INNOMATICS
RESEARCH LABS

MACHINE LEARNING PROJECT ON

MOVIE RECOMMENDATION SYSTEM

PROJECT BY **KANAV’s WARRIORS**	AMRESH MISHRA
	DANDA UPENDAR REDDY
	KARTHIKEYAN RADHAKRISHNAN
	PRUDHVI NATH
	SRI PADMAJA

PROJECT GUIDES

KANAV BANSAL (CHIEF DATA SCIENTIST)

RAMYA BHARGAVI (MENTOR)

VIJIT SINGH (MENTOR)

ABSTRACT

On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information to alleviate the problem of information overload, which has created a potential problem to many Internet users.

Recommender systems solve this problem by searching through large volume of dynamically generated information to provide users with personalized content and services. This report shows implementation of two types of recommendation system, i.e., content based, and collaborative filtering based in order to serve as a compass for practice in the field of recommendation systems.

Contents

1. INTRODUCTION:

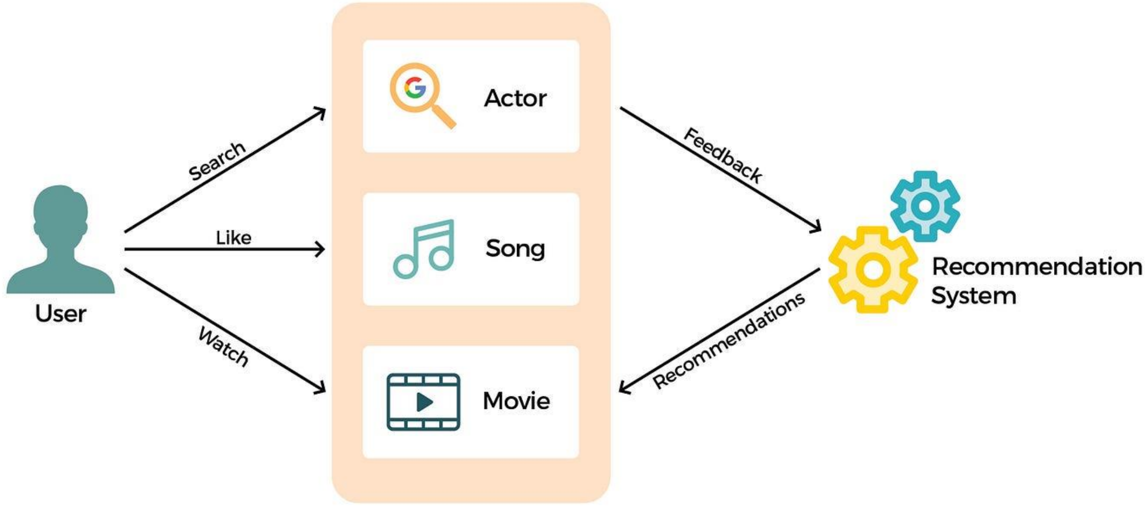
Recommender Systems are an important class of machine learning algorithms which captures the pattern of people’s behaviour and use it to predict what else they might like. For e.g. – watching a movie and liking it most probably gives a chance that a movie of the same genre can be watched in the future.

There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google.

2. PURPOSE OF THE PROJECT:

The goal of this project is to make a movie recommendation engine by making use of user based collaborative filtering technique and combining content-based results along with it. The system makes use of numerical ratings of similar items between the active user and other users of the system to assess the similarity between users’ profiles to predict recommendations of unseen items to active user. The results show that the system rests in its assumption that active users will always react constructively to items rated highly by similar users, shortage of ratings of some items, adapt quickly to change of user's interest, and identification of potential features of an item which could be of interest to the user.

This project will focus on making use of content-based approach in addition to Collaborative Filtering approach to recommend quality content to its user.



3. LITERATURE SURVEY

MOVREC is a movie recommendation system presented by D.K. Yadav et al. based on collaborative filtering approach. Collaborative filtering makes use of information provided by user. That information is analysed, and a movie is recommended to the users which are arranged with the movie with highest rating first.

Luis M Capos et al has analyzed two traditional recommender systems i.e., content based filtering and collaborative filtering. As both have their own drawbacks, he proposed a new system which is a combination of Bayesian network and collaborative filtering. A hybrid system has been presented by Harpreet Kaur et al. The system uses a mix of content as well as collaborative filtering algorithm. The context of the movies is also considered while recommending. The user - user relationship as well as user - item relationship plays a role in the recommendation.

The user specific information or item specific information is clubbed to form a cluster by Utkarsh Gupta et al. using chameleon. This is an efficient technique based on Hierarchical clustering for recommender system. To predict the rating of an item voting system is used. The proposed system has lower error and has better clustering of similar items.

Urszula Kużelewska et al. proposed clustering to deal with recommender systems. Two methods of computing cluster representatives were presented and evaluated. Centroid-based solution and memory-based collaborative filtering methods were used as a basis for comparing effectiveness of the proposed two methods. The result was a significant increase in the accuracy of the generated recommendations when compared to just centroid-based method.

Costin-Gabriel Chiru et al. proposed Movie Recommender, a system which uses the information known about the user to provide movie recommendations. This system attempts to solve the problem of unique recommendations which results from ignoring the data specific to the user. The psychological profile of the user, their watching history and the data involving movie scores from other websites is collected. They are based on aggregate similarity calculation. The system is a hybrid model which uses both content based filtering and collaborative filtering.

To predict the difficulty level of each case for each trainee Hongli LIn et al. proposed a method called content boosted collaborative filtering (CBCF). The algorithm is divided into two stages, first being the content-based filtering that improves the existing trainee case ratings data and the second being collaborative filtering that provides the final predictions. The CBCF algorithm involves the advantages of both CBF and CF, while at the same time, overcoming both their disadvantages.

4. CASE STUDY

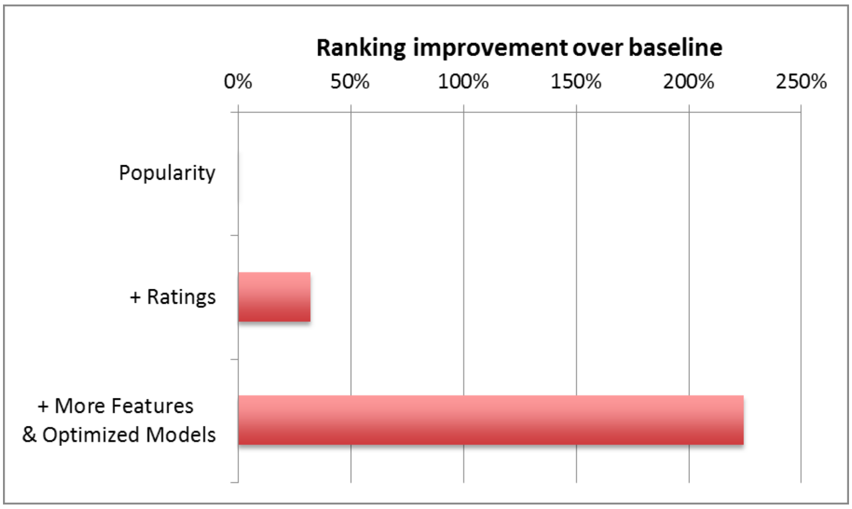
The first algorithm behind this recommendation was called CineMatch. The CineMatch algorithm had a long run and proved successful at predicting movies that subscribers would like. According to Netflix, these predictions were accurate 75 percent of the time, and half of Netflix users who rented CineMatch- recommended movies gave them a five-star rating. The algorithm considered:

- The films themselves, which are arranged as groups of common movies

- The customers' ratings, rented movies and current queue

- The combined ratings of all Netflix users

In 2006, Netflix announced the Netflix Prize, a machine learning, and data mining competition for movie rating prediction. For the result, Netflix looked at the two underlying algorithms with the best 15 performance in the ensemble: Matrix Factorization (which the community generally called SVD, Singular Value Decomposition) and Restricted Boltzmann Machines (RBM). SVD by itself provided a 0.8914 root mean squared error, while RBM alone provided a competitive but slightly worse 0.8990 RMSE. A linear blend of these two reduced the error to 0.88.
To put these algorithms to use, Netflix had to work to overcome some limitations, such as having been built to handle 100 million ratings instead of the more than 5 billion we have, and not being built to adapt as members added more ratings. But after Netflix resolved those obstacles, they put in place the two algorithms where they are now being used as part of our recommendation engine. Netflix launched an instant streaming service in 2007, one year after the Netflix Prize began. Apart from Netflix's popularity and rating prediction, we have tried many other features at Netflix. Some have shown no positive effect while others have improved our ranking accuracy tremendously. The graph below shows the ranking improvement we have obtained by adding different features and optimizing the machine learning algorithm.



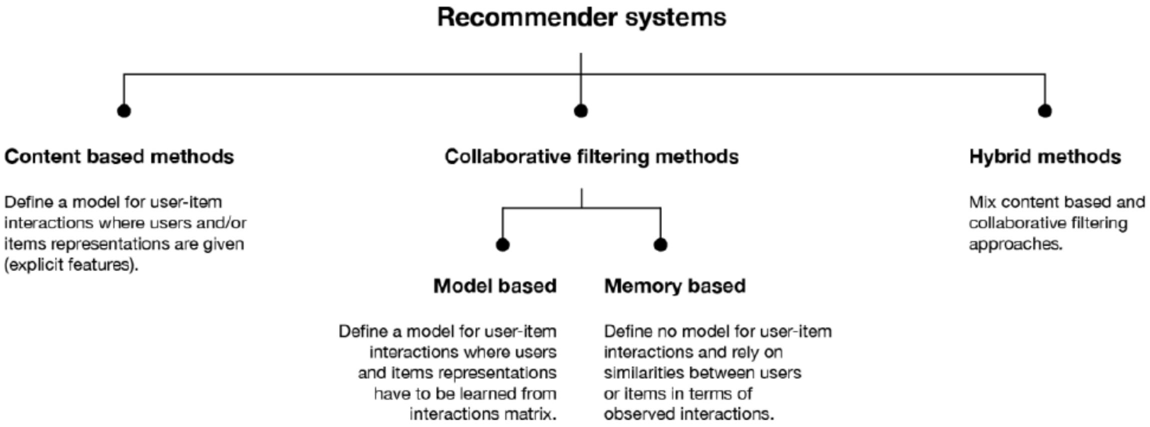
5. WORKFLOW

TYPES OF RECOMMENDER SYSTEMS

The following are the types of Recommender systems most widely used:

- Content Based Methods
- Hybrid Methods
- Collaborative Methods
  - Model Based Methods (Item Based)
  - Memory Based Methods (User Based)

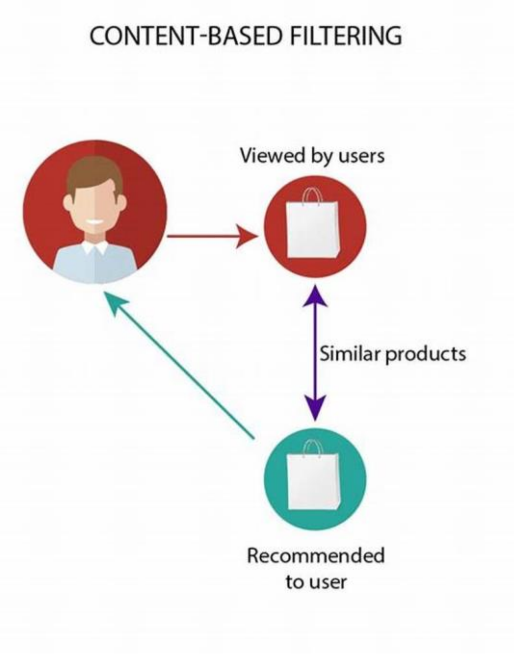
Figure 4 : Types of Recommender Systems


Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.
Hybrid methods
Mix content based and collaborative filtering approaches.

