# Movie Recommender System

Mohammad Rouintan , Reza Mousavi

Computer Science, Shahid Beheshti University

**Abstract**

*Movie recommendation systems are an integral part of the entertainment industry, aiming to provide personalized movie suggestions to the users based on their preferences. The project entails analyzing a large dataset of movies, incorporating user preferences, and implementing algorithms to generate relevant movie recommendations. In this project we use collaborative filtering, content-based filtering, and hybrid methods. Collaborative filtering involves recommending items to users based on their similarity to other users' preferences. Content-based filtering, on the other hand, recommends items based on their similarity to the user's previously liked items. Hybrid methods combine both collaborative and content-based filtering approaches to provide more accurate and diverse recommendations.*

## Introduction

The ***Movies Dataset*** is a collection of data on 45,000 movies and contains information such as budget, revenue, release dates, languages, production countries, and companies for each movie. The dataset also includes ratings of 270,000 users for these movies. The data was obtained from The Movie Database (TMDb) API.

Our goal in this project is to implement a recommender system on the movie dataset, which is done by various methods such as collaborative filtering, content-based filtering, and hybrid methods.

For example, to implement the ***content-based filtering*** method, we can make a decision based on the features available for each movie in the dataset and use the movies that are close to each other in these features as recommendations. In the ***collaborative flltering*** method, based on the previous information of each user, we make a decision for the user, for example, from the scores he gave to the movies, we can understand what his favorite movies are. In the ***hybrid*** method, we use the combination of the two previous methods, that is, we use both the characteristics and descriptions of the movies and the user's previous information.

Because there are several different csv files in this dataset, it is very important which files we use. In this way, to implement different methods, we need to use different files or sometimes merge these files and then use them. It is also important what preprocessing we do on the data and which features we use for each method.

Finally, after implementing these movie recommender systems, we have deployed these models in ***Hugging Face Spaces***.

## Methodologies

### *Exploratory Data Analysis*

first, for data analysis we should read description of dataset and know features of dataset in detail. This dataset contains seven csv files:

- ***movies_metadata.csv***: The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
- ***keywords.csv***: Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.
- ***links.csv***: The file that contains the TMDB and IMDB IDs of all the movies featured in the Full MovieLens dataset.
- ***links_small.csv***: Contains the TMDB and IMDB IDs of a small subset of 9,000 movies of the Full Dataset.
- ***ratings.csv***: 26,000,000 ratings from 270,000 users on 45,000 movies.
- ***ratings_small.csv***: The subset of 100,000 ratings from 700 users on 9,000 movies.

In general, datasets contain numeric, categorical, JSON, etc. features. Before implementing EDA on these datasets, we have tried to extract information from these JSONs and put them in features with new names in the same dataset. Therefore, the analyzes performed on new datasets.

**Movies-Metadata:** This dataset has a large number of missing values, which are in some special features, which we will decide to fill or delete altogether.

According to the analysis, we see that the adult feature of all the movies (almost 99.98%) is False . Therefore, this feature does not give us much useful

information about the dataset (Fig. 1).

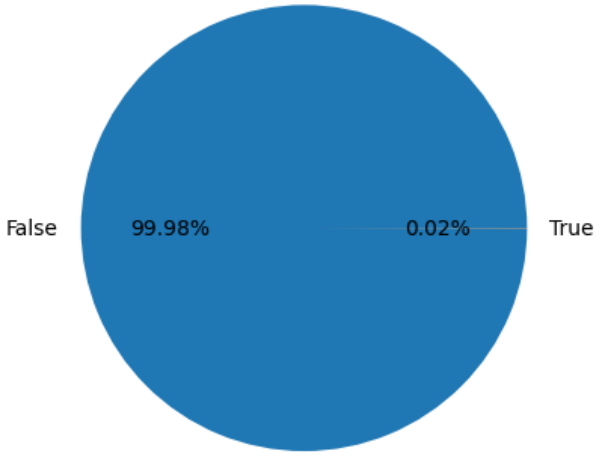


**Figure 1:** *adult feature of all the movies (almost 99.98%) is False.*

According to the describe and also the histogram, we see that the amount of budget for most of the films is zero.

According to the analysis, we see that the homepage feature of almost 83% of movies is NaN . Therefore, this feature does not give us much useful information about the dataset.

