Movie Recommendation System

DATA 602 Final Project

Project By:

1) Ankit Kothari	117754612
2) Nihar Pandya	117533170
3) Rahul Walia	117359619
4) Harshiel Bhaavesh Shah	118183060

Introduction

The entertainment industry piqued our interest from the start, and with the advent of various OTT Platforms in the past few years, we decided to work on movies. As the name suggests, our project is a movie recommendations system. In this digital era, especially with the COVID impact, the importance of online recommendations, be it in terms of products or services, has exponentially increased. Thus, we decided to build a recommendation system. Our system can recommend movies based on either similarity to one particular movie or based on similarity of one user to another.

Data Sourcing and Engineering

The data we used is from Kaggle, and we used the following sources to understand Movie Recommendation System and data cleaning better. It has the following features.

About the Data:

(This following table and definition is taken from the internet source)

https://www.kaggle.com/rounakbanik/movie-recommender-systems/data

Features:

- adult: Indicates if the movie is X-Rated or Adult.
- belongs_to_collection: A stringified dictionary that gives information on the movie series the particular film belongs to.
- budget: The budget of the movie in dollars.
- genres: A stringified list of dictionaries that list all the genres associated with the movie.
- homepage: The Official Homepage of the move.
- id: The ID of the movie.
- imdb_id: The IMDB ID of the movie.
- original_language: The language in which the movie was originally shot in.
- original_title: The original title of the movie.
- overview: A brief blurb of the movie.
- popularity: The Popularity Score assigned by IMDB.
- poster_path: The URL of the poster image.
- production_companies: A stringified list of production companies involved with the making of the movie.
- production_countries: A stringified list of countries where the movie was shot/produced in.
- release_date: Theatrical Release Date of the movie.
- revenue: The total revenue of the movie in dollars.
- runtime: The runtime of the movie in minutes.
- spoken_languages: A stringified list of spoken languages in the film.

- status: The status of the movie (Released, To Be Released, Announced, etc.)
- tagline: The tagline of the movie.
- title: The Official Title of the movie.
- video: Indicates if there is a video present of the movie with TMDB.
- vote_average: The average rating of the movie.
- vote_count: The number of votes by users, as counted by TMDB.

Data Sources:

https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

https://www.kaggle.com/rounakbanik/movie-recommender-systems/data

https://stackoverflow.com/questions/45312377/how-to-one-hot-encode-from-a-pandas-column-containing-a-list

https://appdividend.com/2019/11/12/how-to-convert-python-string-to-list-example/

https://medium.com/kunalrdeshmukh/collaborative-filtering-in-recommendation-systems-2fa49be8f518

https://pandas.pydata.org/docs/

Data Cleaning and Techniques used to clean bad data:

- Column: Tagline
    - It has 25000 null values
- Column: Spoken Language
    - We have the same information in the original language column
- Column: Original_Title
    - We have the same information from the title column
- Column: Status
    - All movies in the database are released movies
- Column: Homepage, Poster_path, Adult, Belongs_to_collection, Video
    - It does not provide helpful information to be used as features
- Converting Date to only Year
- Converting imdb_id from string to numeric datatype
- Converting Production_companies and genres columns to string and then extracting the relevant information
- Dropping rows with null values revenue, original_language, budget and release_date
- ImDb_ID could potentially be important thus filing the missing values with 0
- Runtime deemed to be an essential feature, the missing data for runtime was filled with the median value
- Genre and Production_Countries contains lists so they are exploded and one hot encoded for easier manipulation and clustering
    - Production_countries had terrible data. To further impute it, we created a dictionary mapping language to country and since we have data for all languages used that dictionary to assign a country to the movie.

- We intend to provide the latest recommendations and more well-known movies worldwide. So we did the following:
    - Filtered movies_by_country to contain countries with 500 or more movies
    - Filtered movies by year to include movies during or after 2000
- After Exploratory Data Analysis we dropped all irrelevant features for algorithms such as id, imdb_id, title, overview, production_companies, revenue, budget, movies_by_countries, release date, and original_language

Algorithms Used

Part 1: Recommendation using K Means Clustering:

Our purpose was to provide recommendations based on similarity, and we didn't have any labeled data for this as the similarity is a measure/estimate but no true value. Thus the problem is unsupervised learning, and we chose K means to solve it. The reason for it is, it is quite accurate, simple, commonly used, and fast.

We ran K-means clustering with various cluster values, allocated them in clusters, and calculated the cost. We then used the Elbow method to plot the cost values against the number of clusters to figure out the elbow point that would recommend the ideal number of clusters. From that, we determined the ideal number of clusters to be 40. We then calculated the Silhouette Score for all numbers of clusters; Silhouette Scores tell us the goodness of the clusters with values ranging from -1 to 1, -1 being extremely poor or wrong clustering and one being exact or perfect clustering. We got a value of 0.25 at 40 clusters, and the value didn't improve drastically with an increase in clusters, so we went ahead with 40 clusters.