CONTENT BASED FILTERING TECHNIQUES:

In content-based filtering, items are recommended based on comparisons between item profile and user profile. A user profile is content that is found to be relevant to the user in the form of keywords (or features). A user profile might be seen as a set of assigned keywords (terms, features) collected by algorithm from items found relevant (or interesting) by the user. A set of keywords (or features) of an item is the Item profile.

For example, consider a scenario in which a person goes to buy his favorite cake 'X' to a pastry. Unfortunately, cake 'X' has been sold out and because of this the shopkeeper recommends the person to buy cake 'Y' which is made up of ingredients like cake 'X'. This is an instance of content-based filtering.



PROS OF CONTENT – BASED FILTERING

Advantages of content-based filtering are:

- They can recommend unrated items.
- We can easily explain the working of recommender system by listing the content features of an item.
- Content-based recommender systems need only the rating of the concerned user and not any other user of the system.

CONS OF CONTENT – BASED FILTERING

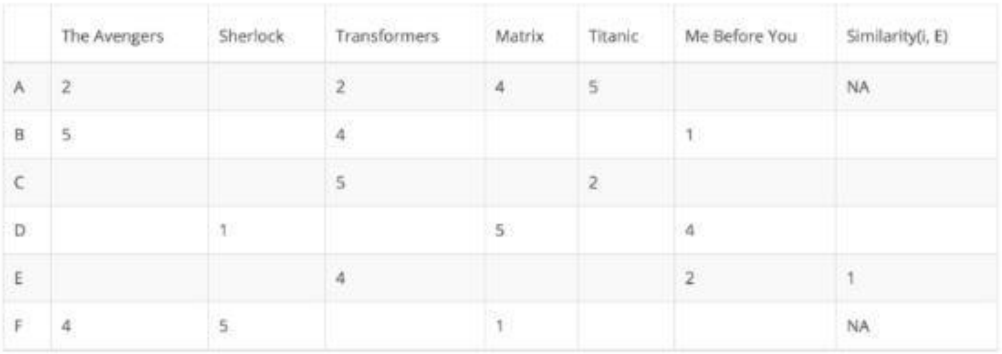
- It does not work for a new user who has not rated any item yet as enough ratings are required content-based recommender evaluates the user preferences and provides accurate recommendations.
- No recommendation of serendipitous items.
- Limited content analysis – The recommendation does not work if the system fails to distinguish the items a user likes from the user he doesn't not like.

COLLABORATIVE BASED FILTERING TECHNIQUES:

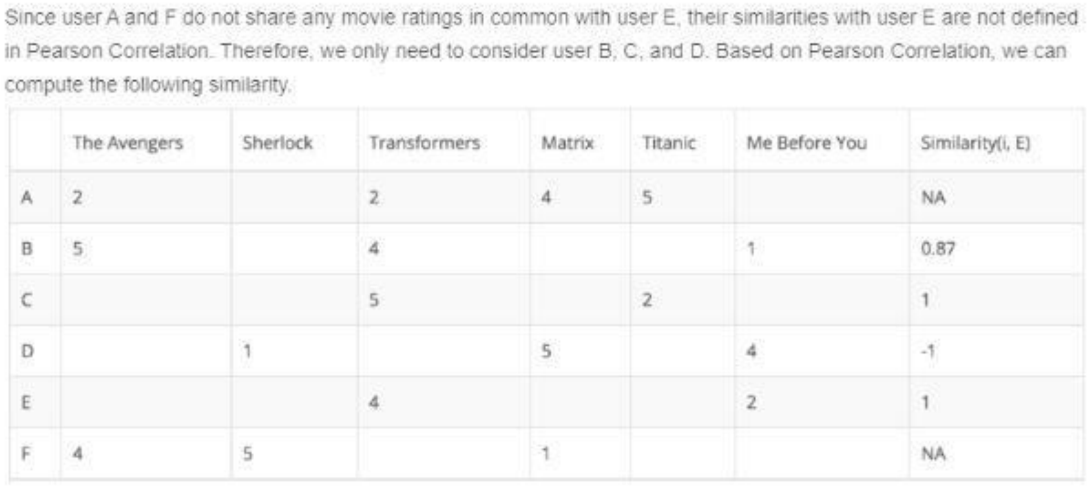
Our content-based engine suffers from some severe limitations. It is only capable of suggesting movies which are close to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres.
Also, the engine that we built is not personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who she/he is.
Therefore, in this section, we will use a technique called Collaborative Filtering to make recommendations to Movie Watchers.

It is basically of two types: -

User based filtering- These systems recommend products to a user that similar users have liked. For measuring the similarity between two users we can either use Pearson correlation or cosine similarity. This filtering technique can be illustrated with an example. In the following matrix's, each row represents a user, while the columns correspond to different movies except the last one which records the similarity between that user and the target user. Each cell represents the rating that the user gives to that movie. Assume user E is the target.



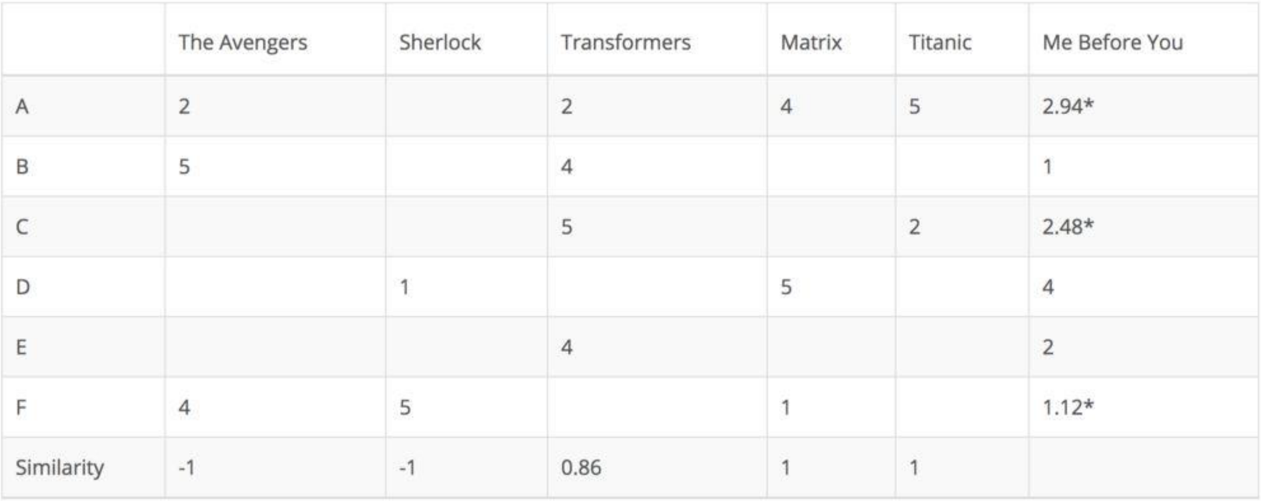
	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	
C			5		2		
D		1		5		4	
E			4			2	1
F	4	5		1			NA



Since user A and F do not share any movie ratings in common with user E, their similarities with user E are not defined in Pearson Correlation. Therefore, we only need to consider user B, C, and D. Based on Pearson Correlation, we can compute the following similarity.

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E			4			2	1
F	4	5		1			NA

Item Based Collaborative Filtering- Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank vertically, as opposed to the horizontal manner that user-based CF does. The following table shows how to do so for the movie Me Before.



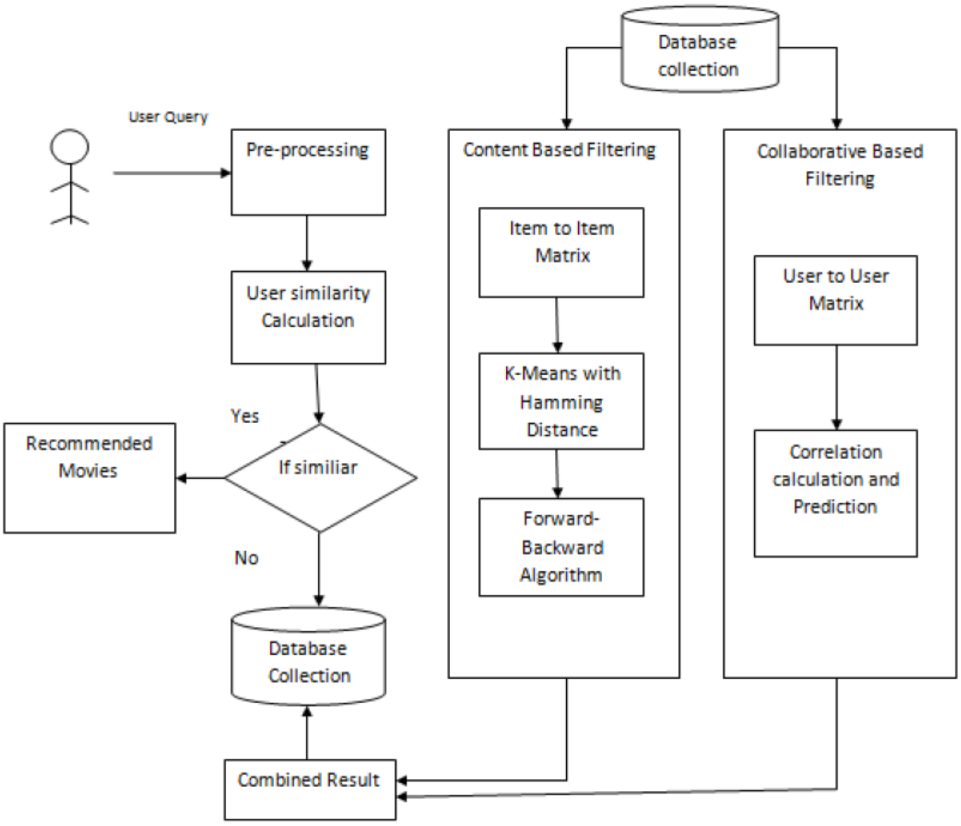
	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You
A	2		2	4	5	2.94*
B	5		4			1
C			5		2	2.48*
D		1		5		4
E			4			2
F	4	5		1		1.12*
Similarity	-1	-1	0.86	1	1	

It avoids the problem posed by dynamic user preference as item-based CF is more static. However, several problems remain for this method. First, the main issue is scalability. Computation grows with both the customer and the product. The worst-case complexity is O(mn) with m users and n items. In addition, sparsity is another concern. Look at the above table again. Although there is only one user that rated both Matrix and Titanic rated, the similarity between them is 1. In extreme cases, we can have millions of users and the similarity between two different movies could be very high simply because they have similar rank for the only user who ranked them both.