According to the analysis, we see that the original_language feature of almost 71 of movies is en (Fig. 2).

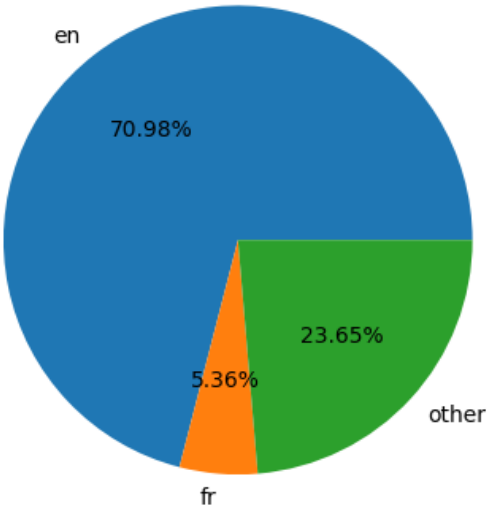


**Figure 2:** *original_language feature of almost 71 of movies is en.*

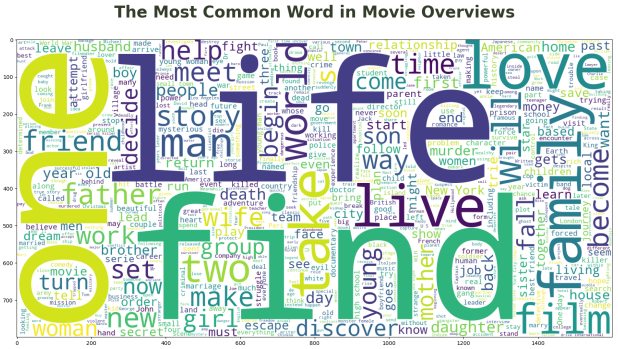


**Figure 3:** *According to the word cloud plot, we see the most common word in Movie Overviews.*

Started from 1930, movies industry had grown significantly from 50 years ago. A drop in total released movies around 2020 is because the dataset only contains a few data in those years (Fig .4).

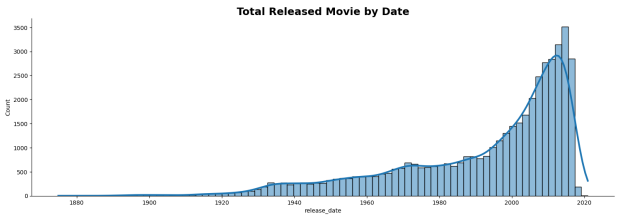


**Figure 4:** *Date of Released Movie*

Budget and Revenue just slighly influence the popularity of the movies. Most of the movies lay on top of the yellow line, indicate that those movies make a profit (Fig. 5).

