Movie Recommendation System for a Group of Friends

Aniansh Raj Singh IIIT Delhi
aniansh19019@iiitd.ac.in

Bhavya Narang
IIIT Delhi
bhavya19462@iiitd.ac.in

Harshal Dev
IIIT Delhi
harshal19306@iiitd.ac.in

Satwik Tiwari
IIIT Delhi
satwik19100@iiitd.ac.in

Abstract

During the pandemic, "Virtual Movie Nights" saw quite a boom. However deciding on what to watch, given everyone has different preferences and tastes and liking was a tedious process. This also resulted in having a lot of time and energy being wasted. We hope to help simplify this decision-making process through our work by building a movie recommendation system that takes into account the preferences of all the participants to recommend movies that are aligned with most of their preferences.

1. Introduction

Given a certain number of movies by consensus with a group of friends, we are trying to predict a movie that these friends can watch and the predicted movie is going to try to accommodate multiple choices and interests of different individuals depending on various factors such as length of the movie, director, actors, language, country of origin, year of release etc which is done by our machine learning system.

2. Literature Review

There are mainly two kinds of recommendation systems used for content recommendation (such as movies, music and many other scenarios), namely, content-based filtering and collaborative filtering. Both the methods have their strengths and weaknesses. Content-based recommendation systems use the metadata about movies such as title, user ratings, genre, cast, director, date of release and many other metrics to present movies similar to what the user has selected or rated. On the other hand, collaborative filtering uses data about the ratings given by different users to the different movies on the platform. This method asks the user to rate a few movies; it then searches among the platform's users to find users with similar tastes and then recommends the movies liked by those users to the active user. Both types of techniques work well in different contexts, and both suffer in different contexts.

2.1. A Hybrid Approach for Movie Recommendation[2]

This paper aims to make a hybrid recommendation system using both recommendation systems and switching between the two whenever deemed appropriate. This switching happens on criteria defined by the authors where the authors use collaborative filtering by default but switch to content filtering when the performance of collaborative filtering is poor. We also plan on implementing a hybrid recommendation system of sorts utilising the metadata about the movies as well as the user interaction data.

2.2. Movie Recommendation System Based on Movie Swarm[1]

In this paper, a new concept/algorithm was explored for movie recommendation systems - movie swarm mining. It uses two pruning rules and vertical data format frequent item mining. It solves the new item recommendation problem and provides an idea about the current trends of the popular movies and user interests. This is very helpful for movie producers to plan new movies. The paper claims that the proposed method is more efficient than the collaborative filtering method and content based recommendation system. The movie recommendation system mines movie databases to collect all the important information, such as, popularity and attractiveness, required for recommendation. The paper went on talking about how content based methods of recommendation or collaborative filtering fail at some point or the other - critical point of failure being unable to recommend to new users. However a hybrid of both these approaches, though minimises the issues, yet it faces difficulties in dealing with new users.

3. Dataset

3.1. Dataset Description

Our dataset contains the following attributes of the movie:

- imdb_title_id : a unique id to uniquely identify the movie
- title: the current title of the movie (after remake/redub etc.)
- original_title: the title of the movie when it was released in its original format
- genre: the genre of the movie which can be one or more for each movie.
- country: the country in which movie was released
- language: the language in which movie was dubbed.
- director: Names of the director(s).
- actor: Names of the actors/actresses.
- avg_votes: average votes given to the movie
- reviews_from_users: number of users given by the users
- reviews_from_critics: number of users given by the critics

3.2. Exploratory Data Analysis

Here, we present some exploratory analysis done on the data.