PROS OF COLLABORATIVE – BASED FILTERING

One significant advantage of collaborative filtering is that it doesn’t need to analyze or understand the object (products, films, books) to recommend complex items precisely. There is no dependence on analyzable machine content, which means it chooses recommendations based on what it knows about the user.

6. SYSTEM ARCHITECTURE



DATA COLLECTION

Download the dataset from Kaggle.

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

Dataset consists of movies_metadata.csv, keywords.csv, credits.csv, links.csv, links_small.csv, and ratings_small.csv for this project. Make sure the dataset is large and diverse enough to capture a wide range of films and user preferences.

DATA PRE-PROCESSING

For pre-processing, connect the data access from various files. Handle missing values, duplication, and inconsistencies in the movie dataset. Transform the dataset into a format that recommendation engines can understand. Using approaches from NLP and Regular Expressions improve the data's use for the recommendation system.

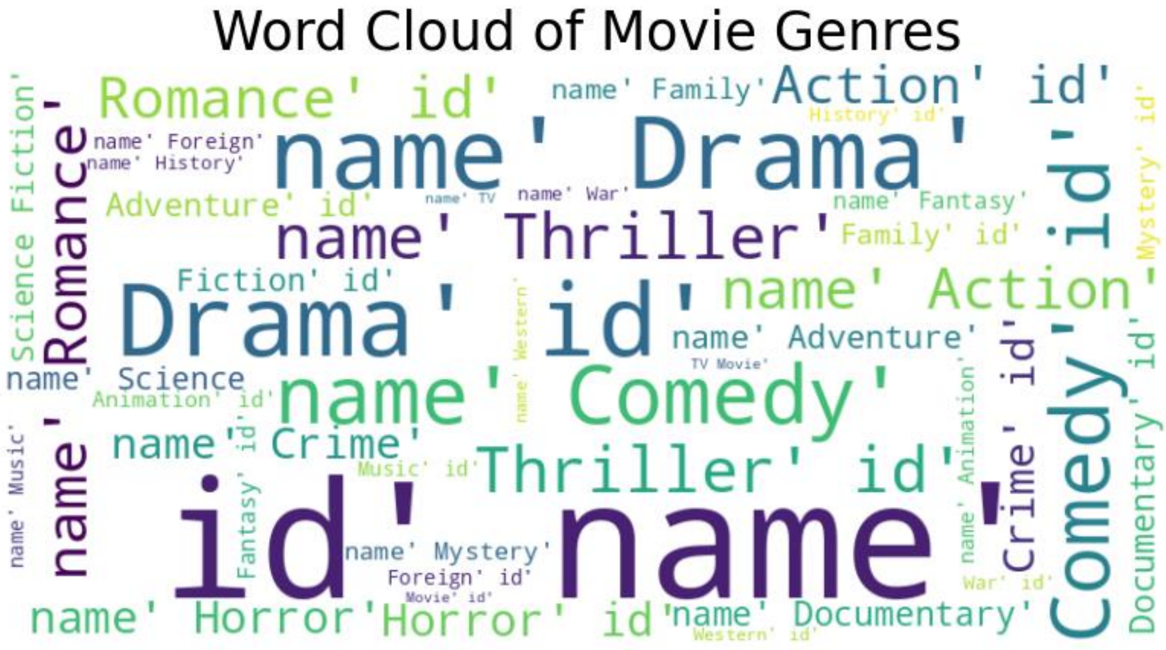
EXPLORATORY DATA ANALYSIS

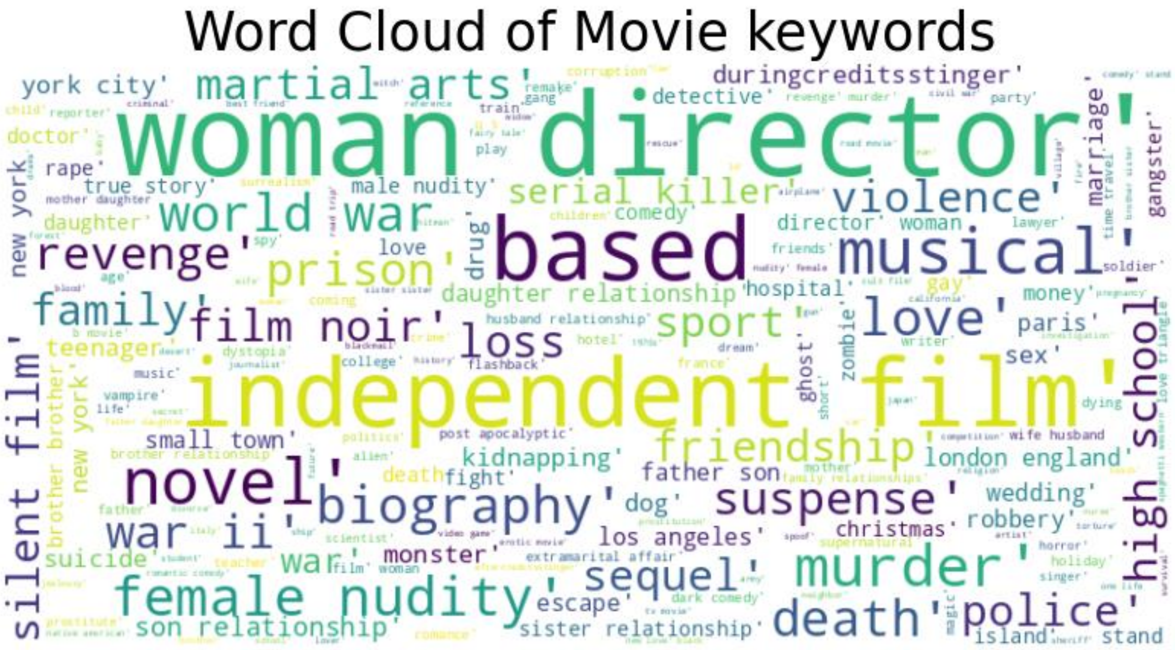
To get insights into the movie dataset, use exploratory analysis. Visualise and summarise movie attributes, and look for any noticeable patterns, trends, or similarities. Examine the collinearity of the data and perform univariate and bivariate analyses to learn more about each property.

7. EXPLORATORY DATA ANALYSIS

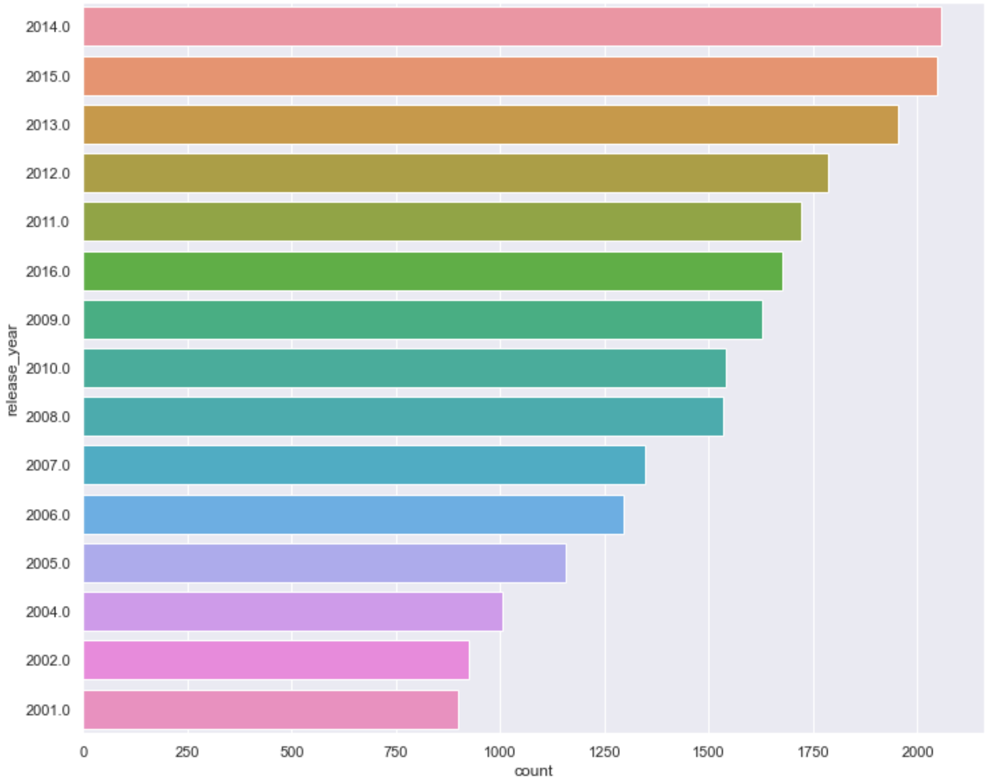
- All the features of the new Data frame were analysed and visualized using visualization techniques.
- Python libraries like MATPLOTLIB and SEABORN was used for this purpose.
- Univariate, Bivariate and Multivariate analysis was done to understand the data and to draw meaningful insights from the data.

Here are some of the plots from the Analysis.