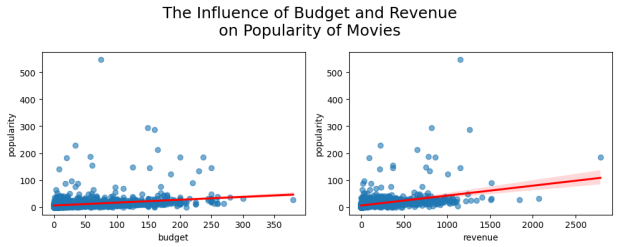


**Figure 5:** *Regplot of popularity based on budget & revenue*

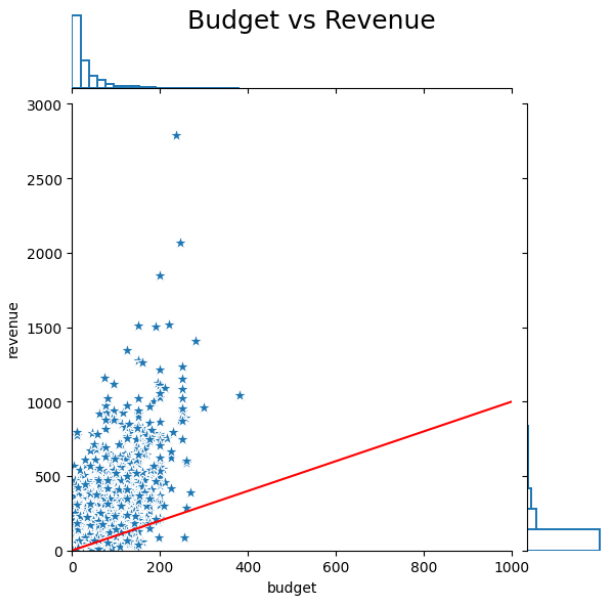


**Figure 6:** *Relationship between budget and revenue*

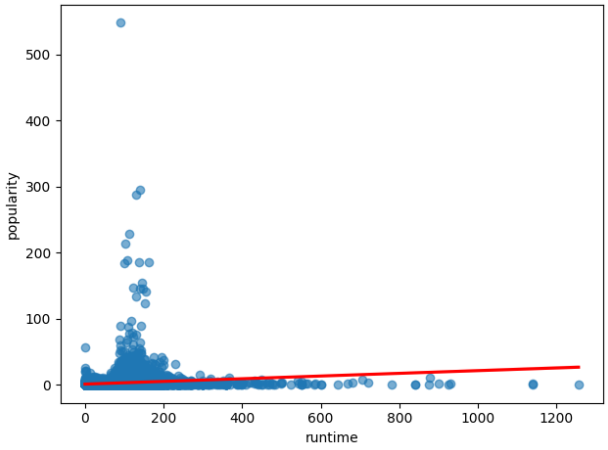


**Figure 7:** *Regplot of popularity based on runtime*

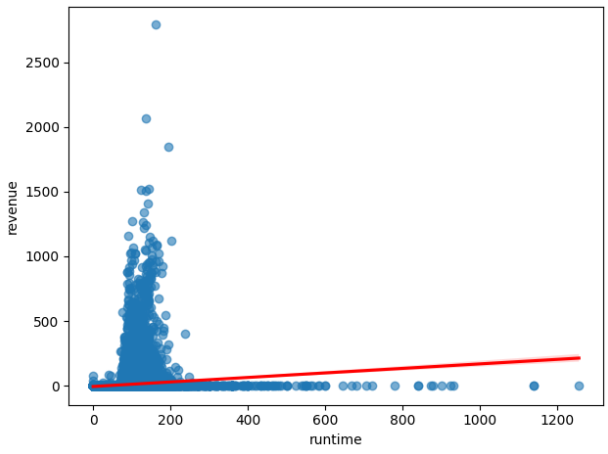


**Figure 8:** *Regplot of revenue based on runtime*

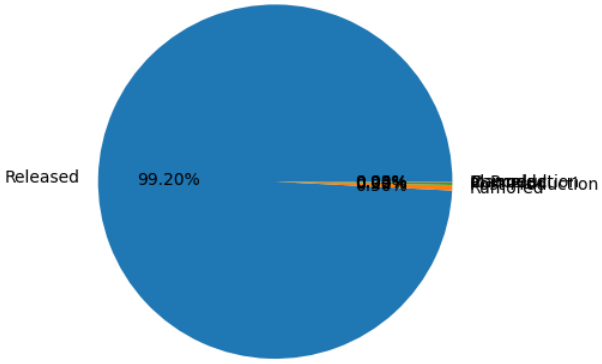


**Figure 9:** *Percentage of Released Movies*

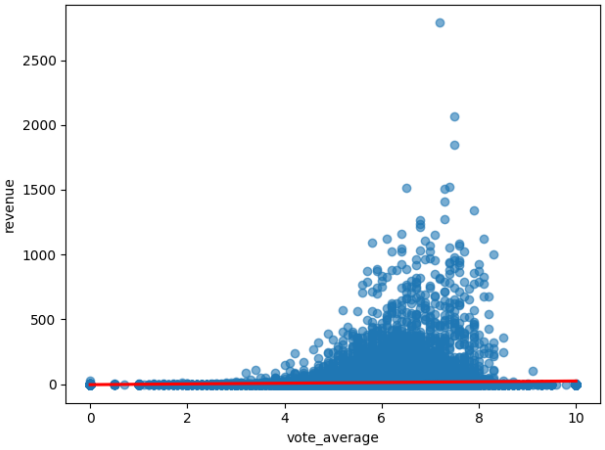


**Figure 10:** *Regplot of revenue based on vote_average*

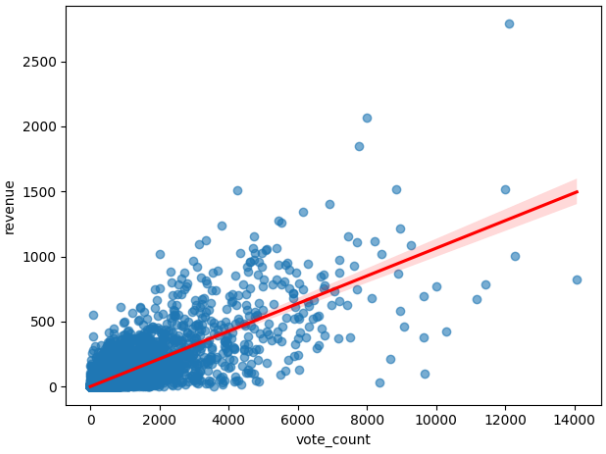


**Figure 11:** *Regplot of revenue based on vote_count*

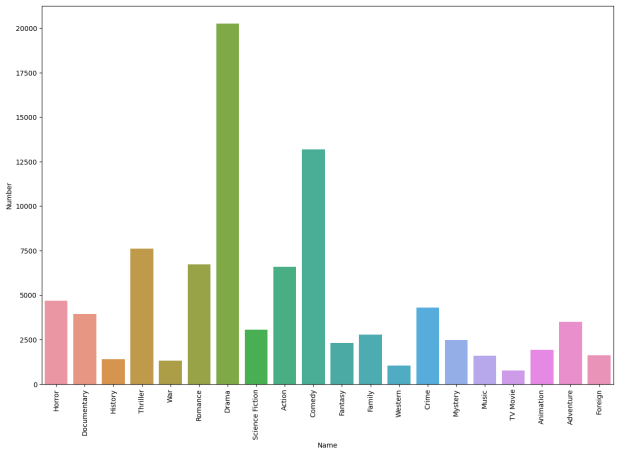