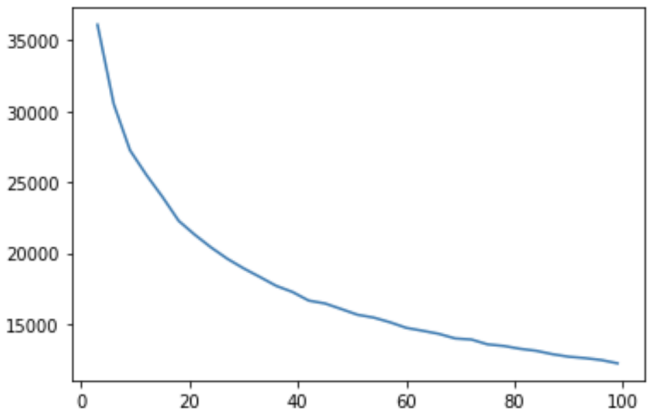
Elbow method:

4.1.1 Elbow Method

```
plt.plot(kinitial,inertia)
```

executed in 222ms, finished 14:04:11 2021-12-15

```
[<matplotlib.lines.Line2D at 0x7f85f94f61f0>]
```



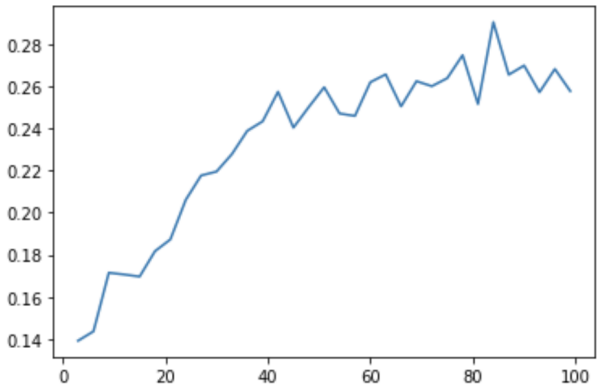
Silhouette Score:

4.1.2 Silhouette Score

```
plt.plot(kinitial,scoree)
```

executed in 205ms, finished 14:04:11 2021-12-15

```
[<matplotlib.lines.Line2D at 0x7f85f9b2cfd0>]
```



Part 2: Recommendation using Collaborative Filtering:

Our purpose was to provide recommendations based on user-to-user similarity, and we didn't have any labeled data for this as the similarity is a measure/estimate but no true value. Therefore, the User-to-User Collaborative Filtering method was used, since it is the future work, the Item-to-Item C.F. is also a good method to be implemented since usually it provides better results than a lot of others but since even after cleaning and scaling the data wasn't the best suitable, therefore, User-to-User C.F. method was finalized.

Since the dataset(divided into 2 separate files) had data about movies and data about the users(including their ratings) therefore after taking the userId as an input we were able to fetch its data into another variable(to use it in the later stage). Then after merging the datasets all the columns except for ratings, userId, movieId, title was removed. Then came the biggest challenge of deciding which method of correlation matrix user be used from:

1. **Pearson correlation coefficient (Pearson)**
2. **Kendall rank correlation coefficient (Kendall)**
3. **Spearman's rank correlation coefficient (Spearman)**

Since both Spearman and Kendall are based on distance and not exactly the projection of one another, Pearson gave us the best result.