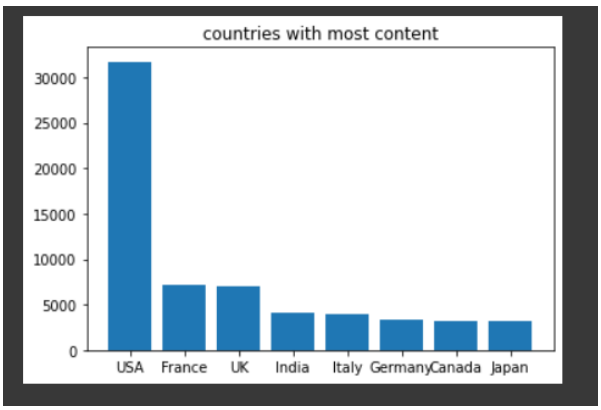

Figure 1. Countries with the most content. The content from USA is the highest among all the countries.

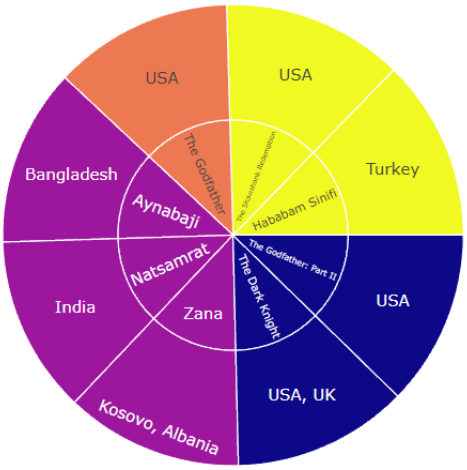

Figure 2. Most rated movies and their corresponding countries and average ratings.

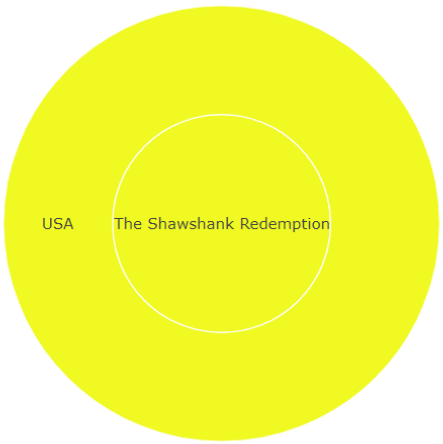

Figure 3. As we can see, the area in yellow has the highest avg rating and the movie is “Shawshank Redemption”. The graph also shows the country of the movies as well.

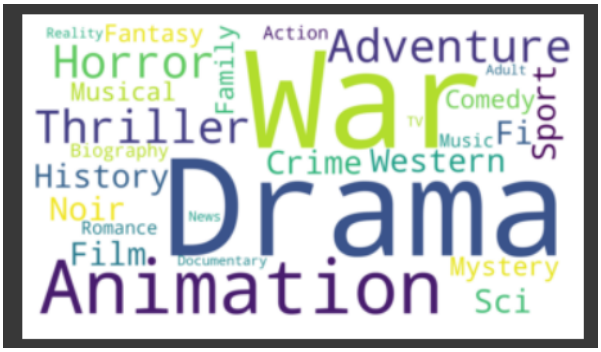

Figure 4. Word Cloud for all movie genres. The size of each genre indicates the count of the movie with this particular genre. As we can see most of the movies were in the genre categories of Drama, Animation and War.

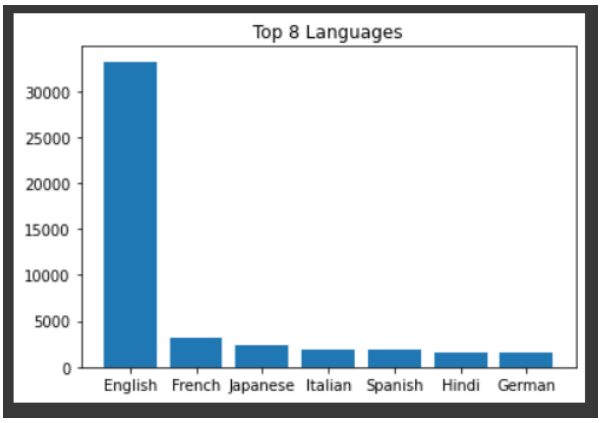

Figure 5. Top movie languages.