**Figure 12:** *Number of movies in each genres*

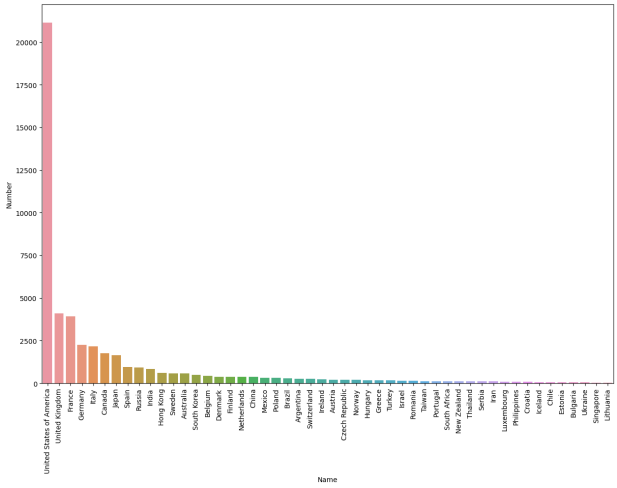


**Figure 13:** *Number of movies in each country*

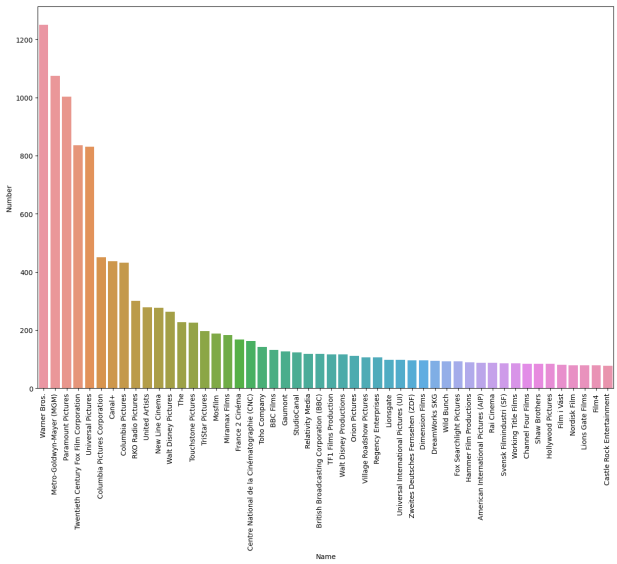


**Figure 14:** *Number of movies in each company*

**Credits:** In this section, we examine the Credits data set. At first glance, this data set contains information of the film crew and the names of the film actors and in general the characters of the film, which are not stored correctly. First, we will examine the crew section:

**Crew:** The review of the crew section is very important because it can give us important information about the actors of the film so that we can make better offers to the users. In general, we have 22 different departments in the crew department.

