Project Report

On

Content Based Recommendation System on Movies

Submitted by
Ch Joy Joshua Paul-R180244

Under the guidance of
T Sandeep Kumar Reddy

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge and Technologies(RGUKT), R.K.Valley,

Kadapa, Andra Pradesh.

Rajiv Gandhi University of Knowledge Techonolgies

RK Valley, Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is to certify that the project work titled "**Content Based Recommendation System on Movies**" is a bonafied project work submitted by CH. **JOY JOSHUA PAUL** in **COMPUTER SCIENCE AND ENGINEERING** in partial fulfillment of requirements for the award of degree of **Bachelor of Technology** for the year **2023-2024** carried out the work under the supervision.

INTERNAL GUIDE

T.SANDEEP KUMAR REDDY

HEAD OF THE DEPARTMENT

N. SATYANANDARAM

DECLARATION

I hereby declare that the project report entitled **"Content Based Recommendation System on Movies"** submitted to the Department of **COMPUTER SCIENCE & ENGINEERING** in partial fulfilment of requirements for the award of the degree of **BACHELOR OF TECHNOLOGY.** This project is the result of our own effort and that it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above.

ACKNOWLEDGEMENT

The satisfaction derived from the successful completion of any endeavor is magnified when we acknowledge the individuals whose unwavering support and guidance have been instrumental in achieving success.

I extend my deepest gratitude to our esteemed Director, **Mr. Kumara Swami Guptha**, whose commitment to fostering an excellent academic environment in our institution has been truly inspiring.

A heartfelt thank you is also extended to our Head of the Department, **Mr. N. Satyanandaram**, for his invaluable encouragement, overall guidance, and recognition of this project as a significant asset.

Special appreciation goes to our college guide, **Mr. T. Sandeep Kumar Reddy**, whose guidance, encouragement, cooperation, and kindness have been indispensable throughout the entire course and academic journey.

I would like to express my sincere thanks to all those who directly and indirectly contributed to the completion of this project. My profound gratitude extends to friends and family members for their unwavering encouragement.

INDEX

ABSTRACT:

User recommendation systems minimize transaction costs and decision-making quality and decisionmaking process. It has made the procedure for finding things easy that we need. A movie recommendation system can suggest a group of movies to users depending on their preferences or popularity of the movies. In this project, we are developing a movie recommendation system to reduce customer effort by proposing movies based on his or her preferences. The method used is content-based filtering. That instance, the suggestion algorithm is based on the genre or cast of the movie which that consumer may choose to watch. The system makes use of the **TMDB 5000 dataset**. The prerequisites of this project include 5 basic points: collecting data, preprocessing, modeling, converting it into a website, and deploying the website. First, the data is collected, and preprocessing is done on the data to convert the data according to our usage. Then, the data is converted into a model (machine learning model). Third, comes to the conversion of the model into a product or website and the last step is deploying the website. Despite the fact that various movie recommender systems have indeed been launched to date, we have come up with a project which makes the user more comfort.

INTRODUCTION

A **movie recommendation system** has become an important part of social life due to its contribution to the entertainment industry. Based on the interests of the users, this system suggests a set of movies. Even though there are many movie recommendation systems available, most of them are not capable of making efficient predictions. The objective of a movie recommendation system is to extract the necessary features to create a list of items for each user/individual and display the movies accordingly.There are three types of filtering techniques which can be used, i.e., **Collaborative filtering**, **Content-Based filtering**, and **Hybrid approach** - a combination of both. Collaborative filtering is dependent on a **user-item matrix** which identifies whether the user liked the item or not. It also makes use of ratings as an important criterion in model creation. Content-based filtering uses the movie information in the **dataset** and recommends movies based on each and every feature in the **feature set**.In this project, a **content-based movie recommendation** method is used. The feature sets, such as movie_id, title, genre, overview, cast, director, and keywords, that describe a movie are considered for recommending the top five movies.

PURPOSE

A **Content-Based Movie Recommendation System** functions as a technological asset for businesses, providing personalized suggestions through the analysis of content features and user preferences. It enables companies to recommend products or content in line with individual user interests, fostering heightened user engagement and satisfaction. By effectively harnessing this technology, businesses can optimize overall customer experiences, elevate conversion rates, and potentially amplify revenue streams. This data-driven approach empowers companies to strategically tailor content suggestions, contributing to enhanced user satisfaction and facilitating business growth in the competitive landscape.