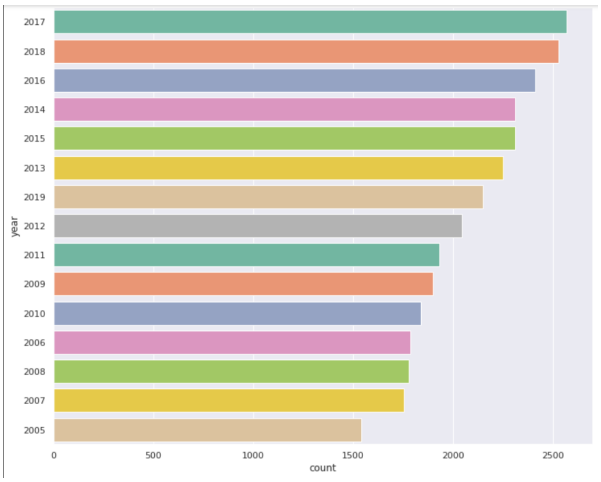

Figure 6. The number of movie release in each year.

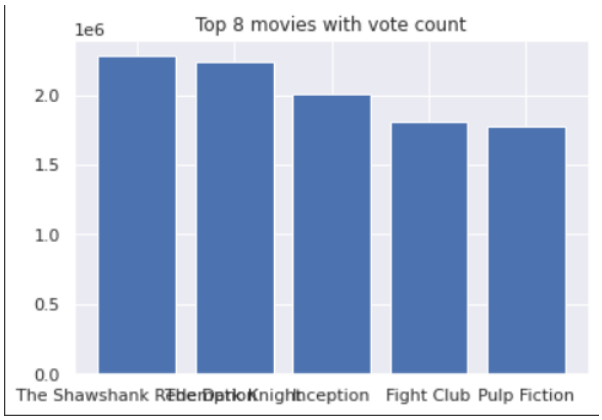

Figure 7. Movies with the most votes.

3.2.1 Ratings of the Top Movies

The following are the distributions of ratings in the top movies in the database.