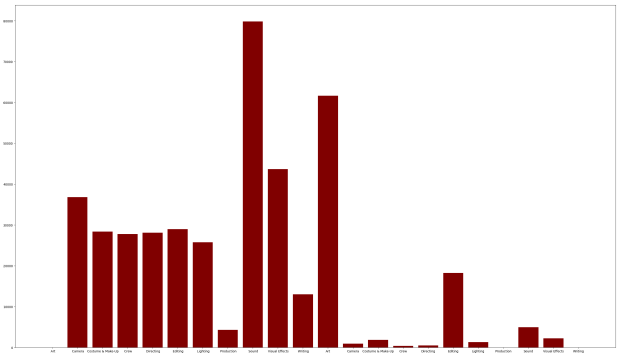

**Figure 15:** *22 different departments of the crew*

It can be seen that the voice recorder department has the highest value among all departments. In general, most departments have the same amount of crew on average among films, such as directors. Next, we will examine different crews based on their names. In the chart below, you can see the names of the 50 most frequent crews.

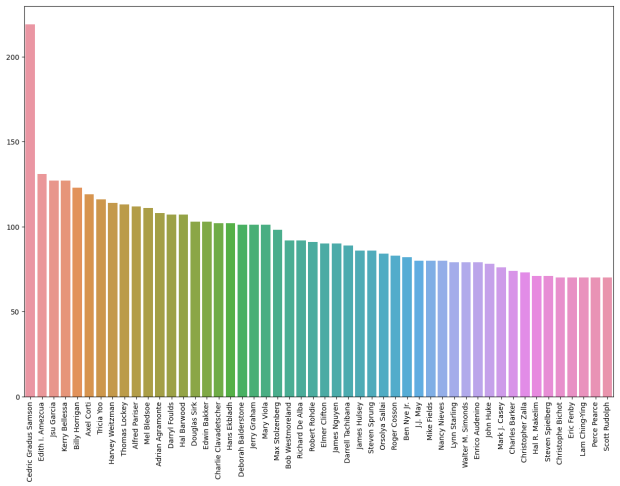

**Figure 16:** *50 repeated crew names*

It can be seen that in this part, you can see the most popular crews among different producers because these people have been most active and their names have been repeated the most among all the names of the crews. In the following, we will examine different

jobs in the crew.

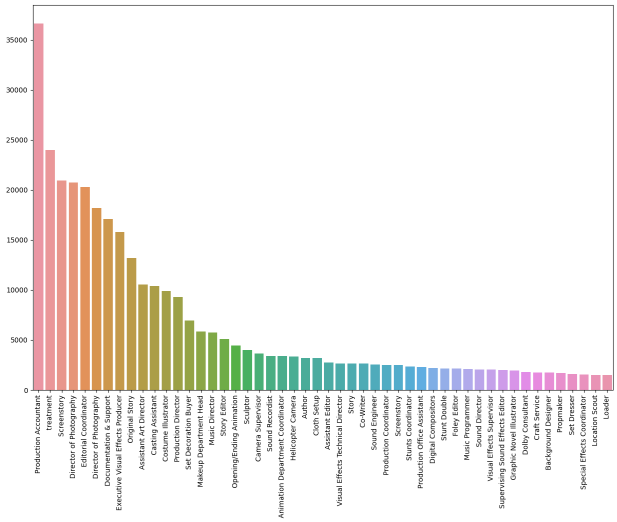


**Figure 17:** *50 repetitive jobs*

It can be seen that these jobs were the most needed in all the movies.

