comedy’, ‘Crime’, ‘Drama’, ‘Horror’, ‘Romance’, ‘Sci-Fi’, ‘Thriller’). It is also after we switched to the Boolean rating system rather than numeric averages. Clearly using this system results in greater benefits from increasing the number of clusters. Instead of reducing the silhouette scores, with this model higher numbers of clusters increase the score.

Chart

Description automatically generated

FIGURE 6

Using these results, we narrowed in on a number of clusters between 12 and 20. We next considered silhouette plots (exemplified in Figures 6 and 7) for a more detailed look at the clusters generated.

Chart

Description automatically generated

FIGURE 7

Figure 6 shows the silhouette coefficients of points from 12 clusters, separated by color, while Figure 7 shows 16 clusters. The average silhouette score is denoted by the red dotted line. Although the number of clusters is higher and the sizes more varied in Figure 7, the average silhouette coefficient is higher and there are less points with negative coefficients which suggests most points are correctly placed in their clusters. The size of the clusters is still within reasonable bounds, so we would expect to have enough users in each cluster to get meaningful recommendations based on the movies they have reviewed.

Looking at both the silhouette score and the inertia/elbow plots, there was a clear trade-off between number of clusters and fit/definition of the clusters. Higher numbers of clusters resulted in more processing time to generate models, and may result in clusters too small to be useful if there are not enough users to draw reviews from. However, more clusters could also create a better model with more distinct clusters whose members were more similar. Given these competing concerns, we chose to take a balanced approach and used 16 clusters for our final model.

Evaluation of the SVD Algorithm

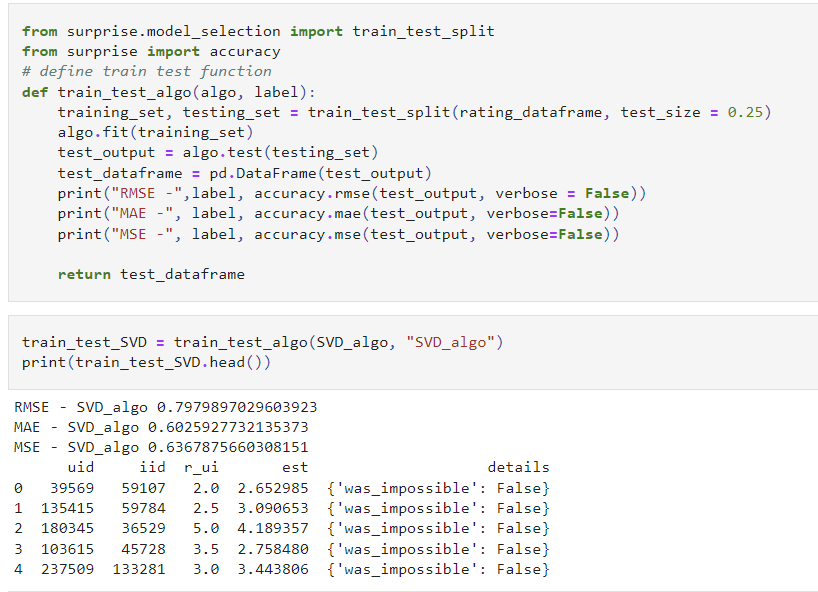


FIGURE 8

For evaluating the SVD algorithm, we use train-test split evaluation, in which we split dataset into 75% for training and 25% for testing. The above figure 8 shows the Root mean square error, mean absolute error, and Mean squared error for the SVD algorithm.

TOOLS

The following tools were used to develop the project.

1. Python

Python programming language is the core tool to build our project. The data extraction, data cleaning, pre-processing and all the models and algorithms were built using python programming language.

1. Pandas

We used the Pandas Python module to work with the dataset. The major role of Pandas in our project is extraction of the data, creating data frames, and working with the data frames.

1. Scikit

Scikit is a Python library that contains various classification, regression, and clustering algorithms. We used the clustering, KNN, and SVD modules in particular.