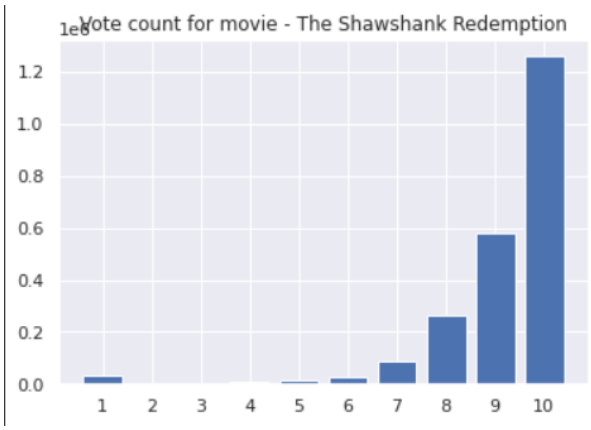

Figure 8. The Shawshank Redemption

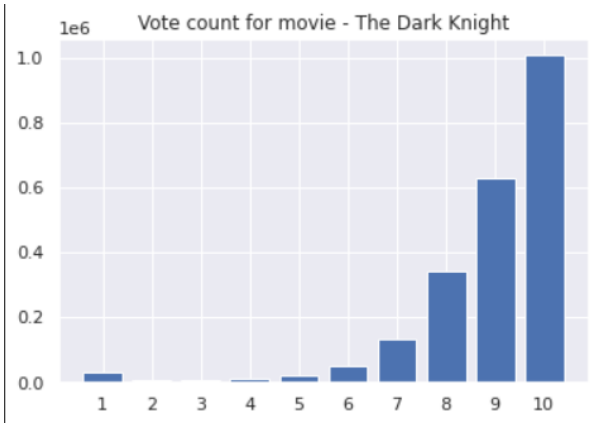

Figure 9. The Dark Knight

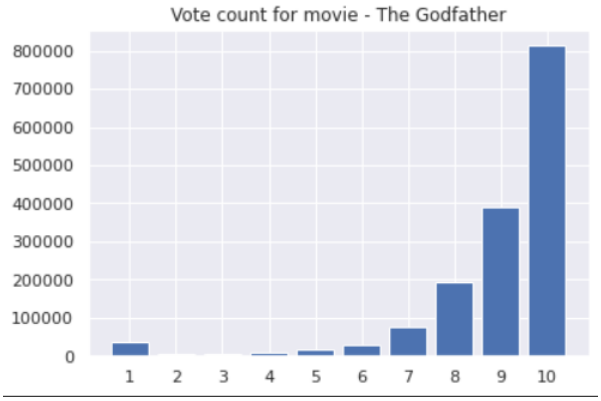

Figure 10. The Godfather

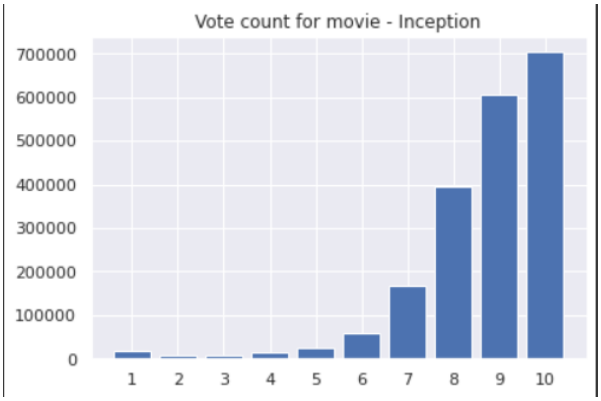

Figure 11. Inception

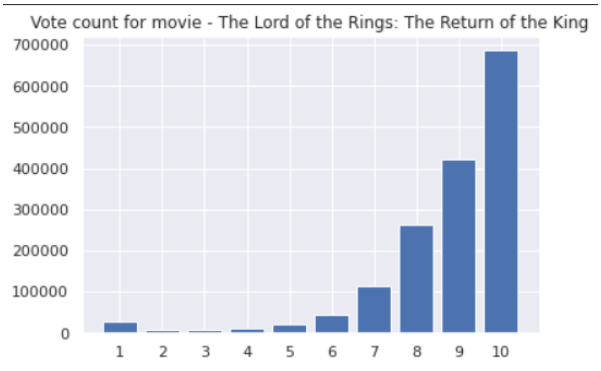

Figure 12. Lord of the Rings

3.3. Data Preprocessing

3.3.1 Dropped Columns

```
df_movies.isnull().sum()
```

```
imdb_title_id                0
title                        0
original_title               0
year                         0
date_published               0
genre                        0
duration                     0
country                     64
language                   833
director                    87
writer                    1572
production_company        4455
actors                      69
description               2115
avg_vote                     0
votes                        0
budget                   62145
usa_gross_income         70529
worlwide_gross_income    54839
metascore                72550
reviews_from_users        7597
reviews_from_critics     11797
dtype: int64
```

Figure 13. Null values analysis

The columns we dropped were :

1. Description: reason is that it had fully textual data and would lead to NLP context which is not expected of the project.
2. production_company, budget, usa_gross_income, worldwide_gross_income,
3. Metascore: columns with a high magnitude of null values are dropped as they cover a higher percentage of samples.

3.3.2 Outlier Detection

As we can see in the languages column of the movies, we can either drop them but that would cause a loss of data. What we observed is, that the given movies are very old and hence we can safely assume their language to being 'Silent' as initially, the movies did not contain any language.

The rows with missing values are then finally dropped as there is no way of handling the data.

