INFO 7390 FINAL PROJECT REPORT

Project Topic:

Movie Recommendation System

Team members (Group-23):

1. Ameya Adwait Ranade 1561048
2. Manisha Bagora 2199292
3. Parul Vig 1356967
4. Neelam Dighe 2989348

Problem Statement:

Recommendation Systems have become very popular with the increasing availability of millions of products online such as movies, music, books etc. Movie recommendation System has been a surge in the usage of the users opting for online streaming services. MRS research continues to face significant obstacles even if today's MRSs significantly assist consumers in finding interesting Movie in these vast archives. Developing, implementing, and evaluating recommendation techniques that utilize knowledge beyond a basic metric and parameter to suggest a recommendation is challenging. In this project, we examine the different algorithms for overcoming these difficulties, then accordingly we come up with the recommendation system.

Data Set Description:

- This dataset (ml-25m) describes 5-star rating and free-text tagging activity from [MovieLens] (http://movielens.org), a movie recommendation service.
- It contains 25000095 ratings and 1093360 tag applications across 62423 movies. These data were created by 162541 users between January 09, 1995 and November 21, 2019.
- This dataset was generated on November 21, 2019.Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.
- The data are contained in the files `genome-scores.csv`, `genome-tags.csv`, `links.csv`, `movies.csv`, `ratings.csv` and `tags.csv`. More details about the contents and use of all these files follows.

Approach & Solution:

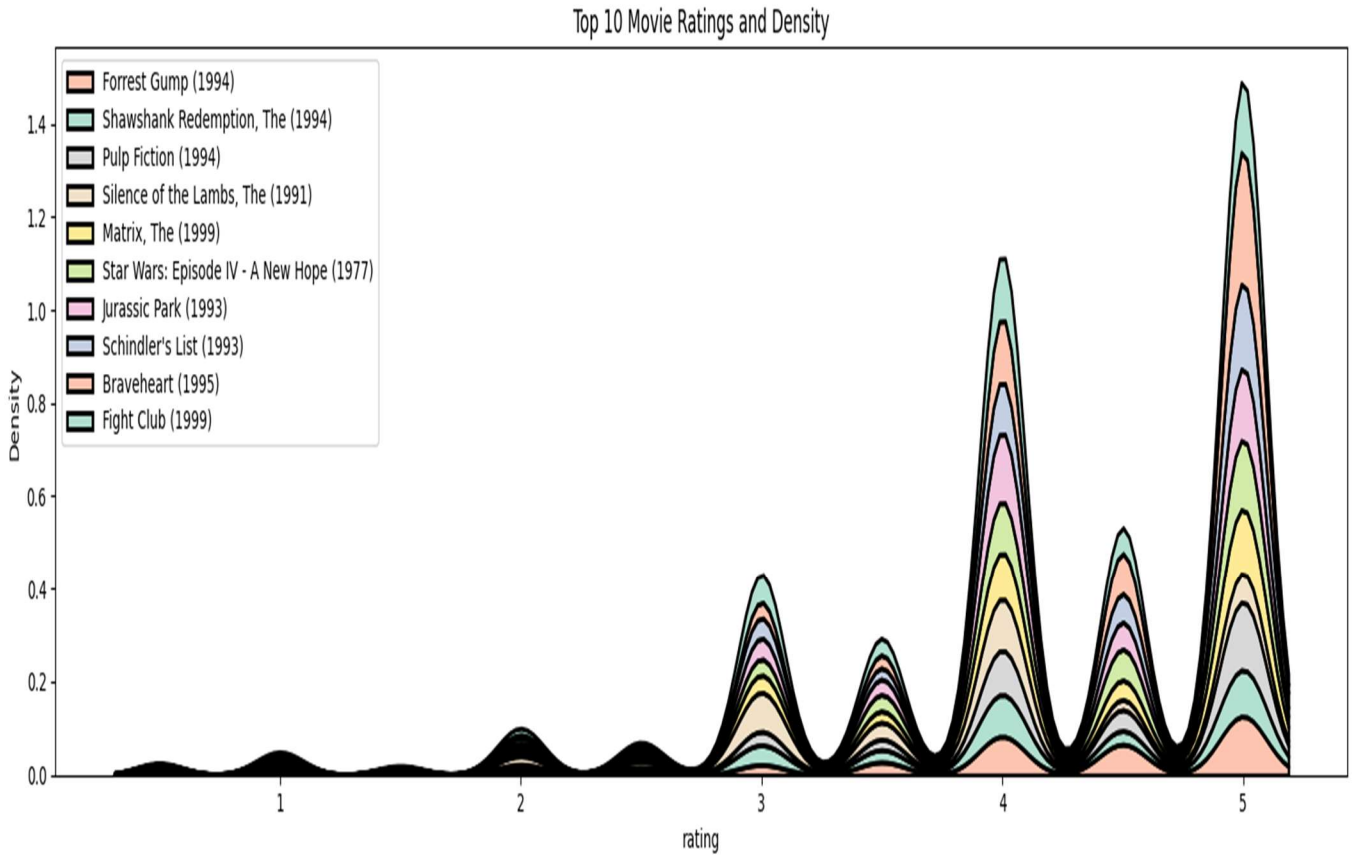
1. Once we read the data set, we first perform some data cleaning and preprocessing, like checking null values, merging files, creation of data frame, sorting, calculating average rating
2. After that we performed EDA to have a clear picture of it and understand the features within the dataset with visualizations
3. Initially we are using a **k-nearest neighbors (k-NN) algorithm.** It begins by loading the movie and ratings data from two CSV files, movies.csv and ratings.csv, using the panda's library.
4. The two data frames are merged on the movieId column to create a combined data frame of movie information and user ratings. The average rating for each movie is calculated and sorted in descending order. This data frame is then used in later stages of the recommendation system. Next, the average rating for each movie is calculated, and the number of ratings for each movie is counted. These two data frames are then merged on the title column. The data frame is then filtered to include only those movies with a minimum number of ratings, as determined by the popularity threshold variable.
5. The filtered data frame is then transformed into a matrix of movie ratings by user, with movies as rows and users as the columns. This matrix is then converted into a sparse matrix for more efficient computation. The k-NN model is then trained on this matrix using the cosine distance metric and the brute algorithm.
6. Finally, the user is prompted to input a movie title. The model finds the k=6 nearest neighbors of this movie and recommends these movies to the user.
7. **User Based Collaborative Filtering:** It takes a random user's ID as an input, along with a data frame containing movie ratings by users, a ratio (which represents the percentage of movies that must have been rated by other users to be considered for recommendation), a correlation threshold for selecting similar users, and a minimum rating score for a movie to be recommended. The function first extracts the movies that the random user has rated. It then selects those users who have rated enough of those movies (as determined by the ratio parameter). The Pearson correlation coefficient is calculated between the random user and these similar users, and those users with a sufficiently high correlation are selected. The function then calculates a weighted average of the ratings of these similar users for each movie and recommends movies with a rating greater than the specified minimum score.
8. **Singular Value Decomposition (SVD) using hyper parameter tuning**: It uses surprise library to implement SVD algorithm on movielens dataset. We have used a smaller dataset (ml-1m) here to fit algorithm. It uses train_test_split to split data to start with it. Then, it make predictions using SVD algorithm and calculates RMSE . We keep on changing n_factors as a part of hyperparameter tuning to get best RMSE score. And then plot a graph for RMSE and n_factors.
9. **Item based collaborative filtering:** Item collaborative filtering is a type of recommendation system that is based on the similarity between items calculated using the rating users have given to items. It helps solve issues that user-based collaborative filters suffer from such as when the system has many items with fewer items rated. Function => get_other_movies accepts a movie name as a parameter. It creates a data frame for that movie and group it by title. Then it merges that data frame with ratings dataset on userId column. Then we further group it by title and get user count. We also calculate percentage of who watched.

10. **Matrix Factorization using TruncatedSVD:** We start by dropping null values of title column and then build a dataset movie rating Count to get movies with their rating count. Merging it again to get user rating count dataset to build a pivot table to perform matrix factorization algo. Then further fitting it using Truncated SVD algo and making a correlation matrix, then getting list of recommended movies.

11. **Logistic Regression and RandomForestClassifier:** These two algorithms we have tried to predict movie ratings. We start by splitting data in to training and test and then perform Logistic Regression to predict movie ratings. We use GridSearchCV as a part of hyperparameter tuning and calculate best accuracy score. For Random Forest Classifier, we use different number if estimators as a hyperparameter tuning to get best accuracy score.