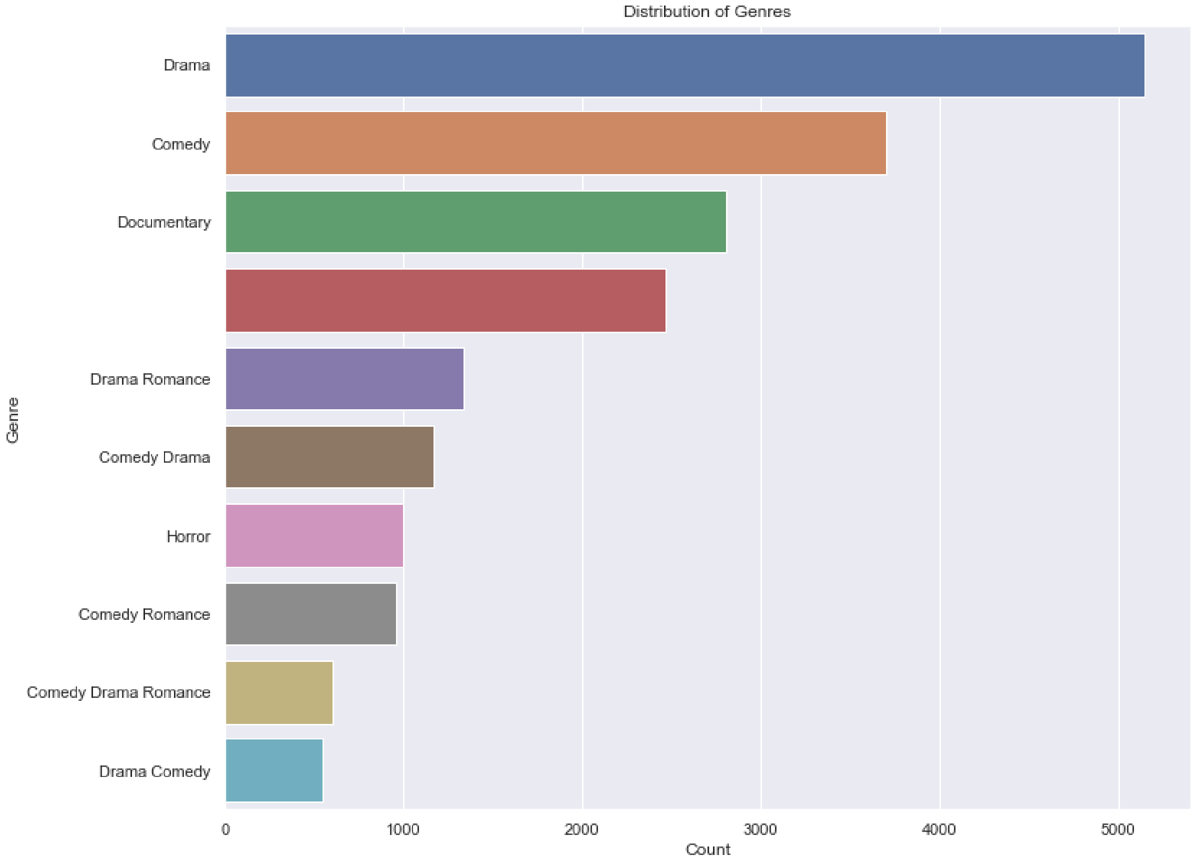




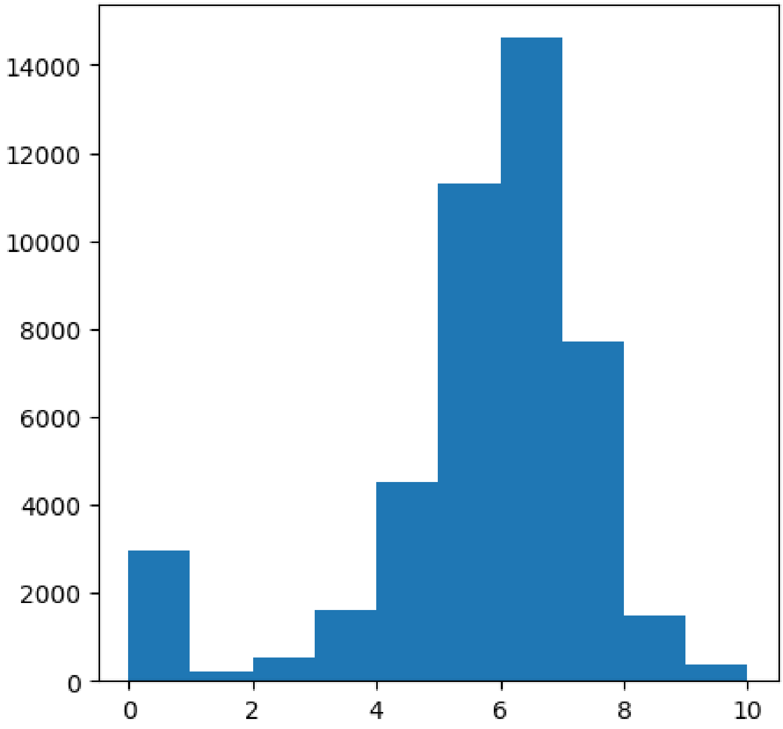
DISTRIBUTION OF MOVIES BASED ON YEAR



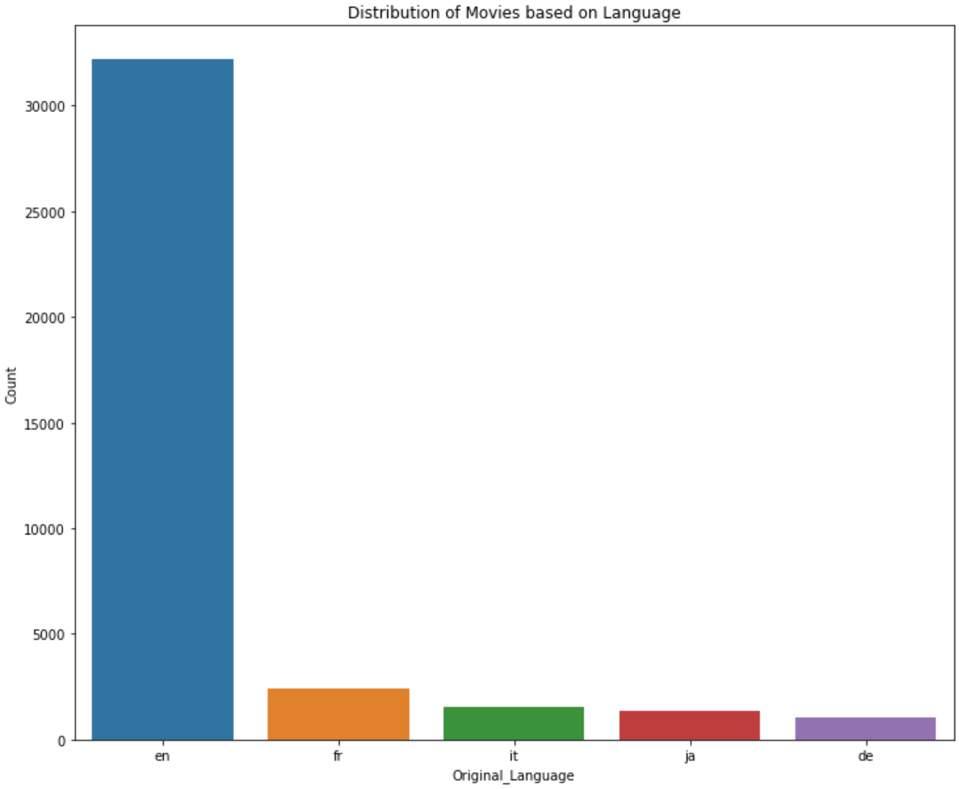
DISTRIBUTION OF MOVIES BASED ON GENRES



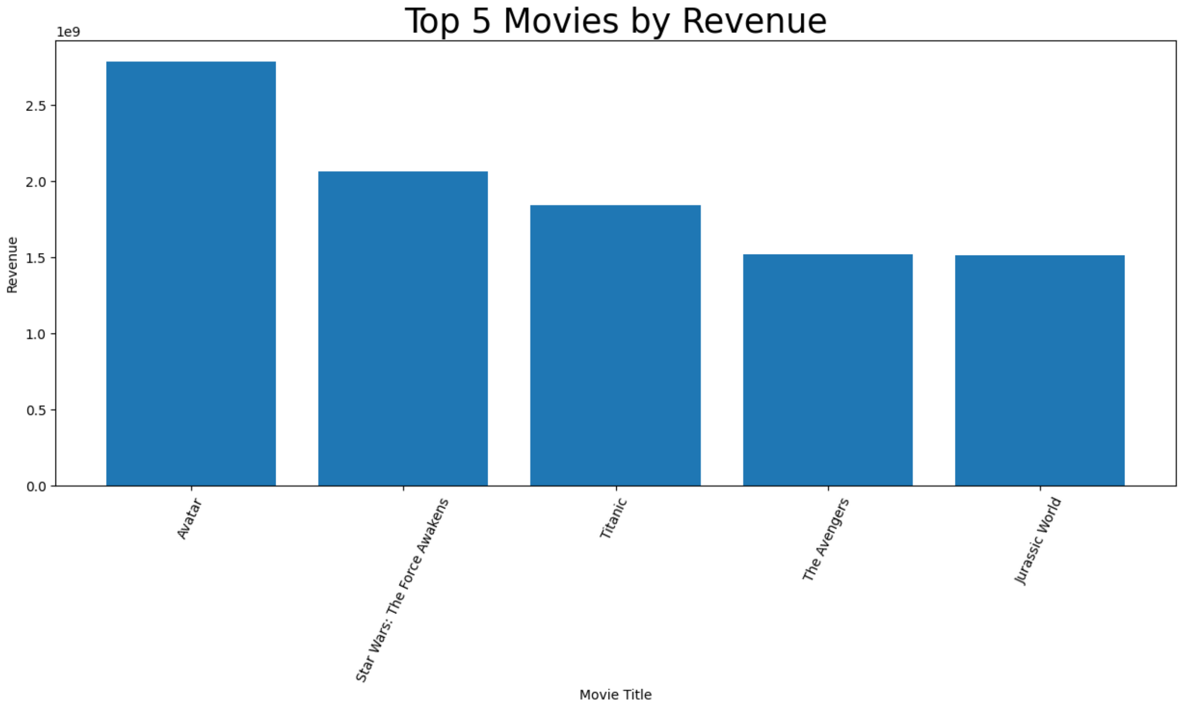
HISTOGRAM PLOT FOR VOTE_AVERAGE



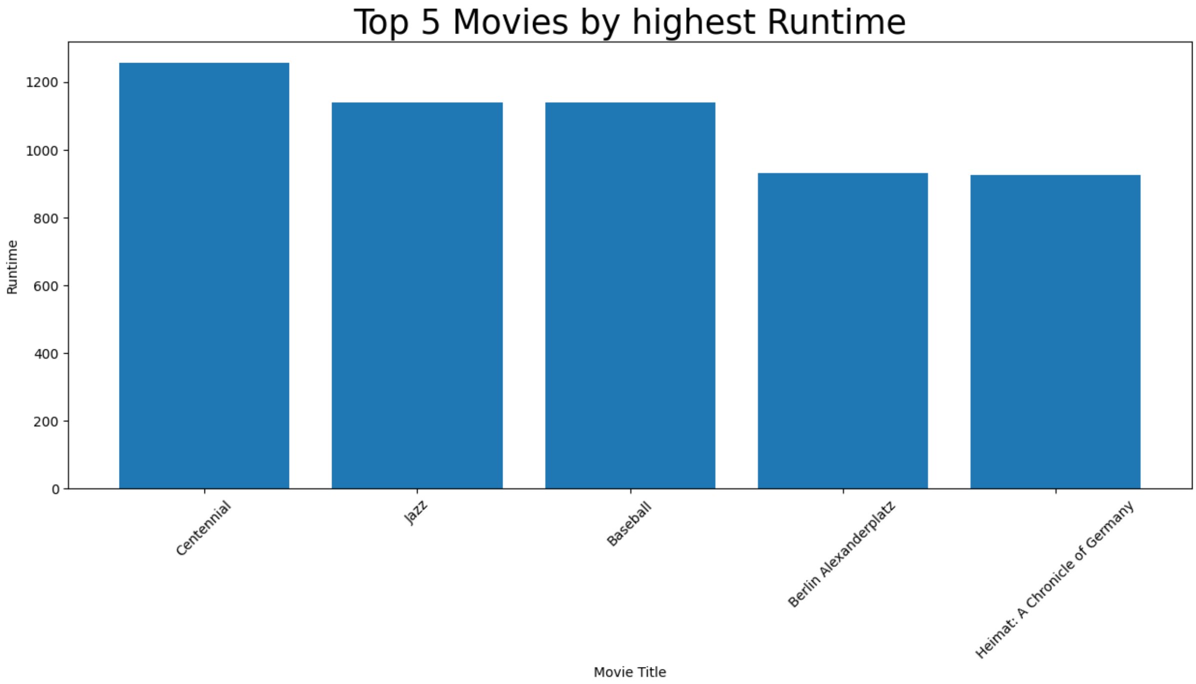
DISTRIBUTION OF MOVIES BASED ON LANGUAGE