Now we had to look out for various genres, languages, actors, directors and countries as this is textual data and what machine understands is only numbers. Hence we found out the number of distinct values in each of the following:

```
Unique values in genre 28
Unique values in country 214
Unique values in language 268
Unique values in director 25403
Unique values in actors 173433
```

Figure 14. Unique Values in our Dataset

We have used a bag of words implementation for the same as we cannot discard this data of text type as doing so would cause harm to the performance of the system.

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df_movies['genre'])
vectorizer.get_feature_names()
```

```
['action',
 'adult',
 'adventure',
 'animation',
 'biography',
 'comedy',
 'crime',
 'documentary',
 'drama',
 'family',
 'fantasy',
 'fi',
 'film',
 'history',
 'horror',
 'music',
 'musical',
 'mystery',
 'news',
 'noir',
 'reality',
 'romance',
 'sci',
 'sport',
 'thriller',
 'tv',
 'war',
 'western']
```

Figure 15. Genres in the Dataset

4. Methodology

We want to build a recommendation system that takes into account the preferences of a number of different users. We approach this problem from the viewpoint of content recommendations systems used by services like Amazon and Netflix.

There are three main approaches to content recommendation. The first is Demographic Filtering, where recommendations are based not on the user preferences but on the overall ranking of the movies based on global reviews received by the movies. For example, if we list the top 100 movies of all time on IMDB, the list will be the same for all users irrespective of their tastes or preferences. This technique is much better than randomly recommending movies and it is useful when you want to recommend content that is generally favoured by the larger population of viewers. But of course, this approach does not at all factor in user preference. The second approach is Content Based Filtering. This technique uses the metadata about the movies such as movie title, synopsis, ratings, reviews, genre, release date, run time, cast, director and many other metrics, to find other movies which are similar to the movies selected by the user. When the user selects a movie, the movie is matched against other movies in the database to find the movies that are the most similar to the selected movie based on the above mentioned parameters. The top matches are then presented to the user and this is how content based filtering works in essence. The similarity metric being used can be something like cosine similarity, Pearson similarity or any other such metric. This approach does take into account the user's selection however it may fail to capture the general taste and inclination of the user. For instance, if the user selects a documentary, the user is most probably not just interested in watching documentaries and is also perhaps interested in science fiction action movies. This approach will most likely fail to capture this because documentaries may rank higher as per this approach.

The last approach is that of collaborative filtering. This approach does not require the movie metadata to make recommendations, instead it relies on the user ratings made by other users of the service. The idea with collaborative filtering is that we first collect ratings on some movies from the active user. We then find users in our database that are similar in their taste and preferences to the active user. We then use the movies preferred by these matched users to make recommendations to the active user. Hence, we are able to make predictions without using any metadata about the movies. This approach tries to match the general taste and behaviour of the active user to existing users in the database and then tries to predict what the active user might like. This method may also make novel recommendations to the