**Cast** In this section, we will first examine 50 frequently used actor names.

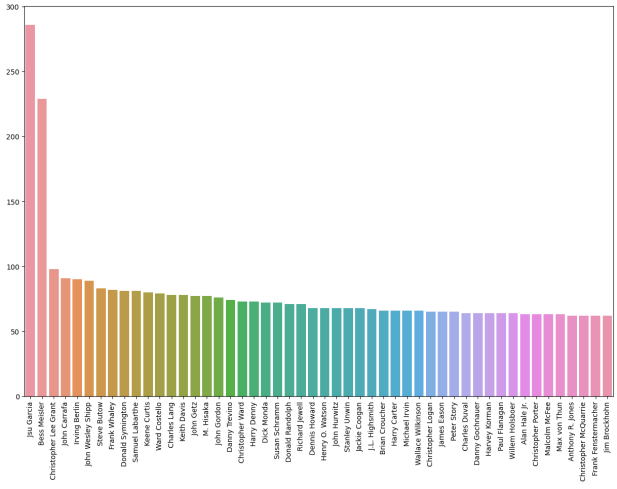


**Figure 18:** *50 most repeated actor names*

The cast has the most repeated names in all the movies. It can be seen that these cats are very popular with the director and the audience and play a very important role in offering the audience. In the following, we will examine 50 popular and frequent characters in movies.

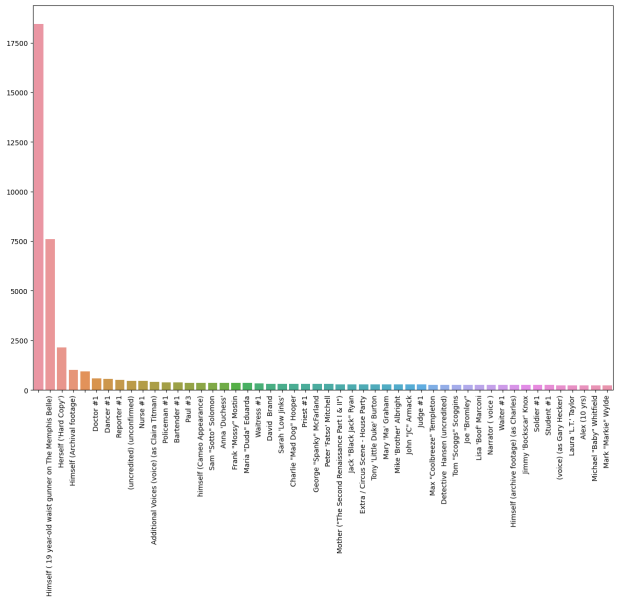


**Figure 19:** *50 most repeated characters in movies*

These characters have been very popular with the audience because they have been repeated the most in different movie titles.

**Keywords:** According to the analysis, we found that the number of unique keywords is equal to 19961. The most used word in these words is the word "woman director", which is used 3039 times. The barplot show top 50 of most used keywords (Fig. 15).

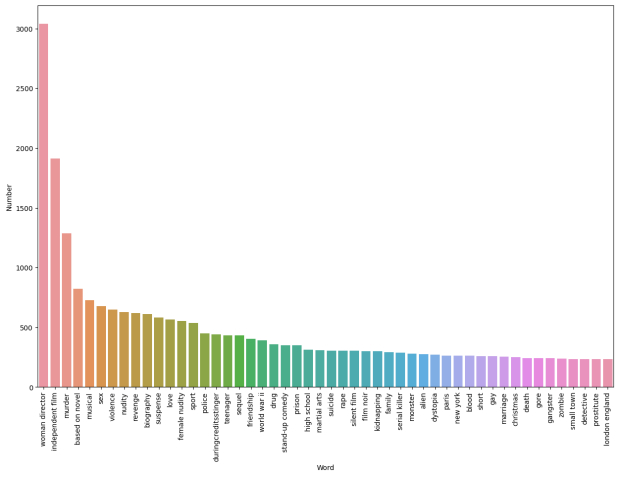


**Figure 20:** *Top 50 of most used keywords*

## *Models*

**Content-Based Filtering:** Content-based filtering is a recommender system technique that recommends items to users based on the similarity of the items' attributes to the attributes of items the user has liked in the past. In the context of movies, this means that a content-based filtering system will recommend

movies to a user based on the similarity of each movie's metadata (such as genre, release year, director, cast, keywords, etc.) to the metadata of movies that the user has rated highly in the past.

To build a content-based filtering engine for movie recommendations, we first need to compute similarity scores between pairs of movies based on their metadata. We can use various metrics to compute this similarity, such as cosine similarity, Euclidean distance, Jaccard similarity, etc. Once we have computed similarity scores for all pairs of movies, we can use these scores to recommend movies that are most similar to a particular movie that a user liked.