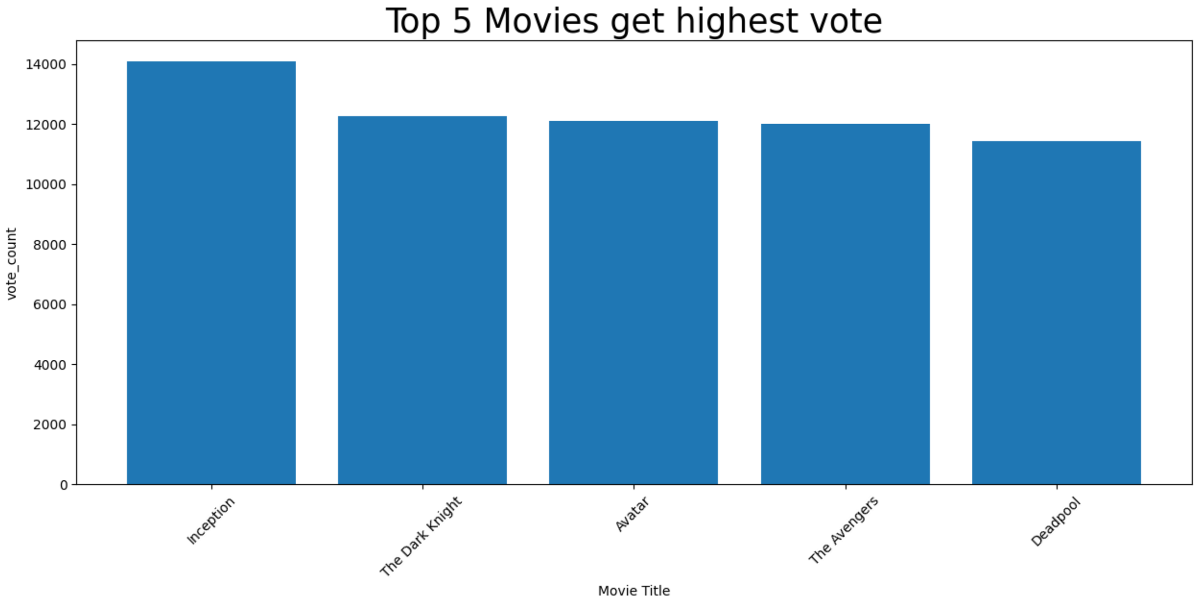
TOP 5 MOVIES BASED ON REVENUE



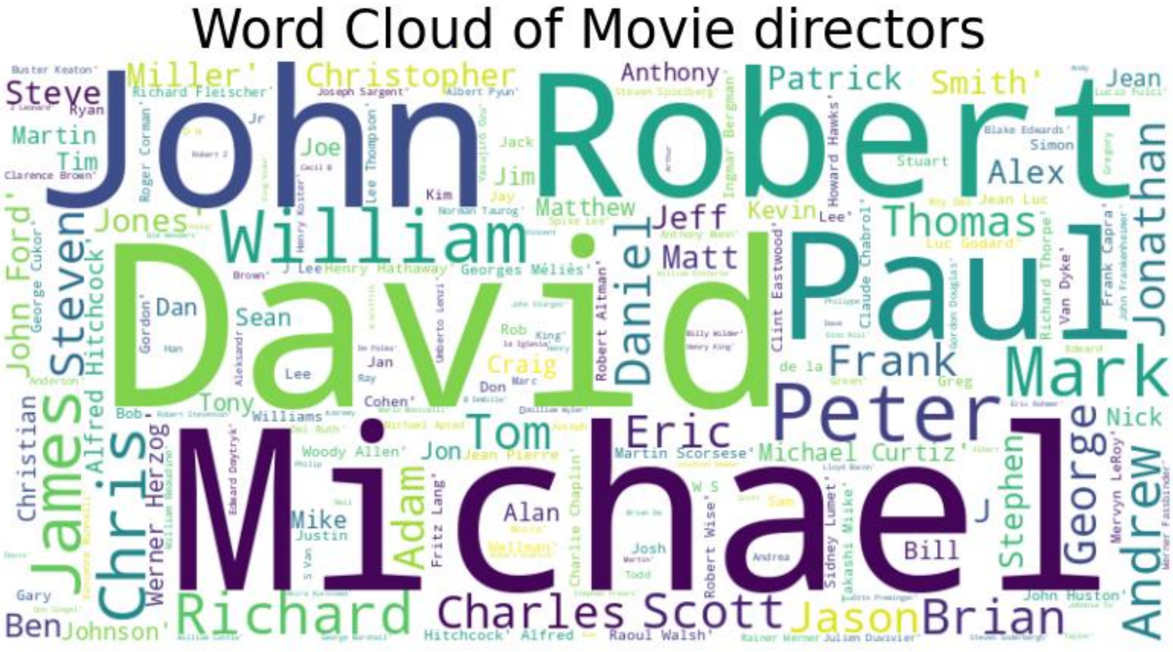
TOP 5 MOVIES BASED ON HIGHEST RUNTIME



TOP 5 MOVIES WITH HIGHEST VOTE_COUNT



WORD CLOUD ANALYSIS FOR MOVIE DIRECTORS



8. RECOMMENDATION ALGORITHMS:

Depending on the needs of the project and the resources at hand, consider and put various suggestion strategies into practise.

- Collaborative Filtering: Create collaborative filtering algorithms based on user or item preferences or item characteristics to recommend films.
- Content-Based Filtering: Build models that use movie attributes such as genre, actors, directors, or keywords to propose films with similar content.
- Hybrid Approaches: To deliver more accurate and diversified recommendations, combine collaborative and content-based filtering algorithms**.**

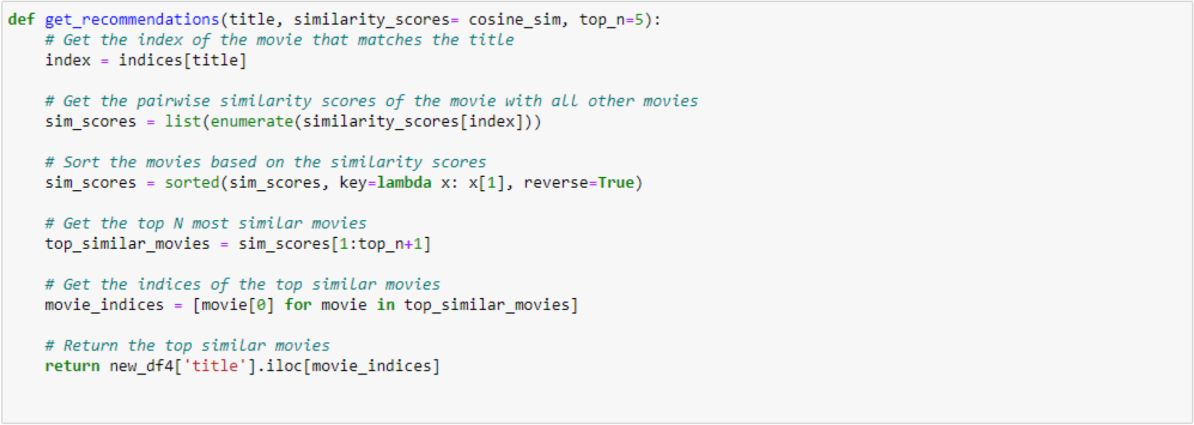
9. MODEL TRAINING AND EVALUATION:

To examine the performance of recommendation algorithms, divide the movie dataset into training and testing sets. Using the training data, train the selected recommendation models.

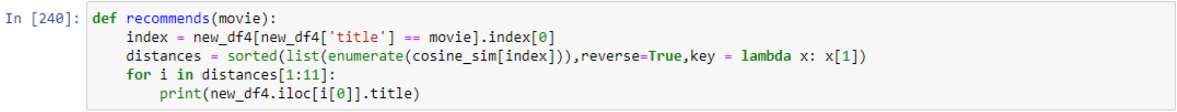
Use relevant assessment metrics like as accuracy, recall, or mean average precision (MAP) to evaluate and compare the performance of different models.

For our model we are going to use Cosine Similarity and Linear Kernel to train our Model and provide recommendations.

CODE SNIPPET TO GET RECOMMENDATIONS USING COSINE SIMILARITY



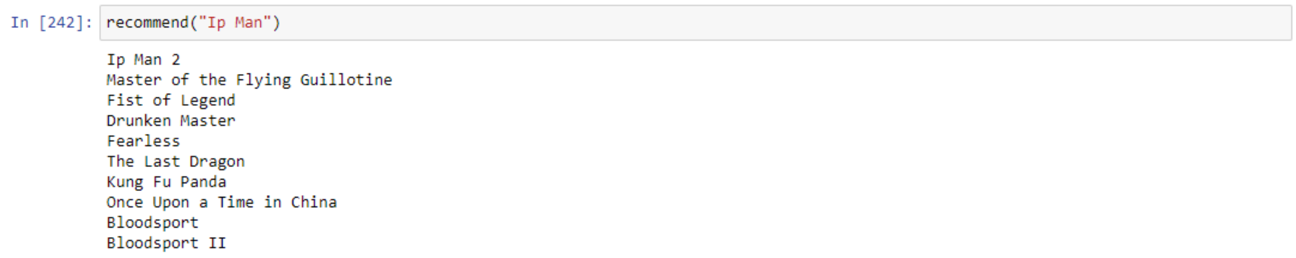
```
def get_recommendations(title, similarity_scores= cosine_sim, top_n=5):
    # Get the index of the movie that matches the title
    index = indices[title]

    # Get the pairwise similarity scores of the movie with all other movies
    sim_scores = list(enumerate(similarity_scores[index]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the top N most similar movies
    top_similar_movies = sim_scores[1:top_n+1]

    # Get the indices of the top similar movies
    movie_indices = [movie[0] for movie in top_similar_movies]

    # Return the top similar movies
    return new_df4['title'].iloc[movie_indices]
```



```
In [240]: def recommends(movie):
              index = new_df4[new_df4['title'] == movie].index[0]
              distances = sorted(list(enumerate(cosine_sim[index])),reverse=True,key = lambda x: x[1])
              for i in distances[1:11]:
                  print(new_df4.iloc[i[0]].title)
```

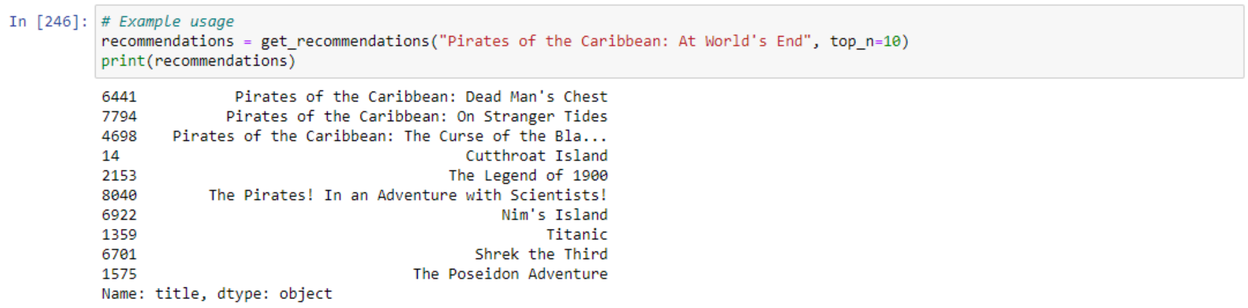
OUTPUT

MOVIE RECOMMENDATION OUTPUT FOR “IP-MAN”