SCOPE

Amidst the dynamic content consumption landscape, the **SCOPE** for a **CONTENT RECOMMENDATION SYSTEM** is vast. This project capitalizes on **TECHNOLOGY** to introduce a system that analyzes **USER PREFERENCES** and content attributes, offering a personalized content discovery experience. This user-friendly tool not only enhances **USER SATISFACTION** but also empowers businesses to optimize content consumption, retention, and overall engagement. The integration of such a system addresses the challenge of navigating extensive content libraries, providing users with tailored recommendations. The project's **SCOPE** lies in transforming content interaction, fostering user loyalty, and leveraging data-driven insights for improved content strategy in our digital era.

ANALYSIS AND DESIGN

Data Preparation

```
import numpy as np
import pandas as pd
#Reading the csv files using pandas
movies = pd.read_csv('tmdb_5000_movies.csv')
credits = pd.read_csv('tmdb_5000_credits.csv')
#Merging the data
movies =movies.merge(credits,on='title')
#It returns the first row of the data
movies.head(1)
#The Important Coloumns
#genres
#id
#keywords
#title
#overview
#cast
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
#This is the main dataset we are working
movies.head()
#find the missing data
movies.isnull().sum
#drop the missing data
movies.dropna(inplace = True)
```

```
#the no of duplicates in data
movies.duplicated().sum()
#To check the format
movies.iloc[0].genres
#Convert into a proper format
#'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"},
{"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]'
#['Action',Adventure,'Fantasy','Scifi']
#Conversion from String to List using ast module
import ast
def convert(obj):
    L = []
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L
#Store in the same coloumn
movies['genres']= movies['genres'].apply(convert)
movies.head()
movies['keywords'] = movies['keywords'].apply(convert)
movies.head()
#function to find the top 3 cast using a counter
def convert3(obj):
    L = []
    counter = 0
    for i in ast.literal_eval(obj):
        if counter != 3:
            L.append(i['name'])
            counter+=1
        else:
            break
    return L
#Assigning to the same coloumn
movies['cast']= movies['cast'].apply(convert3)
movies.head()
```

```
#To fetch the director
def fetch_director(obj):
    L = []
    for i in ast.literal_eval(obj):
        if i['job'] == 'Director':
            L.append(i['name'])
            break
    return L
#Assigning to the same coloumn
movies['crew'] = movies['crew'].apply(fetch_director)
movies.head()#to print the data
#Conversion of overview into the list
movies['overview'] = movies['overview'].apply(lambda x:x.split())
movies.head()
#Eliminating the spaces in the words so that they would look like one entity
movies['genres'] = movies['genres'].apply(lambda x:[i.replace(" ","") for i in x])
movies['keywords'] = movies['keywords'].apply(lambda x:[i.replace(" ","") for i in x])
movies['cast'] = movies['cast'].apply(lambda x:[i.replace(" ","") for i in x])
movies['crew'] = movies['crew'].apply(lambda x:[i.replace(" ","") for i in x])
movies.head()
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast']+ movies['crew']
movies.head()
#Creating a new dataframe of our desired coloumns
new_df = movies[['movie_id','title','tags']]
#Convert the List into String and reassign
#Just a warning
new_df['tags'] = new_df['tags'].apply(lambda x:" ".join(x))
new_df.head()
#Stemming after ln64 inorder to remove same words with s,ed, to find root word
import nltk
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()


#Creating a helper Function
```

```
def stem(text):
    y= []
    for i in text.split():
        y.append(ps.stem(i))
    return " ".join(y)
#applying stem for the newdataframe
new_df['tags'] = new_df['tags'].apply(stem)
#Converting into Lower
new_df['tags'] = new_df['tags'].apply(lambda x: x.lower())
new_df.head()
#TextVectorization,frequent words
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 5000,stop_words = 'english')
#conversion scipy matrix to numpy array
vectors = cv.fit_transform(new_df['tags']).toarray().shape
#assigning
vectors = cv.fit_transform(new_df['tags']).toarray()
#tag1
vectors[0]
#frequent words
cv.get_feature_names_out()
#legth for checking
len(cv.get_feature_names_out())
#distance between the vectors using Cosine -> angle measure
from sklearn.metrics.pairwise import cosine_similarity
#storing in a variable
similarity = cosine_similarity(vectors)
#Sort acc to the second attribute using enumerate and lambda
sorted(list(enumerate(similarity[0])),reverse = True,key = lambda x: x[1])[1:6]
#creating the recommend function
def recommend(movie):
    movie_index = new_df[new_df['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),reverse = True,key = lambda x: x[1])[1:6]
```

```
    for i in movies_list:
        print(new_df.iloc[i[0]].title)
 return
import pickle
pickle.dump(new_df, open('movies.pkl', 'wb'))
#array that contain names of movies
new_df['title'].values
pickle.dump(new_df.to_dict(),open('movie_dict.pkl','wb'))
pickle.dump(similarity,open('similarity.pkl','wb'))
```