1. Matplotlib

Matplotlib is a plotting library for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language and its numerical mathematics extension [NumPy](https://en.wikipedia.org/wiki/NumPy). In our project, we use matplotlib to plot the graphs or visualize the results.

1. Yellowbrick

Yellowbrick is a Python library that contains visualization methods. We used the Silhouette Visualizer to generate silhouette plots of the clusters in K-means

RESULTS

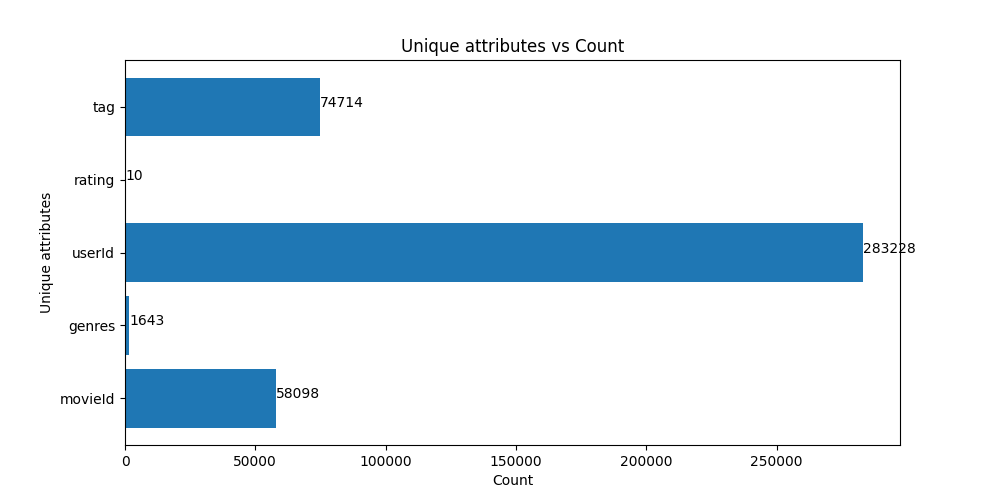


FIGURE 9

Figure 9 shows the Distinct count of each unique attribute like, number of tags, rating, userID, genres, movies. The Dataset used Latest Full 27 Million data points.

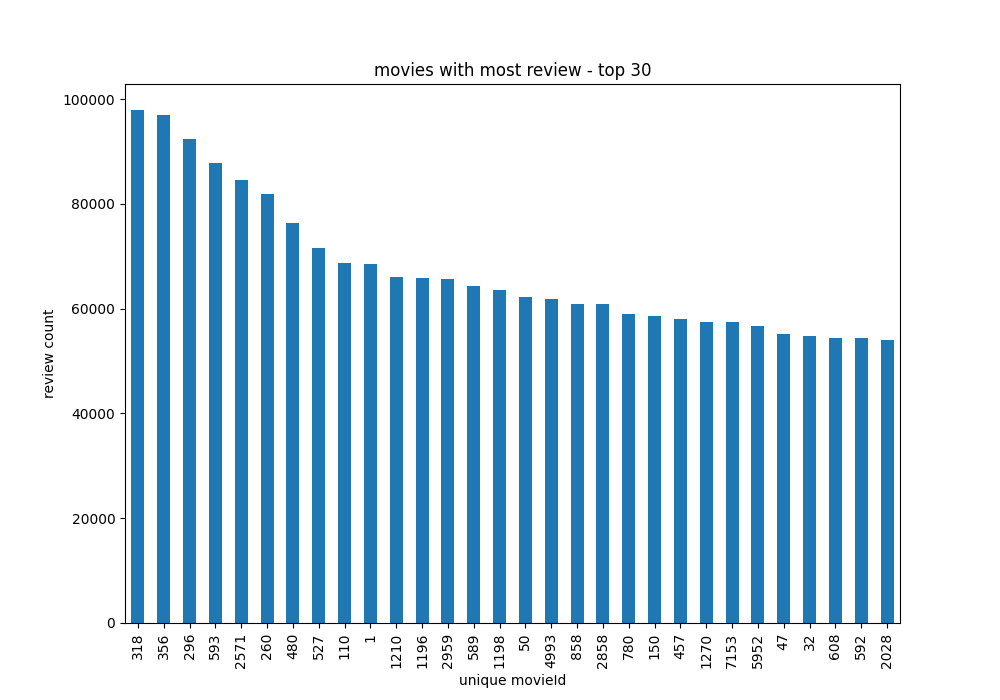


FIGURE 10

Figure 10 is a histogram of the top 30 most-reviewed movies from the full dataset.