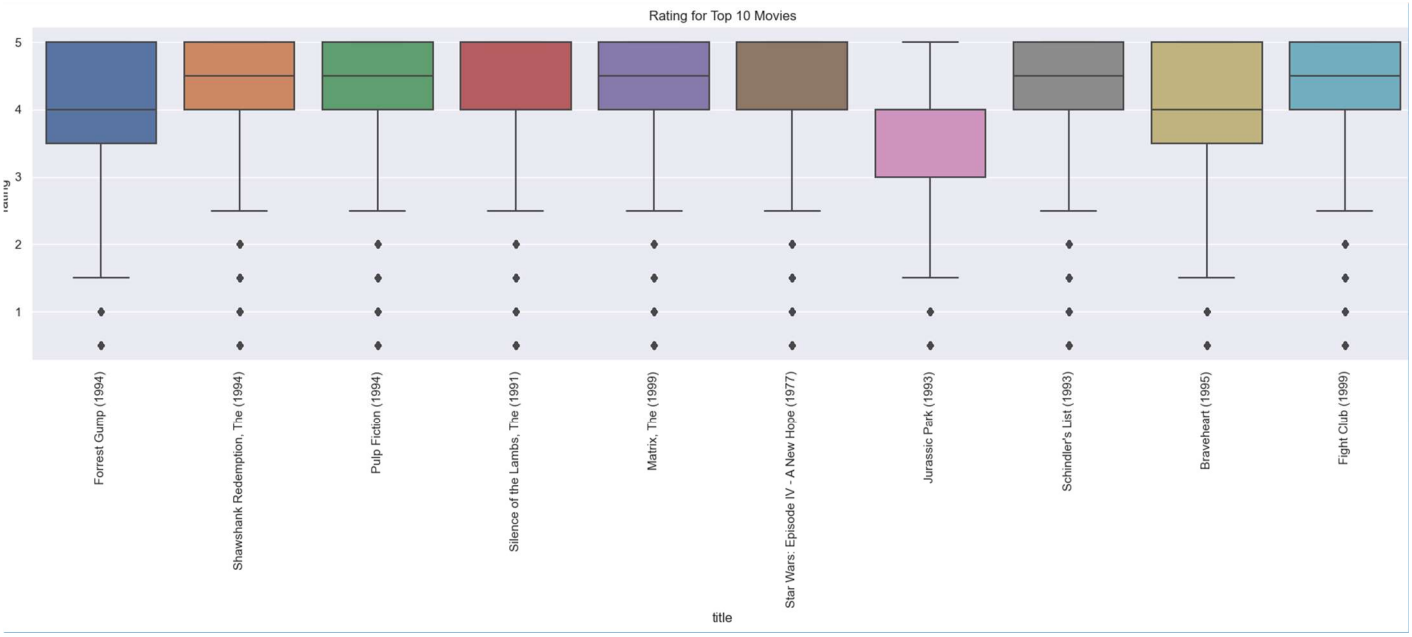
Visualization:

KDE Plot- Top 10 Movie Rating

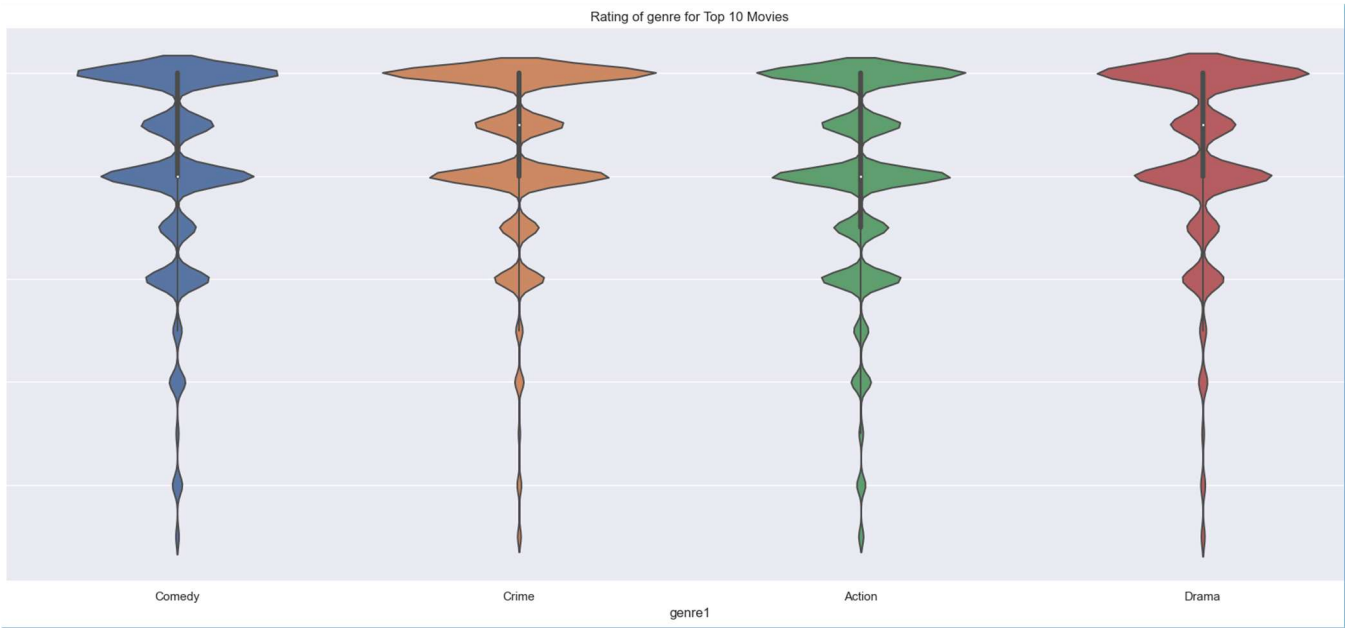


World Cloud- Based on Movie Tags

Box Plot- Top 10 Movie Rating



Violinplot- Genre Rating



Output:

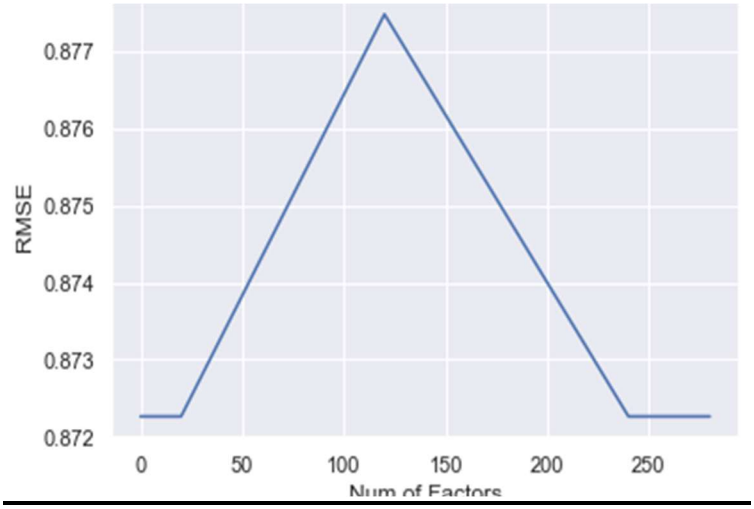
Logistic Regression- Accuracy

```
In [35]: best_clf.best_estimator_

Out[35]: LogisticRegression(C=0.08858667904100823, max_iter=1000, penalty='l1',
                            solver='liblinear')

In [36]: print (f'Accuracy - : {best_clf.score(X,y):.3f}')

         Accuracy - : 0.288
```

Recommended Movie

```
Recommendations for Salem's Lot (1979):

1: Let's Scare Jessica to Death (1971), with distance of 0.7544864416122437:
2: Sentinel, The (1977), with distance of 0.7555797696113586:
3: Magic (1978), with distance of 0.757941722869873:
4: Silver Bullet (Stephen King's Silver Bullet) (1985), with distance of 0.7692347764968872:
5: It's Alive (1974), with distance of 0.7699768543243408:
```

RMSE- Number of Factors



Conclusion:

Recommender system has become increasingly imperative since the data over-burden. In this project, we examine the different algorithms for overcoming these difficulties. We have implemented different algorithms like KNN Model, User Based Collaborative Filtering, Matrix Factorization, Logistic regression, Random Forecast Classifier & Item based collaborative filtering, Truncated SVD and accordingly we come up with the recommendation system. Based on the all the implementation of Models we have found out that Collaborative Filtering, KNN & SVD seems to be best approach for recommendation systems. We can do more research on Hyper parameter tuning and other machine algorithms to come up with more advanced recommendations systems.