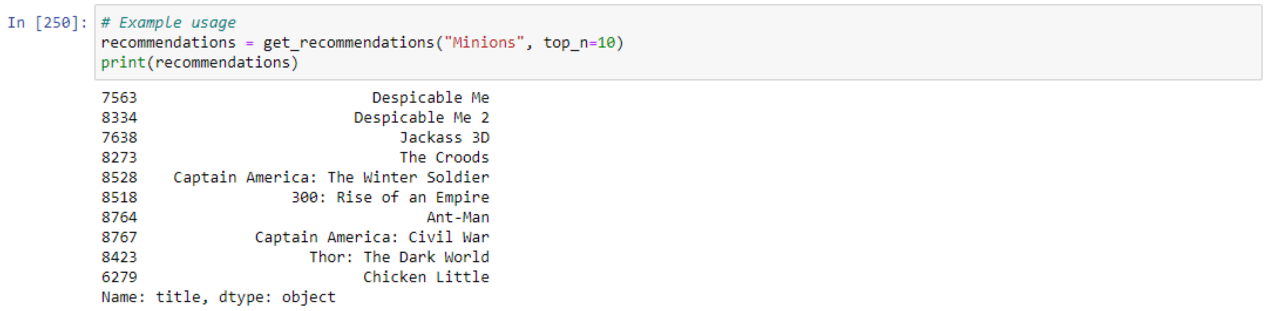
```
In [242]: recommend("Ip Man")

          Ip Man 2
          Master of the Flying Guillotine
          Fist of Legend
          Drunken Master
          Fearless
          The Last Dragon
          Kung Fu Panda
          Once Upon a Time in China
          Bloodsport
          Bloodsport II
```

MOVIE RECOMMENDATION OUTPUT FOR “PIRATES OF CARIBEAN”



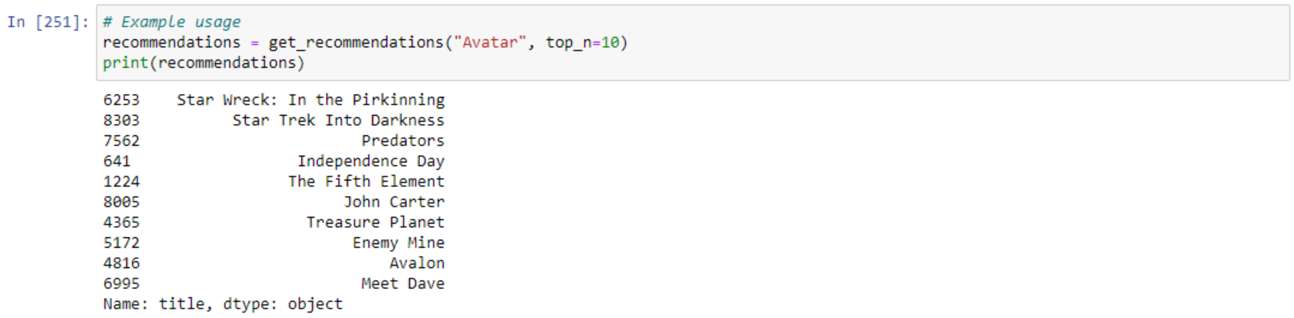
```
In [246]: # Example usage
          recommendations = get_recommendations("Pirates of the Caribbean: At World's End", top_n=10)
          print(recommendations)

          6441            Pirates of the Caribbean: Dead Man's Chest
          7794           Pirates of the Caribbean: On Stranger Tides
          4698    Pirates of the Caribbean: The Curse of the Bla...
          14                                      Cutthroat Island
          2153                                  The Legend of 1900
          8040         The Pirates! In an Adventure with Scientists!
          6922                                        Nim's Island
          1359                                             Titanic
          6701                                     Shrek the Third
          1575                              The Poseidon Adventure
          Name: title, dtype: object
```

MOVIE RECOMMENDATION OUTPUT FOR “MINIONS”



```
In [250]: # Example usage
          recommendations = get_recommendations("Minions", top_n=10)
          print(recommendations)

          7563                          Despicable Me
          8334                        Despicable Me 2
          7638                             Jackass 3D
          8273                             The Croods
          8528    Captain America: The Winter Soldier
          8518                 300: Rise of an Empire
          8764                                Ant-Man
          8767             Captain America: Civil War
          8423                   Thor: The Dark World
          6279                         Chicken Little
          Name: title, dtype: object
```

MOVIE RECOMMENDATION OUTPUT FOR “AVATAR”



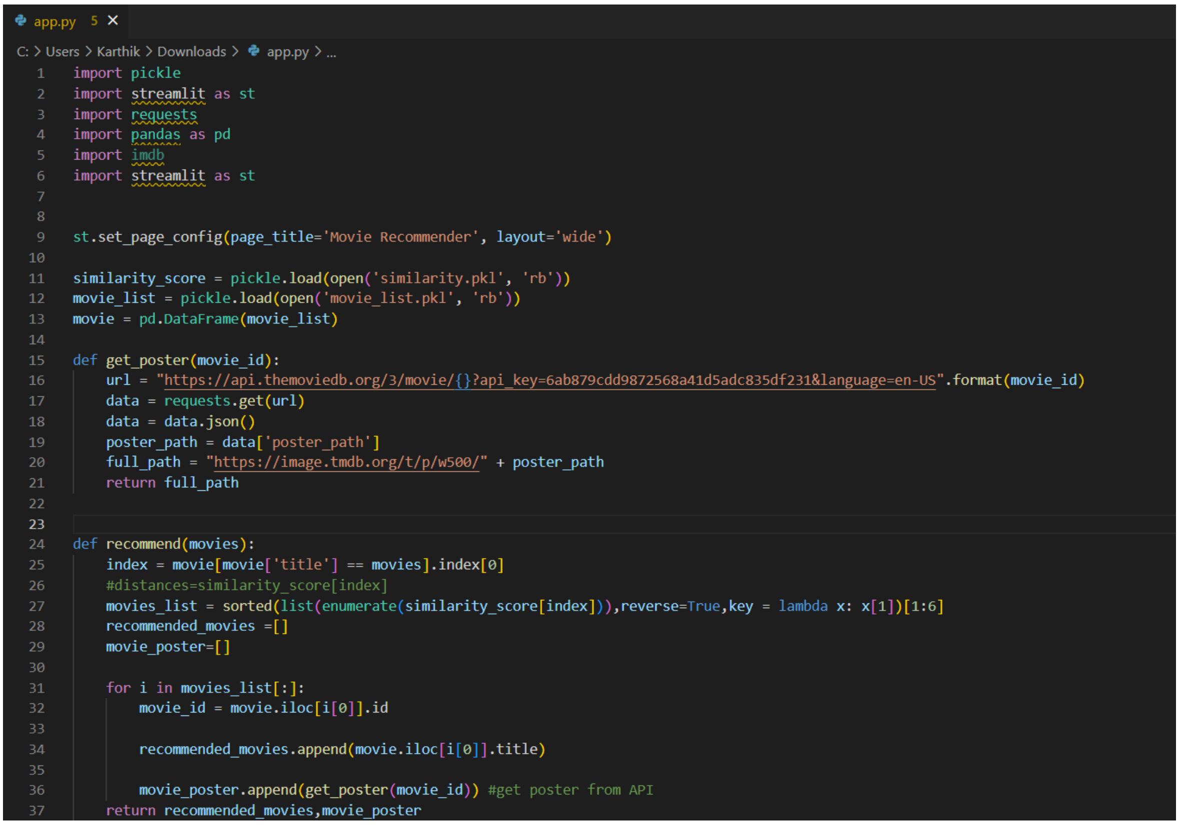
```
In [251]: # Example usage
          recommendations = get_recommendations("Avatar", top_n=10)
          print(recommendations)

          6253    Star Wreck: In the Pirkinning
          8303          Star Trek Into Darkness
          7562                        Predators
          641                  Independence Day
          1224                The Fifth Element
          8005                      John Carter
          4365                  Treasure Planet
          5172                       Enemy Mine
          4816                           Avalon
          6995                        Meet Dave
          Name: title, dtype: object
```

10. SYSTEM DESIGN AND IMPLEMENTATION:

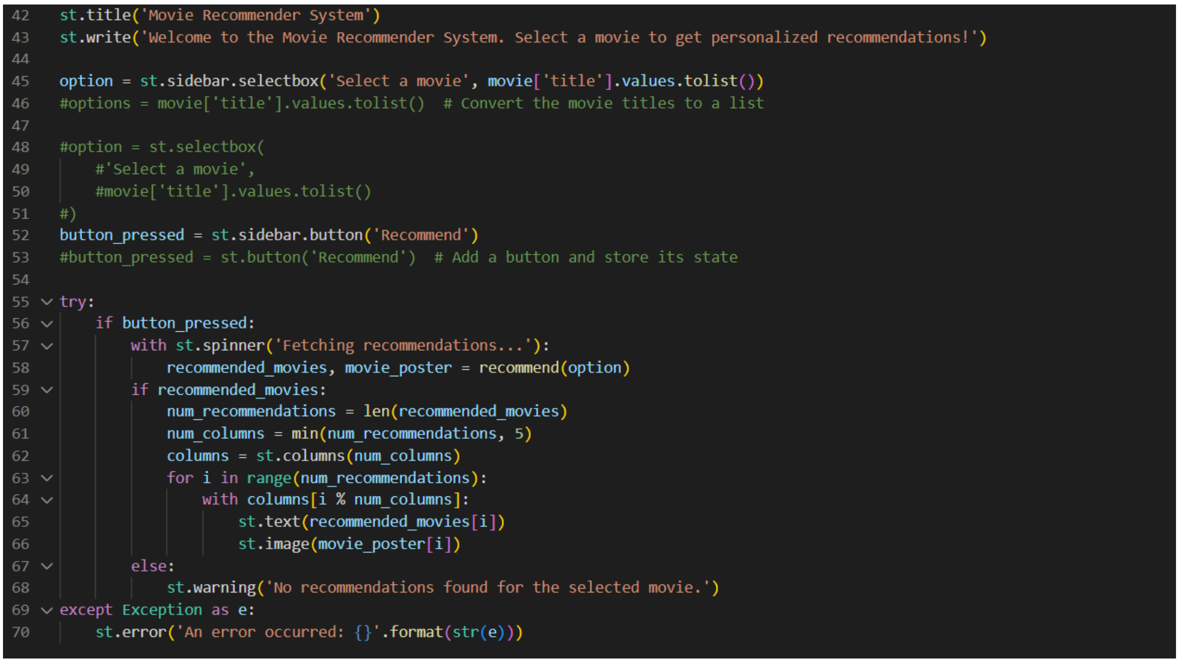
Using streamlit, create a web application that allows users to interact with the recommendation system. Include options for users to score films, submit feedback, and view personalised suggestions.

Ensure that the interface is user-friendly, responsive, and visually appealing.