Problem Statement:

A recommendation system helps in showing the best thing which can be a movie, song, book, etc to the user among the huge amount of data.

Here, the task is to find out the interests of specific users and recommend them accordingly. . The user enters a movie name present in the dataset as the input. The model extract features such as movie_id, title, genre, keywords, overview, cast, crew from the dataset and provides recommendation of the top five similar movies as the output.

Model configuration:

- **Feature Extraction:**
  - Features include movie_id, title, genre, keywords, overview, cast, and crew.
  - Text features are processed using techniques like Count Vectorizer and stemming.

- **Vectorization:**
  - Bag of Words technique is used for vectorization.
  - Count Vectorizer from sklearn.feature_extraction.text is employed.
  - Max_features is set to 5000, and stop words are removed from the 'tags' column.
- **Distance Calculation:**
  - Cosine Similarity is used as the distance measure.
  - The cosine_similarity() function is applied.
- **Main Function:**
  - User-defined function 'Recommend' is utilized.
  - This function identifies the top five most similar movies based on cosine similarity distances.
    - The function takes a movie title as input and outputs the recommended movies

- **API Extraction:**
  - Posters for each movie are extracted through an API from api.themoviedb.org.

<u>Model function:</u>

Dataset Linking Phase:

- Acquires data from the TMDB movie dataset, consisting of 'tmdb_5000_movies'
- and 'tmdb_5000_credits' files.
- Extracts relevant attributes like movie_id, title, genre
- And others like keyword, overview, cast, and crew.

Data Pre-Processing Phase:

- Removes null and duplicate values from the dataset.
- Converts attributes into lists of strings using ast.literal_eval().
- Applies user-defined functions to extract necessary content and create the 'tags' column.

Vectorization Phase:

- Utilizes Count Vectorizer for converting text features into vectors.

- Applies stemming using PorterStemmer.
- Removes stop words and sets max_features to 5000.

Distance Calculation Phase:

- Uses Cosine Similarity as the distance measure.
- Calculates the cosine similarity between vectors.

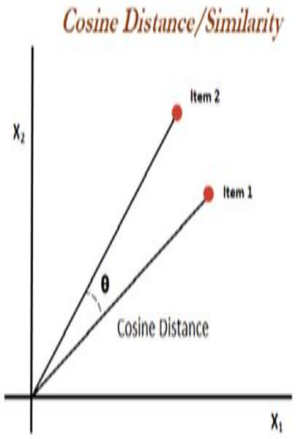
Main Function Phase:

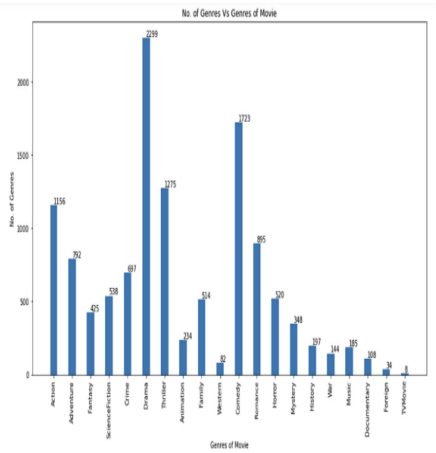
- Implements the 'Recommend' function, a user-defined function.
- Identifies the top five most similar movies based on cosine similarity distances .