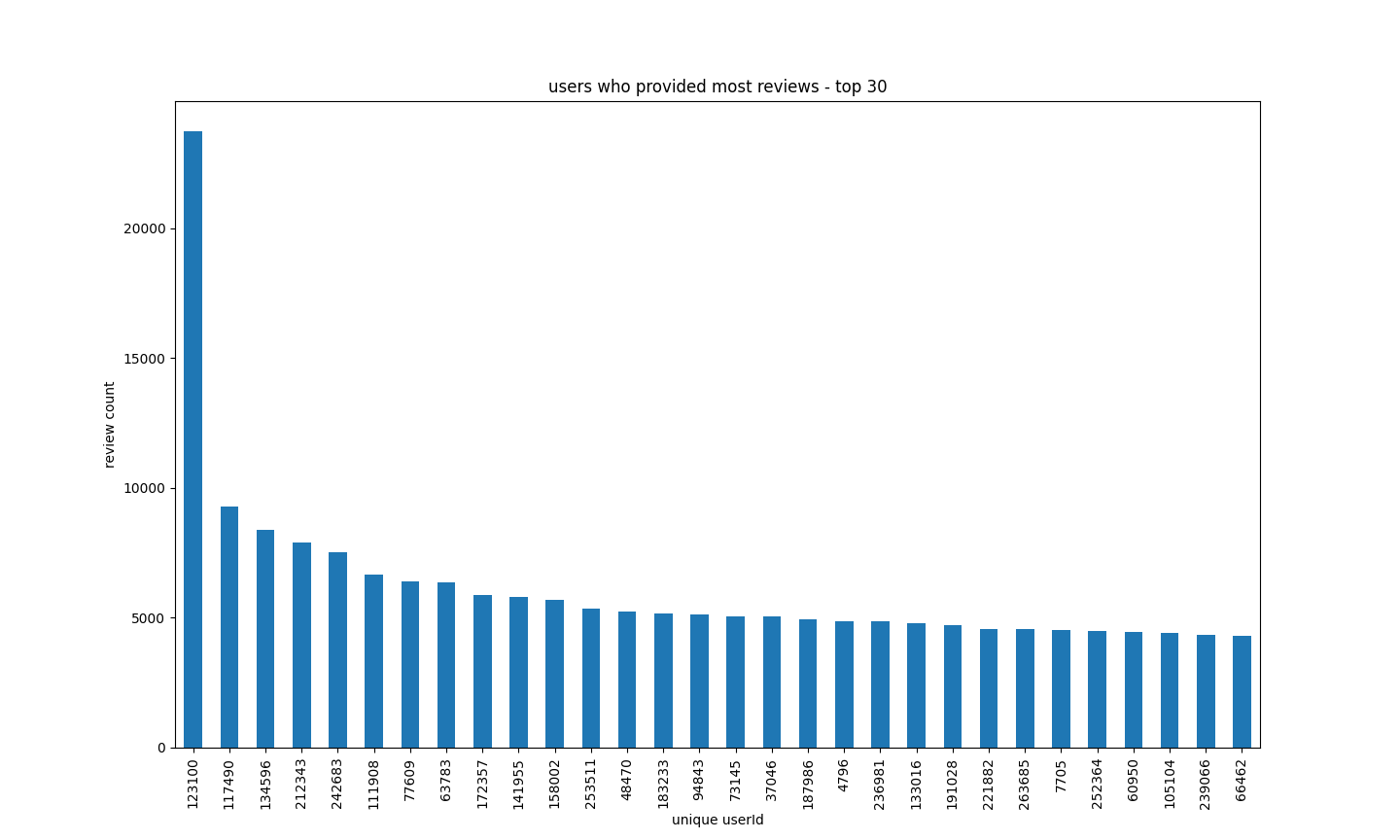


FIGURE 11

Figure 11 is a histogram of the top 30 users with the most reviews

SVD Results

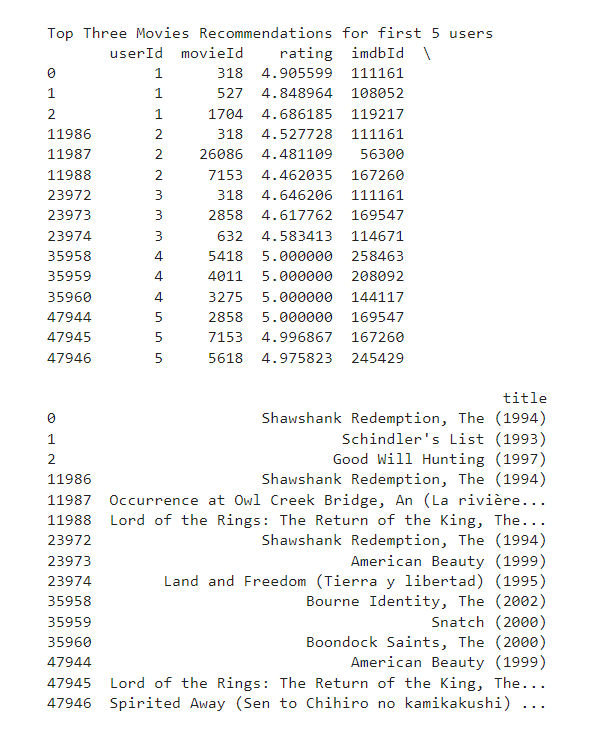


FIGURE 12

Figure 12 shows the top three movies recommended for first 5 users using the Singular Value Decomposition algorithm. Here we used the rating.csv file to train the SVD algorithm. We split dataset into 75% for training and 25% for testing. After the model is built, we created a rating prediction data frame in which it predicts the ratings for each user for all the movies. Using the SVD algorithm, the rating of the combination of userId and movieId is predicted on the scale of 0 to 5. Then the prediction results