user, which may differ quite a bit content-wise from the selected movies but still be rated high by the active user. This makes collaborative filtering a powerful technique. A major downfall of this approach, however, is that data sparsity can negatively impact its performance. The data about user ratings and reviews is likely to be sparse as not many users rate too many movies and the ratings of the active user may not overlap with any of the pre existing users in the database. Or the match might be made with a user who is in fact very dissimilar in terms of movie preferences from the active user. Thus, used on its own, the technique is prone to poor performance.

The idea with our approach is to combine some aspects of all three of the above techniques to produce an effective, robust and reliable recommendation system which uses both the movie metadata and the user reviews and rating data to make recommendations thus making full use of the available data. We want the three kinds of filtering to complement and support one another to make a hybrid recommendation system.

4.1. Model Details

For our baseline model, we have implemented a content filtering model that uses our preprocessed data to make movie recommendations. The model computes the Pearson Correlation between a movie of our choice and the rest of the movies in the database. The model then ranks the movies in the database according to the Pearson Correlation values and presents the top 5 movies to the user. We have used Pearson Correlation instead of cosine similarity because Pearson correlation is indifferent to the absolute values of the metrics and it only considers the relative variations unlike cosine similarity. This is because of the way Pearson Correlation is formulated.

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}}$$

For our model we have converted the textual data into one hot encoding wherever possible and for numerical data we have used bucketization techniques to bucket the data into 5 buckets. Now as the one hot representation data is of 4000+ columns we cannot directly train as the training time would be too high. We have used PCA for dimensionality reduction.

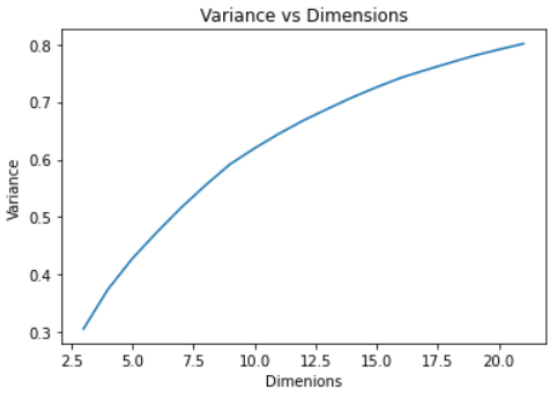

Figure 16. Variance vs Dimensions.

We have used the first 20 principal components as they are preserving 80% variance.

Then for the obtained data we have applied K means clustering and using the elbow method we have used the value of the number of clusters as 9.

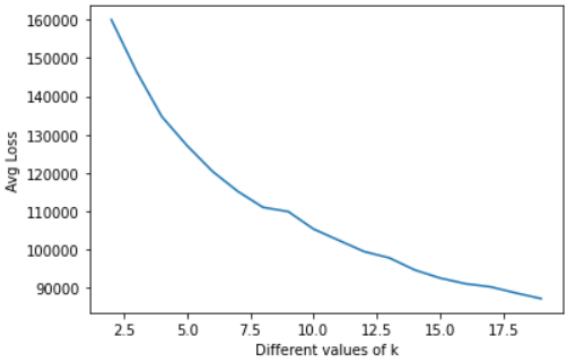

Figure 17. Elbow Curve : K vs Loss

We can observe that we have got an efficient representation of the data which was 4000 columns after one-hot representation.

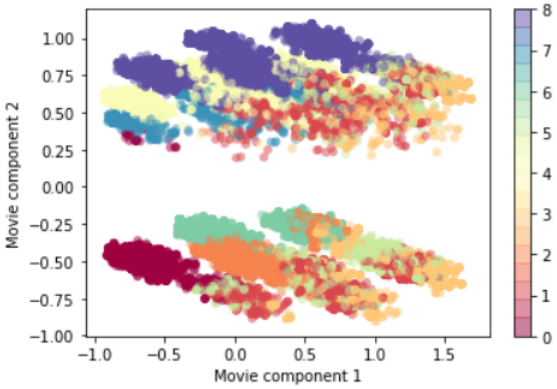

Figure 18. Visualization of Clustered Data

We have designed our custom similarity metric, that is, now for the obtained clusters we have grouped the points in them. Now the user/s can input any 2 movies such that they belong to our dataset and now we have divided the problem into subproblems which are as follows:

1. If the given movies belong to the same cluster then according to our defined similarity score we find the top 'n' movies from our dataset.

2. If they do not belong to the same cluster then we use the cluster centres to find a similarity score between our given clusters and then we use the highly rated cluster to predict top 'n' movies with our given 2 movies with our similarity score.

4.2. Similarity measure used

Given two movies we have used 3 types of distances of 3 types which are Manhattan, euclidean, Mankowski. Now we calculate the distance between these 2 movies in PCA projected space and then for every movie to compare with we have computed these three types of distances separately and then taken the product of a sufficiently lar ge enough number subtracted by the distances as the distance is inversely related to the closeness of the movies in the projected space.