API Extraction Phase:

- Extracts movie posters through an API from api.themoviedb.org.
- Links posters with movie indexes using the Streamlit library.
- Displays the top five recommended movies along with their posters.

Cosine Similarity Graph





.

Code(Streamlit)

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

def fetch_poster(movie_id):
 response = requests.get(
 'https://api.themoviedb.org/3/movie/{
}?api_key=40af966aad40c5929d094304ca5aafc6'.format(movie_id))
 data = response.json()
  # st.text(data)
 # Check if 'poster_path' key exists in the response
 if 'poster_path' in data:
 return "https://image.tmdb.org/t/p/w500" + data['poster_path']
 else:
 return "https://via.placeholder.com/150" # Placeholder image if poster_path is not available


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

for i in movies_list:
 movie_id = movies.iloc[i[0]].movie_id
 recommended_movies.append(movies.iloc[i[0]].title)
 recommended_movies_posters.append(fetch_poster(movie_id))
return recommended_movies, recommended_movies_posters
movies_dict = pickle.load(open('movie_dict.pkl', 'rb'))#To load data
movies = pd.DataFrame(movies_dict)
similarity = pickle.load(open('similarity.pkl', 'rb'))
# Streamlit app
```

```
st.title('Movie Recommender System')
selected_movie_name = st.selectbox(
 'Select a movie',
 movies['title'].values)
if st.button('Recommend'):
 names, posters = recommend(selected_movie_name)
 col1, col2, col3, col4, col5 = st.columns(5)
 with col1:
 st.text(names[0])
 st.image(posters[0])
 with col2:
 st.text(names[1])
 st.image(posters[1])
 with col3:
 st.text(names[2])
 st.image(posters[2])
 with col4:
 st.text(names[3])
 st.image(posters[3])
 with col5:
 st.text(names[4])
 st.image(posters[4]
```

PROJECT OUTPUT:

 # Project Output

Recommendation Results

The content-based movie recommendation system demonstrates effective movie recommendations

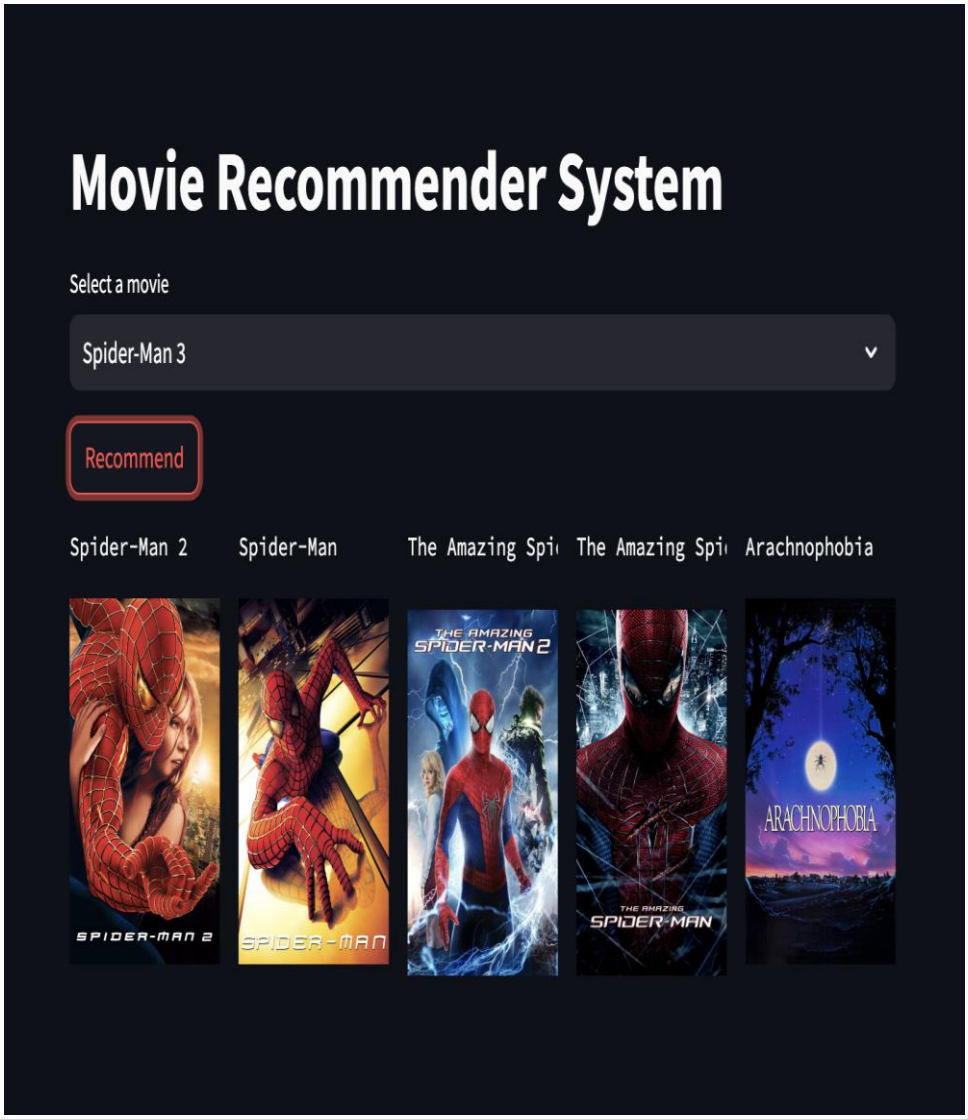
Below are sample recommendations for popular movies:

Movie: Spider-Man 3

1. - Spider-Man 2
2. - Spider –Man 3
3. - The Amazing Spider-Man 2

4. - The Amazing Spider-Man 1
5. - Arachnophobia

-



Movie: Superman

Recommended Movies:

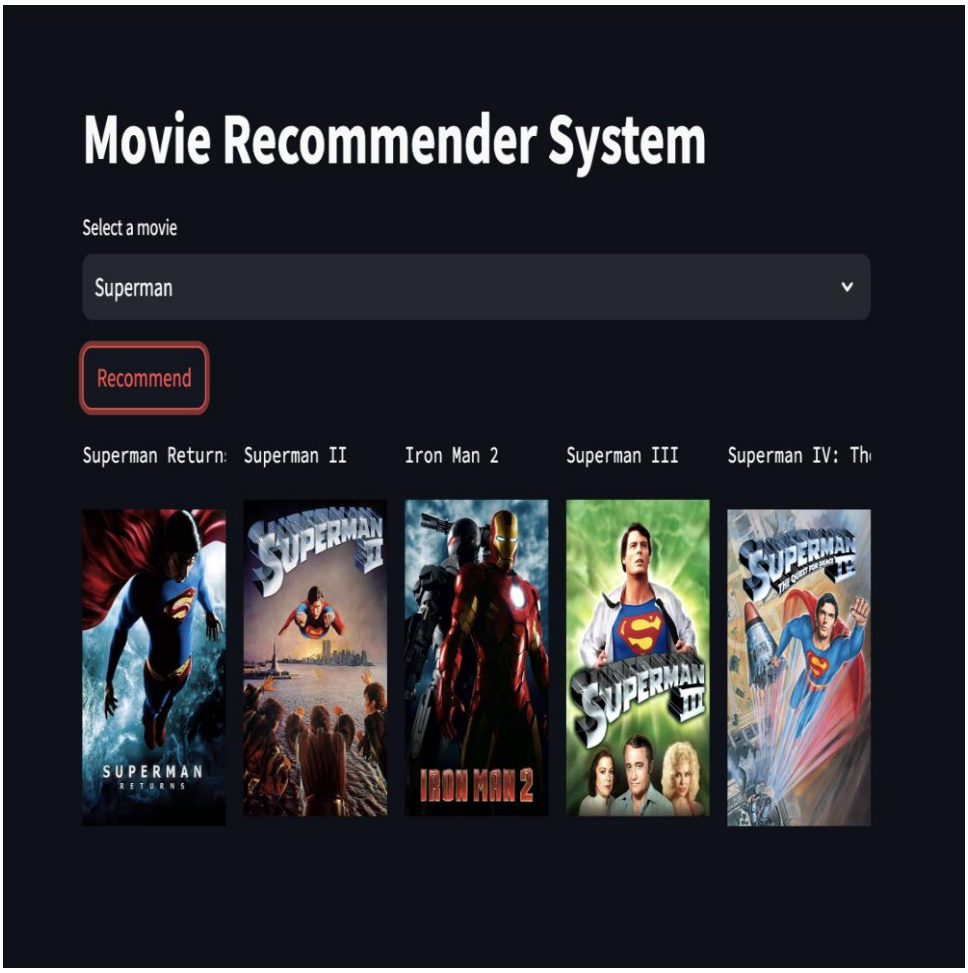
Superman Returns

Superman ll

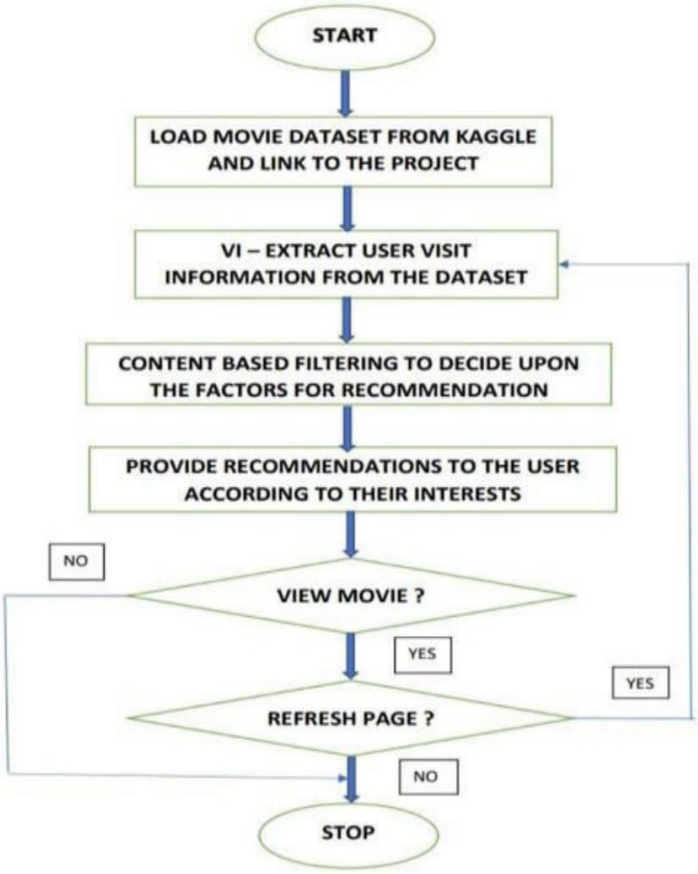
Superman lll

Superman lV

Iron Man 2



Architecture of Movie Recommendation System


START
LOAD MOVIE DATASET FROM KAGGLE AND LINK TO THE PROJECT
VI – EXTRACT USER VISIT INFORMATION FROM THE DATASET
CONTENT BASED FILTERING TO DECIDE UPON THE FACTORS FOR RECOMMENDATION
PROVIDE RECOMMENDATIONS TO THE USER ACCORDING TO THEIR INTERESTS
NO
VIEW MOVIE ?
YES
YES
REFRESH PAGE ?
NO
STOP

CONCLUSION AND FUTURE WORK

How to describe a movie, is the most important task because the more accurate a movie is described, the better results recommender system generates. So, from this perspective, an attempt is made for improvement of approach in content-based recommendation System. Firstly, a content-based recommender algorithm is used, which means there is no cold start problem. Then, the features in the recommender system are listed. Some of them are required for consideration and some are not. Then, the cosine similarity technique is used which is commonly used for calculating the similarities between the different movies. Based on the results, five different movies are recommended for a specific movie that has been selected. This model can be improved by introducing metrics to check the efficiency of the recommendations. A Hybrid model can be used which is the combination of collaborative and content-based filtering techniques to improve the efficiency of the overall model.

REFERENCES

- **https://numpy.org/doc/stable/**
- [pandas](https://pandas.pydata.org/)
- https://scikit-learn.org/stable/index.html
- https://streamlit.io/

**********THANKYOU**********