are linked to movie data frame(movie.csv – title) and link data frame (link.csv - imdbID) based on the movieId attribute to merge the data frames. The top recommendations are sorted by the top ratings by the Users. In this way, SVD algorithm gives the top recommendations by all the users.

KNN Results

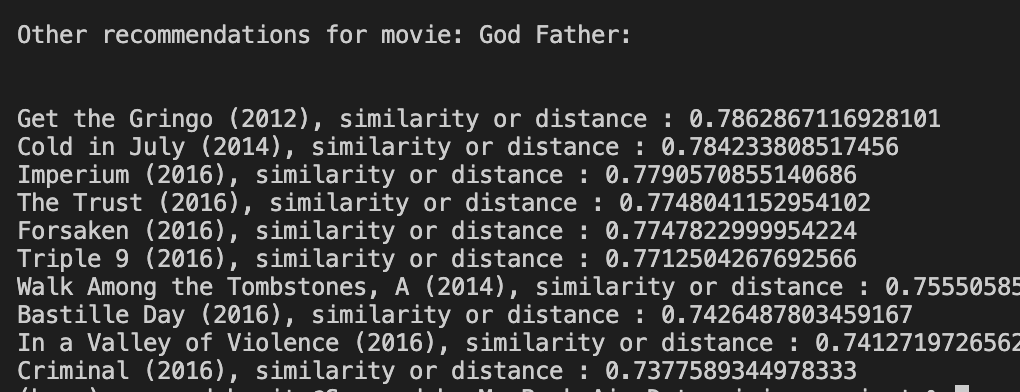


FIGURE 13

Figure 13 shows the top 10 movies recommended by a collaborative filter based recommend model when user searched for a movie name “God Father”. Distance values shows a similarity between the recommended movies and the input movie name “God Father”. We can see that the recommended movies are also crime mafia related, similar to God Father.