INPUT CODE APP.PY FILE

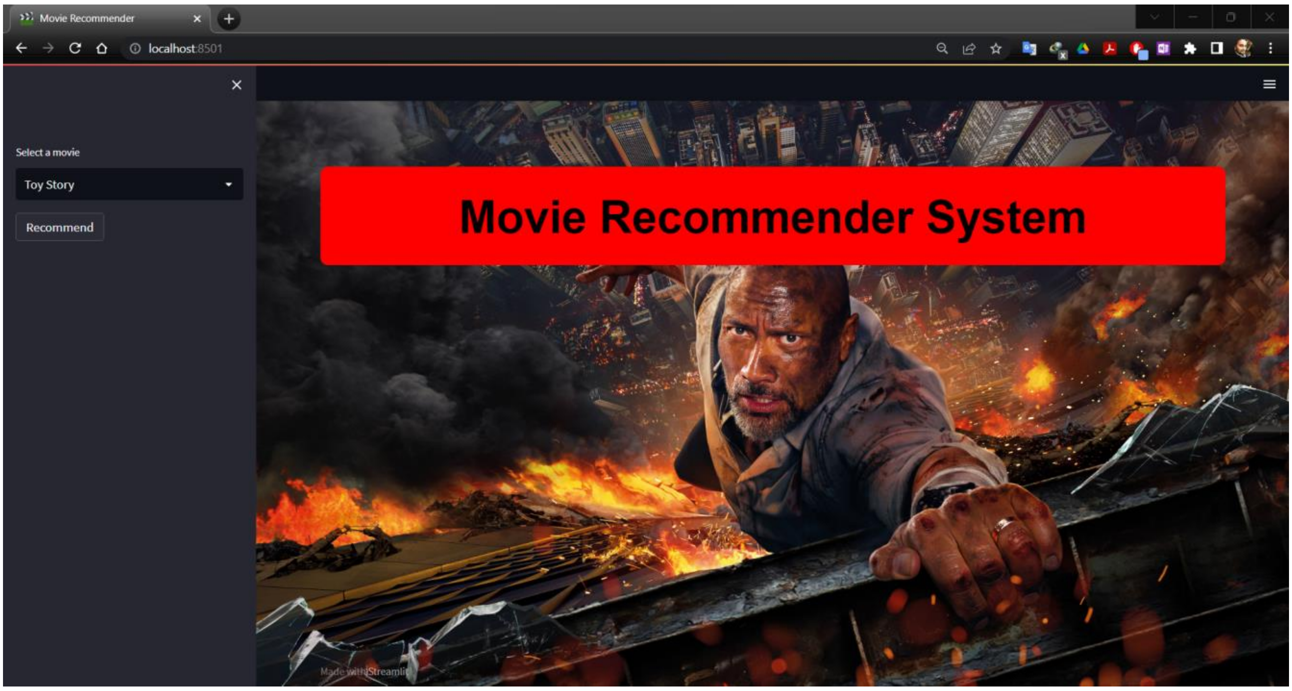


```python
import pickle
import streamlit as st
import requests
import pandas as pd
import imdb
import streamlit as st


st.set_page_config(page_title='Movie Recommender', layout='wide')

similarity_score = pickle.load(open('similarity.pkl', 'rb'))
movie_list = pickle.load(open('movie_list.pkl', 'rb'))
movie = pd.DataFrame(movie_list)

def get_poster(movie_id):
    url = "https://api.themoviedb.org/3/movie/{}?api_key=6ab879cdd9872568a41d5adc835df231&language=en-US".format(movie_id)
    data = requests.get(url)
    data = data.json()
    poster_path = data['poster_path']
    full_path = "https://image.tmdb.org/t/p/w500/" + poster_path
    return full_path


def recommend(movies):
    index = movie[movie['title'] == movies].index[0]
    #distances=similarity_score[index]
    movies_list = sorted(list(enumerate(similarity_score[index])),reverse=True,key = lambda x: x[1])[1:6]
    recommended_movies =[]
    movie_poster=[]

    for i in movies_list[:]:
        movie_id = movie.iloc[i[0]].id

        recommended_movies.append(movie.iloc[i[0]].title)

        movie_poster.append(get_poster(movie_id)) #get poster from API
    return recommended_movies,movie_poster
```



```python
st.title('Movie Recommender System')
st.write('Welcome to the Movie Recommender System. Select a movie to get personalized recommendations!')

option = st.sidebar.selectbox('Select a movie', movie['title'].values.tolist())
#options = movie['title'].values.tolist()  # Convert the movie titles to a list

#option = st.selectbox(
    #'Select a movie',
    #movie['title'].values.tolist()
#)
button_pressed = st.sidebar.button('Recommend')
#button_pressed = st.button('Recommend')  # Add a button and store its state

try:
    if button_pressed:
        with st.spinner('Fetching recommendations...'):
            recommended_movies, movie_poster = recommend(option)
        if recommended_movies:
            num_recommendations = len(recommended_movies)
            num_columns = min(num_recommendations, 5)
            columns = st.columns(num_columns)
            for i in range(num_recommendations):
                with columns[i % num_columns]:
                    st.text(recommended_movies[i])
                    st.image(movie_poster[i])
        else:
            st.warning('No recommendations found for the selected movie.')
except Exception as e:
    st.error('An error occurred: {}'.format(str(e)))
```

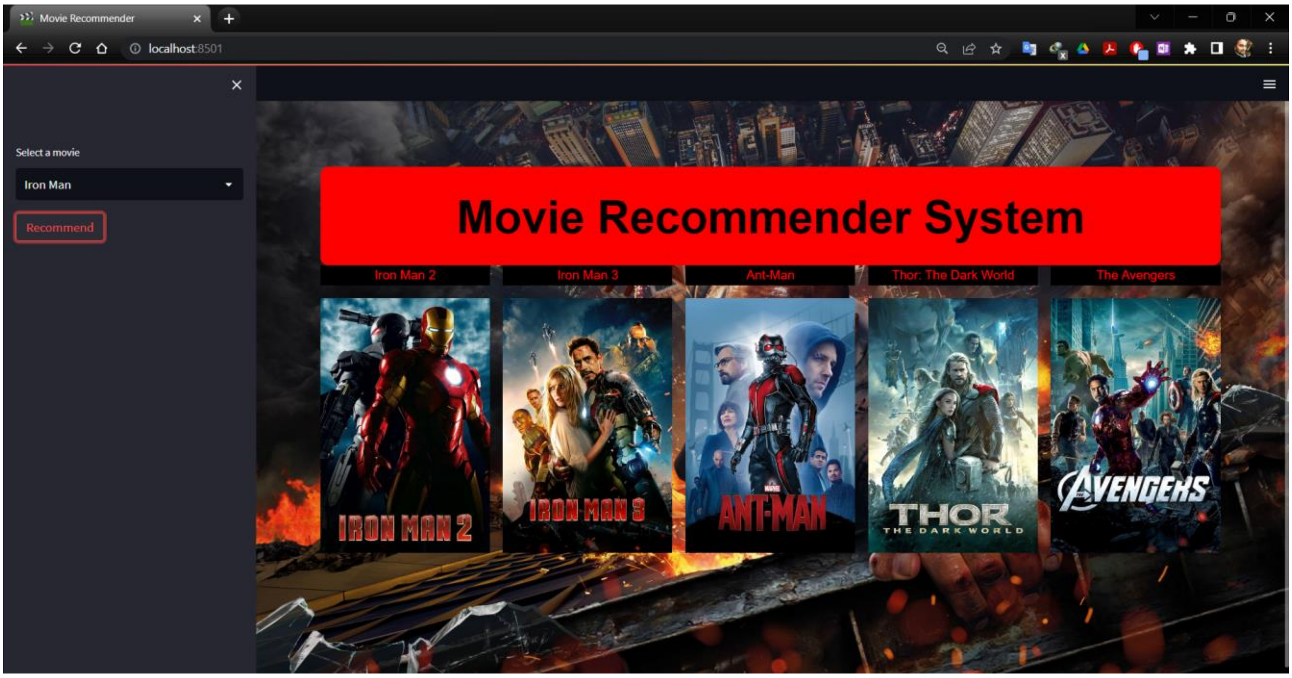
RECOMMENDATIONS FROM THE STREAMLIT APP



Select a movie

Toy Story

WEB APPLICATION UI – FRONTEND



Select a movie

Iron Man

Iron Man 2

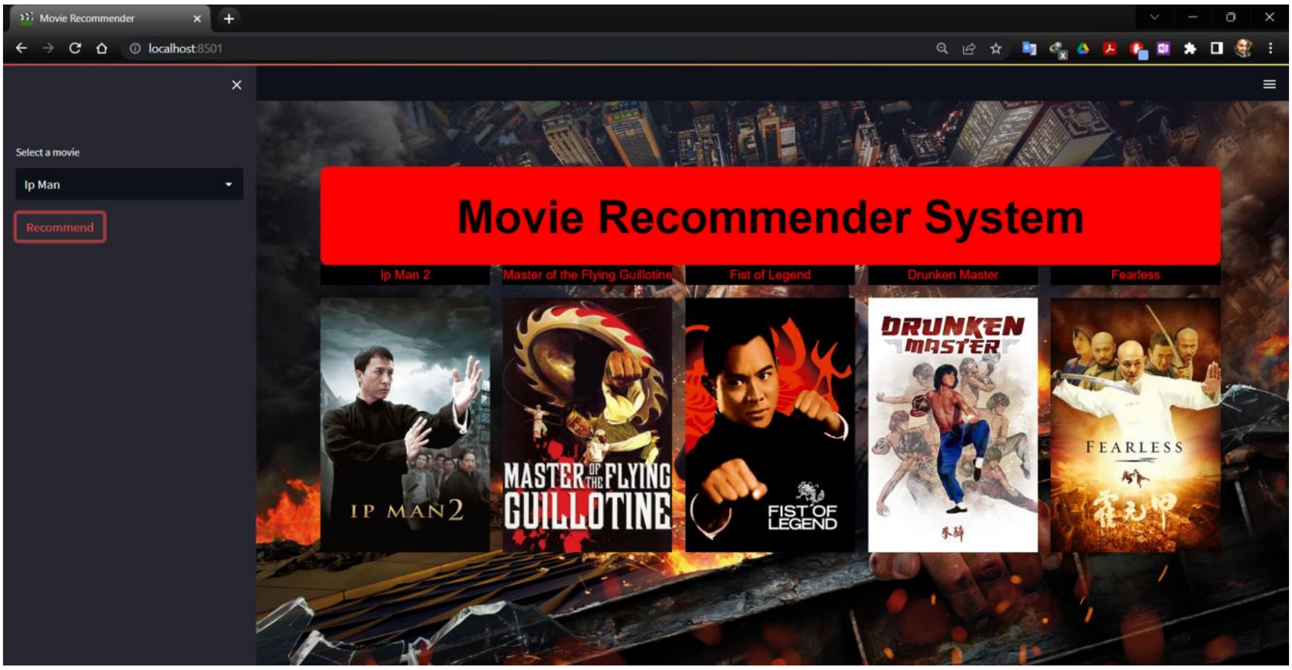
Iron Man 3

Ant-Man

Thor: The Dark World

The Avengers

TOP 5 RECOMMENDATIONS FOR MOVIE “IRON MAN”