Content-based filtering is useful when there is not enough user data available to build a collaborative filtering system, or when user preferences are highly specific to certain attributes of the items (such as genre or director). However, it may not be effective in recommending items outside of the user's known preferences or in situations where the metadata is not rich enough to differentiate between items.

On the other hand, TMDB rating is a rating system used by the website "The Movie Database" (TMDb) to rate movies and TV shows. TMDb is a popular online database that provides information about movies and TV shows, including details such as cast, crew, plot summaries, and user ratings.

The TMDB rating is calculated based on the average rating given by users who have rated the movie or TV show on the TMDb website. The rating scale ranges from 0 to 10, with higher scores indicating better ratings. The rating is calculated using a formula that takes into account the number of ratings received by the movie or TV show, in order to give more weight to movies or TV shows with a larger number of ratings.

It's important to note that the TMDB rating is a user-generated rating and may not necessarily reflect the critical consensus or commercial success of a movie or TV show. Additionally, ratings can be influenced by factors such as the popularity of a movie or TV show, the demographics of the user base, and the timing of when the ratings were submitted. Therefore, it is advisable to use TMDB ratings as one of several factors to consider when evaluating a movie or TV show.

The formula for calculating the TMDB rating is as follows:

$$WeightedRating = (\frac{v}{v+m}) \cdot R + (\frac{m}{v+m}) \cdot C \quad (1)$$

- R = the average rating of the movie or TV show (on a scale of 0 to 10)
- v = the number of votes or ratings received by the movie or TV show
- m = the minimum number of votes or ratings required for the movie or TV show to be considered in the calculation
- C = the mean rating across the entire dataset of movies or TV shows

TF-IDF (Term Frequency-Inverse Document Frequency) is a commonly used technique in information retrieval and text mining for determining the importance of a term (word or phrase) in a document or a collection of documents. It is often used as a way to represent documents as feature vectors for machine learning models.

The TF-IDF value for a term in a document is calculated as the product of two values:

1. Term Frequency (TF): This represents the number of times a term appears in a document. It is calculated as the raw frequency of the term in the document, normalized by the total number of terms in the document. The idea behind this is that more frequent terms are generally more important in describing the content of a document.
2. Inverse Document Frequency (IDF): This represents the rarity of a term in a collection of documents. It is calculated as the logarithm of the total number of documents in the collection divided by the number of documents that contain the term. The idea behind this is that terms that appear in fewer documents are generally more important in distinguishing the content of a document from other documents in the collection.

The TF-IDF value for a term in a document is then simply the product of the TF and IDF values.

The TF-IDF vectorizer is a tool that calculates the TF-IDF values for all terms in a collection of documents and converts each document into a vector of TF-IDF values. This vector representation can then be used as input to machine learning models for tasks such as text classification, clustering, and information retrieval. The vectorizer is typically used in conjunction with pre-processing steps such as tokenization, stop-word removal, and stemming to produce meaningful features for the machine learning model. The TfidfVectorizer is a popular implementation of this technique in Python's scikit-learn library.

In this section, we have calculated the weighted rating for all movies using this formula and placed it as a column of the dataset. After that, we have calculated a feature called score for each movie in such a way that we have added a coefficient of weighted rating with a coefficient of popularity. It is up to us to determine these coefficients and for this we have set the weighted rating coefficient to 0.4 and the popularity coefficient to 0.6.

Then, by extracting the names of the casts, directors, genres and keywords of each movie, we have created a new feature that contains the combination of all these, and by applying the TF-IDF algorithm on that feature, we can turn it into a vector. Then we find the similarity between them using cosine similarity.

But the final algorithm is that for each movie, we first find similar movies with the same cosine similarity matrix, then we define a feature called final score, in which we add a coefficient of the score with a coefficient of similarities, then with sort By doing these numbers, we find the recommended movies.

**Collaborative Filtering:** Collaborative filtering is a popular method used in recommendation systems to provide personalized recommendations to users based on their past interactions and similarities with other users. This method relies on the idea that users who have similar preferences in the past are likely to have similar preferences in the future.

Our method shows the contribution using Singular Value Decomposition (SVD) algorithm. The first step involves loading the user's item rating data into a dataset object that contains information about users, items, and their respective ratings.