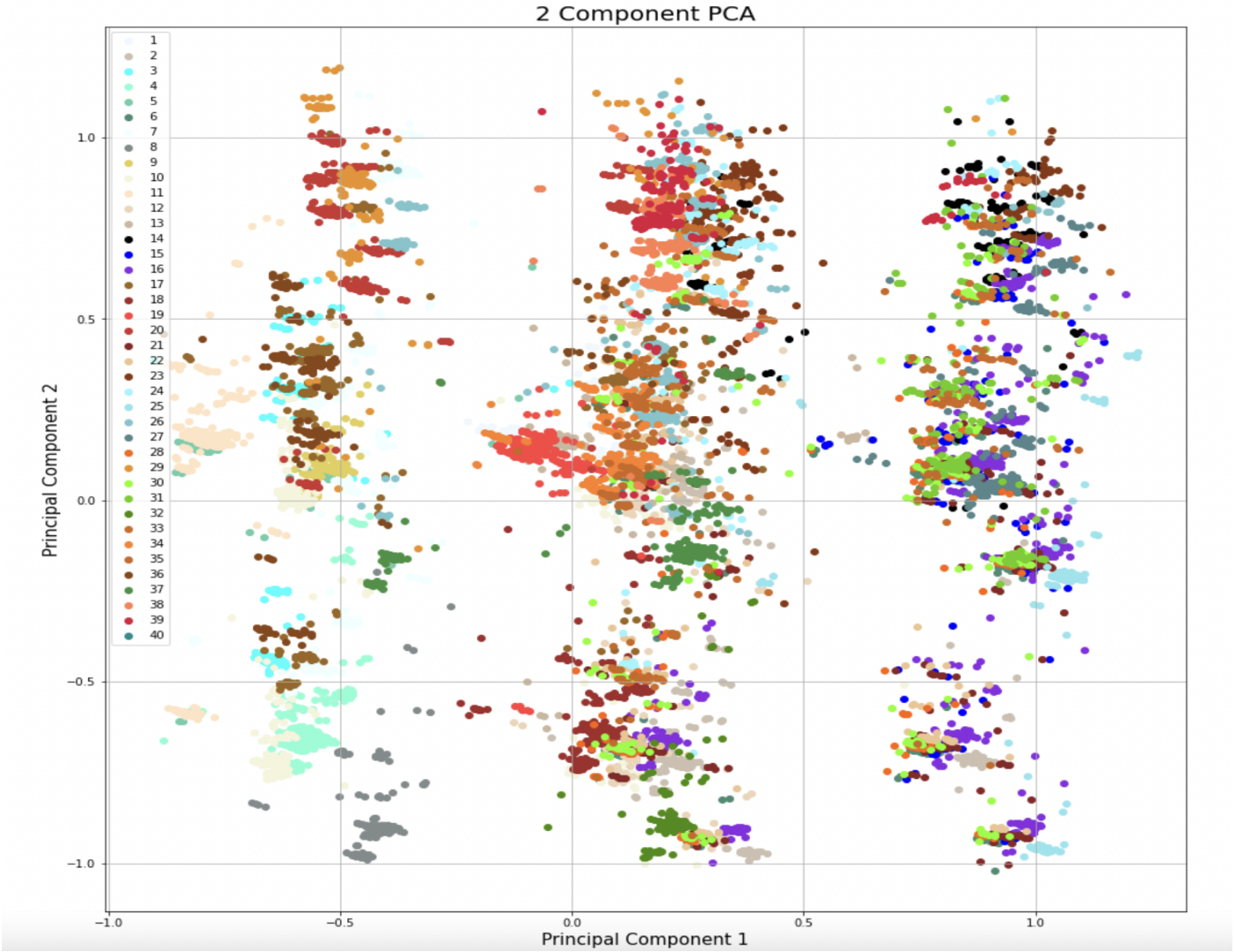
Method Evaluation

Part 1 : Recommendations using K-Means

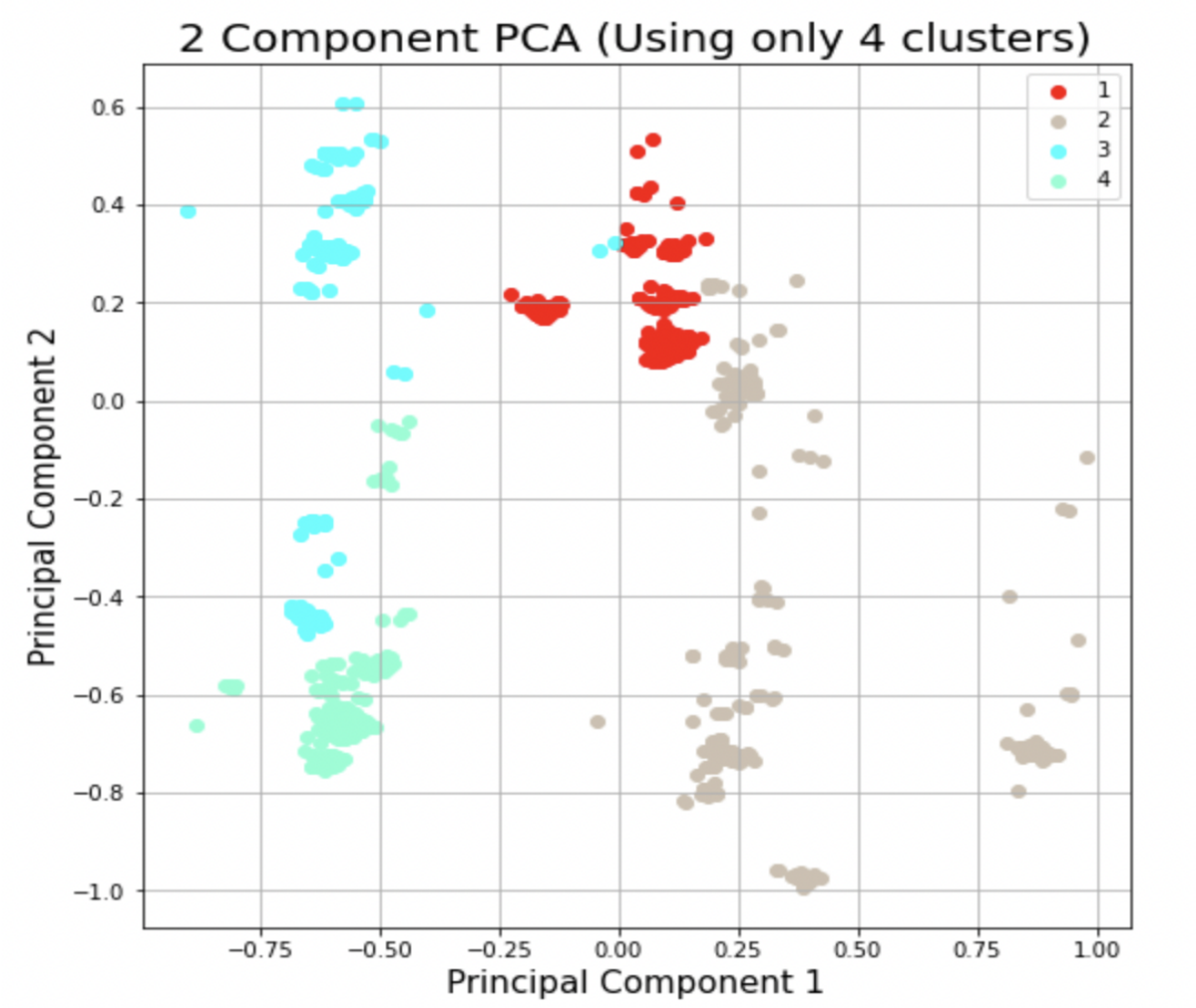
Principal Component Analysis (PCA)