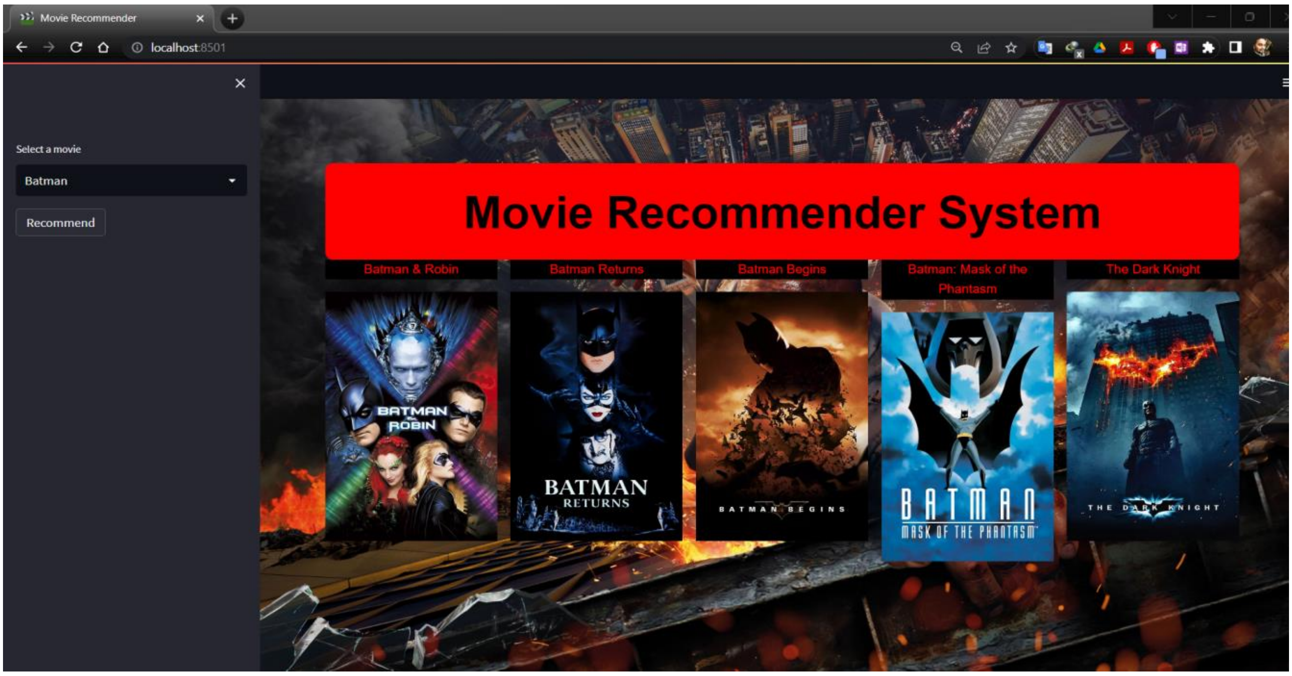
Select a movie

Ip Man

Movie Recommender System

Ip Man 2 | Master of the Flying Guillotine | Fist of Legend | Drunken Master | Fearless

TOP 5 RECOMMENDATION FOR MOVIE “IP – MAN”



Select a movie

Batman

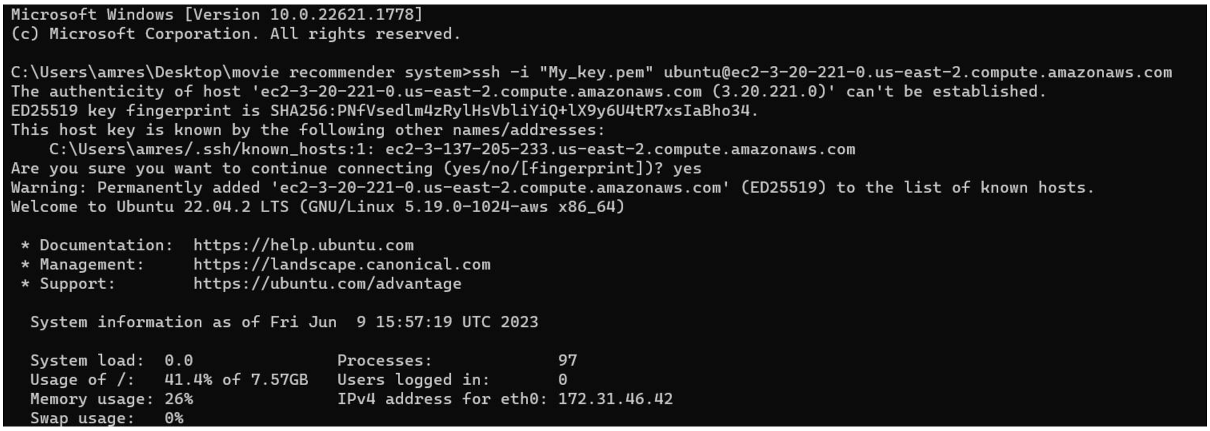
Movie Recommender System

Batman & Robin | Batman Returns | Batman Begins | Batman: Mask of the Phantasm | The Dark Knight

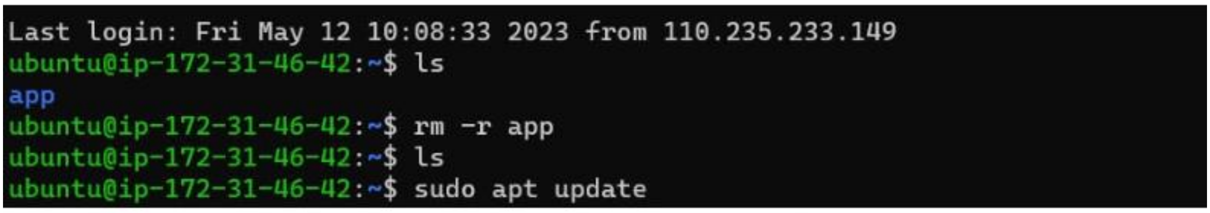
TOP 5 RECOMMENDATION FOR MOVIE “BATMAN”

11. AWS IMPLEMENTATION:

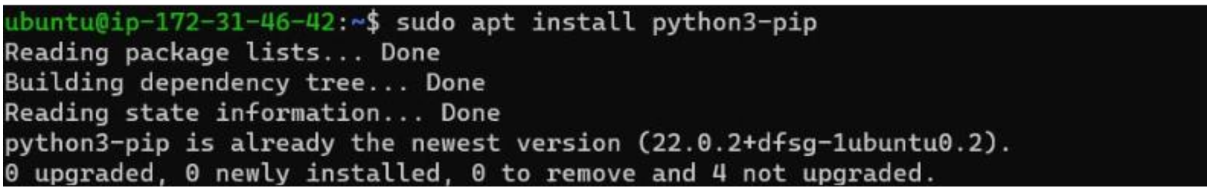
CODE SNIPPET OF AWS IMPLEMENTATION



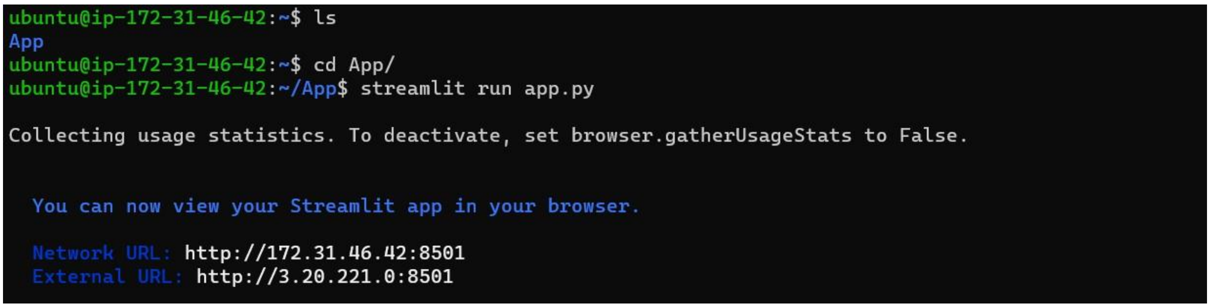
```
Microsoft Windows [Version 10.0.22621.1778]
(c) Microsoft Corporation. All rights reserved.

C:\Users\amres\Desktop\movie recommender system>ssh -i "My_key.pem" ubuntu@ec2-3-20-221-0.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-3-20-221-0.us-east-2.compute.amazonaws.com (3.20.221.0)' can't be established.
ED25519 key fingerprint is SHA256:PNfVsedlm4zRylHsVbliYiQ+lX9y6U4tR7xsIaBho34.
This host key is known by the following other names/addresses:
    C:\Users\amres/.ssh/known_hosts:1: ec2-3-137-205-233.us-east-2.compute.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-20-221-0.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jun  9 15:57:19 UTC 2023

  System load:  0.0               Processes:             97
  Usage of /:   41.4% of 7.57GB   Users logged in:       0
  Memory usage: 26%               IPv4 address for eth0: 172.31.46.42
  Swap usage:   0%
```



```
Last login: Fri May 12 10:08:33 2023 from 110.235.233.149
ubuntu@ip-172-31-46-42:~$ ls
app
ubuntu@ip-172-31-46-42:~$ rm -r app
ubuntu@ip-172-31-46-42:~$ ls
ubuntu@ip-172-31-46-42:~$ sudo apt update
```



```
ubuntu@ip-172-31-46-42:~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-pip is already the newest version (22.0.2+dfsg-1ubuntu0.2).
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
```



```
ubuntu@ip-172-31-46-42:~$ ls
App
ubuntu@ip-172-31-46-42:~$ cd App/
ubuntu@ip-172-31-46-42:~/App$ streamlit run app.py

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.


  You can now view your Streamlit app in your browser.

  Network URL: http://172.31.46.42:8501
  External URL: http://3.20.221.0:8501
```

12. CHALLENGES FACED

- TYPECASTING OF FEATURES

Some columns contained mixed datatypes with categorical and datetime where the actual datatype is int64.It was handled by deleting the rows with datetime and was typecasted to integer.

- While running visualizations on streamlit some of the plots were not supported. They were handled by changing the plots which had support.
- While doing content-based filtering, due to large number of rows in the features column error was thrown which indicated lack of memory. Hence, we have used overview feature after doing all necessary preprocessing to allocate memory to run within the limits available.
- While running the applications for some input values key value error and other types of errors have been raised. Initially it was a challenge to figure it out but later understood that some of the movie titles were same maybe some movies released with similar movie names in different years. All the duplicated values are dropped by which error was resolved.

13. FUTURE SCOPE

Collect regular user feedback to improve the accuracy and relevance of the recommendation system.

To provide up-to-date recommendations, the system should be updated on a regular basis with new movie releases, user preferences, and evolving algorithms. Monitor system performance and make necessary optimisations.

Deep learning techniques can be used to improve the accuracy of predictions of our model.

Database can be integrated to store the user inputs and preferences to provide better recommendations. Also new movies need to be updated regularly so that user can also get recommendations for that.

These techniques can be integrated with similar apps like YouTube, Spotify etc. to generate recommendations.