Next, the SVD algorithm is instantiated as an object. The algorithm is then evaluated using cross-validation, which helps assess its performance by splitting the dataset into multiple folds and calculating evaluation metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). This step helps in determining how well the algorithm performs in predicting user-item ratings.

The dataset is again loaded into a new dataset object, and a K-Fold cross-validation is applied. This splits the data into multiple folds, allowing for a more robust evaluation of the algorithm's performance.

Finally, the SVD model is fitted to the trainset, which optimizes its parameters based on the user-item ratings. The resulting model can then be used to make predictions and provide personalized recommendations to users.

Overall, this code demonstrates the application of collaborative filtering using the SVD algorithm to create a recommendation system based on user-item ratings. The process involves data loading, cross-validation for evaluation, model training, and prediction generation.

**Hybrid:** The hybrid method is a combination of content-based filtering and collaborative filtering. This approach leverages the strengths of both models to generate more accurate recommendations. By combining the content-based and collaborative filtering methods, the hybrid approach is able to take into account both the characteristics of the movies and the preferences of the users. The content-based filtering method utilizes the metadata of movies to generate recommendations, while the collaborative filtering method predicts user ratings based on similar user preferences. The hybrid method combines these two sources of information to generate recommendations that are tailored to the individual user's preferences and interests.

Our algorithm works in the hybrid section in such a way that we first find movies similar to each of the 3 movies and find the highest values of predicted ratings using the SVD model. We display movies corresponding to those values as recommended movies.

### *Deployment on Hugging Face Space*

To deploy on Hugging Face, we have created a simple app using the Streamlit package, which performs recommending in three ways. These three types are: recommend based on movie, recommend based on user and recommend based on user and movie.

The first method is based on content-based filtering, the second method is based on collaborative filtering and the third method is based on hybrid algorithm.

Except for the main file of the app, we have implemented a class called ***Model***, in which the required datasets of each model are read, and then based on the type of model, the same algorithms as we said in the previous section are implemented, and by calling them, the movie recommendation are displayed.

Also, to improve the speed of the program, we have designed this program as multi-threaded so that it can achieve results in less time. For example, for each of the three movies, we have assigned a thread to make the work faster.

As we said, this program works based on three different methods, so it has one user ID input and three movie name inputs, and it has three buttons based on which three methods are recommended. For content-based filtering method, we only need to fill three movie inputs. For the collaborative filtering method, you only need to enter the user ID, and for the hybrid method, you need to fill all the fields.

### Link of App

`https://huggingface.co/spaces/MohammadRouintan/Movie-Recommender-System`

## Results

Our experiments show that the hybrid approach outperforms both the content-based and collaborative

filtering methods individually. The content-based filtering method utilizes the metadata of movies to generate recommendations, while the collaborative filtering method predicts user ratings based on similar user preferences. However, the hybrid approach can leverage the strengths of both models to generate more accurate recommendations. By combining the content-based and collaborative filtering methods, the hybrid approach is able to take into account both the characteristics of the movies and the preferences of the users. As a result, it achieves better performance in terms of accuracy and coverage.

We also deployed the movie recommender system on Hugging Face Space, where users can test each of the three models and compare their performance. This platform allows users to input their movie preferences and receive personalized recommendations based on the three different methods. The results show that the hybrid approach is able to provide more accurate and diverse recommendations than the individual methods.

In future work, we plan to explore other methods for movie recommendation, such as deep learning-based approaches, and further improve the scalability and performance of our system. Overall, we believe that our movie recommender system represents a valuable contribution to the field of recommendation systems and has the potential to enhance the user experience for movie lovers.

## Extra Points

- Implementation of Exploratory Data Analysis (EDA)
- Using TMDB API to load movies posters (please turn on your VPN when using the app to load posters)
- Using all three methods to recommend (Content-Based Filtering, Collaborative Filtering, Hybrid)
- Using MultiThread programming for improve speed of app
- Implementation of a simple neural network (but not used in the application) `https://github.com/MohammadRouintan/Movie-Recommender-System/blob/master/Recommender%20System/NCF.ipynb`

## References

[1] Kaggle

[2] TMDB

[3] Deployment on HuggingFace

[4] Hugging Face

[5] Streamlit