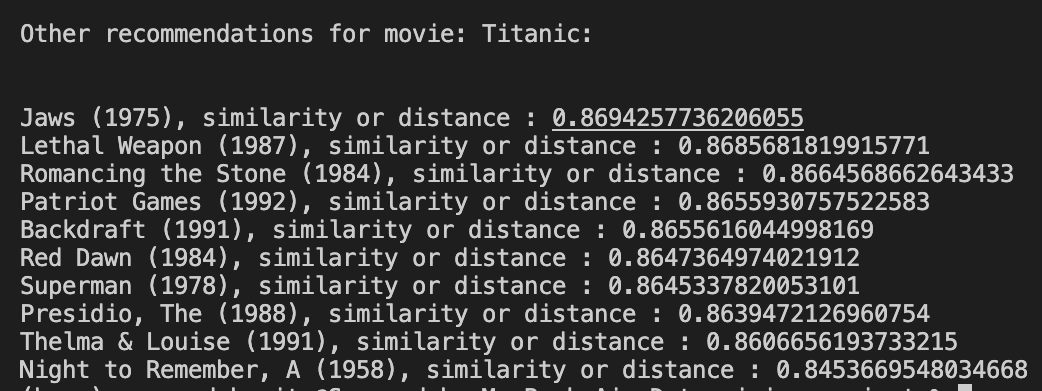


FIGURE 14

Figure 14 also shows similar recommendation for movie titanic. This time the model is showing sea related like Jaws or romantic movies.

Clustering Results

Our final clustering model was based on 16 clusters.

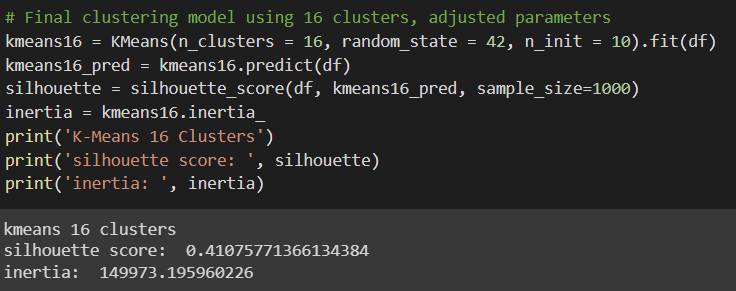


FIGURE 15

Figure 15 shows the parameters used to generate the model. The random\_state was set to ensure the results could be repeated. The silhouette coefficient was approximately 0.4 (this is based on a random sample so the exact score varied). The inertia was 149973, a value in the middle of our tested models.



FIGURE 16

Using our model, we generated a sample recommendation list for users in cluster 11. These were obtained by using our function to get the top ten movies with 50 or more reviews in the given cluster. This function could be expanded to return recommendations from a given genre, time period, or any number of other specialized recommendations.

REFERENCES

*1] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix recommender system: Algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. 6, 4, Article 13 (December 2015), 19 pages. DOI: http://dx.doi.org/10.1145/2843948*

*[2] Eyrun A. Eyjolfsdottir, Gaurangi Tilak, Nan Li (2008), “MovieGEN: A Movie Recommendation System”, 2008 Conference Proceedings.*

*[3] Roman, Victor (2019), “Unsupervised Classification Project: Building a Movie Recommender with Clustering Analysis and K-Means”, Towards Data Science, https://towardsdatascience.com/unsupervised-classification-project-building-a-movie-recommender-with-clustering-analysis-and-4bab0738efe6*

*[4] Nixon, Alex Escola (2020), "Building a movie content based recommender using tf-idf", Towards Data Science,* [*https://towardsdatascience.com/content-based-recommender-systems-28a1dbd858f5*](https://towardsdatascience.com/content-based-recommender-systems-28a1dbd858f5)

*[5] https://analyticsindiamag.com/singular-value-decomposition-svd-application-recommender-system/#:~:text=In%20the%20context%20of%20the%20recommender%20system%2C%20the,ratings%20that%20are%20given%20to%20items%20by%20users.*