5. Results

movie1	Avvocato di me stesso
movie2	Mente omicida
recomendation - 1	L'anniversario
similarity metric value 1	5.07873
recomendation - 2	Un attico sopra l'inferno
similarity metric value 2	5.07873
recomendation - 3	Cul de sac
similarity metric value 3	5.07873
recomendation - 4	Rancho bravo
similarity metric value 4	4.83005
recomendation - 5	Fireball 500
similarity metric value 5	4.83005

Figure 19.

movie1	Des plumes dans la tête
movie2	FrightWorld
recomendation - 1	Dragons of Camelot
similarity metric value 1	3.98751
recomendation - 2	Apocalypse and the Beauty Queen
similarity metric value 2	3.98751
recomendation - 3	Iconoclast
similarity metric value 3	3.79069
recomendation - 4	Beverly Hills Christmas
similarity metric value 4	3.70924
recomendation - 5	An Easter Bunny Puppy
similarity metric value 5	3.70924

Figure 20.

movie1	Operazione Crossbow
movie2	Kape neuwareu
recomendation - 1	Peppermint Frappé
similarity metric value 1	4.23758
recomendation - 2	La caccia
similarity metric value 2	4.23758
recomendation - 3	Rosa blanca
similarity metric value 3	4.23758
recomendation - 4	I cinque del bunker
similarity metric value 4	4.15725
recomendation - 5	La brigata del diavolo
similarity metric value 5	4.11789

Figure 21.

movie1	Kolonya Cumhuriyeti
movie2	Obce niebo
recomendation - 1	Hikkoshi daimyô!
similarity metric value 1	1.28691
recomendation - 2	Neko zamurai
similarity metric value 2	1.28691
recomendation - 3	Kiyosu kaigi
similarity metric value 3	1.28691
recomendation - 4	Zohi Sdom
similarity metric value 4	1.18532
recomendation - 5	N (Io e Napoleone)
similarity metric value 5	1.43099

Figure 22.

movie1	La montaña rusa
movie2	The Other Side
recomendation - 1	The Second Coming of Christ
similarity metric value 1	5.32505
recomendation - 2	Lay It Down
similarity metric value 2	5.32505
recomendation - 3	Karate Kid 4
similarity metric value 3	5.32505
recomendation - 4	Muse
similarity metric value 4	5.22646
recomendation - 5	E-Demon
similarity metric value 5	5.22646

Figure 23.

These are the predictions of our model with the given inputs by the users and these contains our similarity score and along with them the movie values.

6. Conclusion

After conducting our tests and training our content based filtering model using a number of different similarity metrics, we get varying results in terms of the predictions offered by our recommendation system. We ask the user to select some movies based in their preferences and our recommendation system then presents the user with movies whose content is deemed by the algorithm to be similar in nature to the user's selections.

The results from our content-based predictions look promising and we can try to incorporate collaborative filtering methods as well for future work if we have a dataset of user ratings of multiple users on multiple movie titles. The inclusion of collaborative based filtering can improve the effectiveness of our current model.

Learning from the project:

- Teamwork.
- How to handle large data.
- EDA techniques.
- How to make use of visualizations.
- How to infer them to detect outliers.
- How to handle null/nan values/missing values in the data.
- How to represent textual data in the form of vectors.
- Which models to choose and how to choose an evaluation metric for unsupervised learning type of models.
- How to use similarity metrics in unsupervised learning techniques.
- Importance of dimensionality reduction and principal components variance analysis.
- Unsupervised Learning.
- Clustering algorithms.

References

[1] Sajal Halder, A.M. Jehad Sarkar, and Young-Koo Lee. Movie recommendation system based on movie swarm. In *2012 Second International Conference on Cloud and Green Computing*, pages 804–809, 2012. 1

[2] George Lekakos and Petros Caravelas. A hybrid approach for movie recommendation. *Multimedia Tools and Applications*, 36(1):55–70, Jan 2008. 1