Since we had too many dimensions, we had to reduce it to 2-D or 3-D to visualise our dataset. Therefore, we used Principal Component Analysis for dimensionality reduction to reduce our dimensions to two. Then we used a scatter plot to graph the reduced dimensions with the number of clusters.

The following graph shows the 40 clusters formed in a 2-D scatter plot.



The following graph shows how 4 clusters look using the 2-D scatter plot.



It is important to note that as we reduce dimensions, we are also reducing the variance so this data does not represent the entire information of the dataset, only about 25% of the data.

```
In [17]: pca.explained_variance_ratio_
Out[17]: array([0.13863003, 0.11238456])
```

Part 2: Using Collaborative Filtering

- For a given input user, we predicted similar users and created a list of movies that have a rating.
- We compared the recommended movies to the one the input user has seen and that was the metric we used and our accuracy was around 90% for the few sample users we tested.

3.1 Performance Evaluation

```
movies_suggested_and_he_watched=0
total_suggest_movies = 0
for movies in movie_match:
    total_suggest_movies=total_suggest_movies+len(movies)
    for movie in movies:
        if movie in select_user['title'].to_list():
            movies_suggested_and_he_watched=movies_suggested_and_he_watched+1
print(movies_suggested_and_he_watched)
print(total_suggest_movies)
```

executed in 9ms, finished 21:30:54 2021-12-15

```
27
30
```

Findings

When we input the movie name “Final Destination”, we get the following result:

```
0                            Wendigo
1                           Scream 3
2                  Final Destination
3                          Session 9
4                       The Mesmerist
5                Final Destination 2
6                            Willard
7             House of 1000 Corpses
8                         Cabin Fever
9     The Texas Chainsaw Massacre
10                         Dark Wolf
11                 Da Hip Hop Witch
12                        Club Dread
13                 Dawn of the Dead
Name: title, dtype: object
```

Which is accurate because when we used google to find similar movies to Final Destination, we are able to get some movies recommended by our algorithm such as :

Final Destination 2, Scream 3, The Texas Chainsaw Massacre, Cabin Fever, Dawn of the Dead. Here is the link for the google search:

Movies similar to Final Destination

Biggest Challenge

- The biggest challenge was to find a good data set, the dataset was really messy and a lot of cleaning was required.
- Budget and revenue columns were garbage otherwise could have been good factors to cluster well.
- The dataset was not labeled in the sense we did not have a way to measure train and test performance, so we had to use a different approach with clustering and performed clustering analysis with PCA and elbow methods.
- In our 2nd approach of collaborative filtering and measuring the performance by the number of movies watched by the user out of the total recommended.

Next Steps

- The next steps for our dataset to perform more rigorous cluster analysis
- Add more features to the data
- Create a robust metric to measure the performance of the recommendation
- Use more complex models and evaluate performance.
- The currently used method is User-to-user C.F., so:
    - Implementing Item-to-item C.F. on a much larger and much detailed dataset.
    - Making it hybrid, by including Bayesian classification and other methods like that.
    - Employing a structure of multiple features including genres, ratings etc.