Information Retrieval and Web Analytics
IT3041

MOVIE RECOMMENDATION

GROUP NAME: SHADOW

Name	Registration Number
Maddumage P.W	IT21007538
Hewage R.P	IT21054686
Kiriella K.G.A.K	IT21035876

Git hub Link-
https://github.com/it21054686/IRWA-MOVIE-RECOMMENDATION-SYSTEM.git

Table of Contents

1.BACKGROUND

1.1. Movie Dataset – TMDB 5000 Movie Dataset

We use the TMDB movie data collection to create the movie recommendation system. When referring to the dataset, it is a thorough and rich collection of movie-related data that acts as an important resource for many data analysis and recommendation system applications. This dataset is a valuable resource for research, analytics, and developing movie recommendation apps because it includes a wide variety of movie-related data.

The main elements of this dataset are as follows :

- Movie information- The dataset offers information on a large number of films, such as titles, release dates, plot summaries, and screenshots of the movie posters. Each movie has one or more genre and keyword tags, which makes it easier to categories and propose movies depending on user interests.
- Information about the cast and crew: The dataset contains information about the actors, directors, producers, and other crew members connected to each film, allowing recommendations based on the participation of favorite actors or filmmakers.
- User-generated ratings and reviews are accessible, and they can be used to modify suggestions depending on a user's preferences and taste.

Data on a movie's popularity and box office receipts is also provided, making it possible to suggest popular or financially successful movies. The TMDb Movie Dataset is an imperative asset for developing movie recommendation systems, carrying out movie-related research, and improving the user experience when watching films. can be used to create creative applications that offer tailored movie suggestions, enhancing user engagement and happiness in the movie-going experience.

Then we start a project to recommend movies with the goal of building a solid and thorough database by assembling a diversified dataset that includes movie genres, ratings, and user reviews. We are putting in place modern collaborative filtering and recommendation algorithms that carefully examine user behavior and movie properties to make this possible. We will then be able to create highly customized movie recommendations based on each person's preferences. We're working carefully to create an easy interface that will enable users to easily obtain movie recommendations and actively participate by giving input on the suggested movies as part of our

dedication to user-friendliness. Our recommendation engine was created with a user-focused approach, taking user ratings, viewing history, and genre preferences into account to make sure the recommendations are appropriate for user. To validate the effectiveness of our recommendation machine, we are employing evaluation criteria like perfection, recall, and user satisfaction checks. This comprehensive approach ensures that our movie recommendation system not only delights users but also continually improves by learning from their feedback.

Dataset Link - https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

2.STEPS

In creating our movie recommendation algorithm, we carefully followed to a systematic process that included the following significant steps:

- **Data collection:** We started the study by collecting a large and diversified dataset that included details about the films, user reviews, and user ratings. We used a carefully selected dataset for this reason, making sure it matched the goals of our recommendation system.

- **Data preprocessing:** We cleaned and formatted the dataset while giving close attention to the accuracy of the data. This required dealing with any missing values and removing any records that were duplicates. Our recommendation algorithm is built on a user-item matrix that we further developed.

- **Collaborative Filtering Algorithm**: In order to generate movie recommendations, we used the well-known collaborative filtering algorithm. Both user-based and item-based collaborative filtering are used in our strategy. We calculated metrics for user similarity, such as cosine similarity, and suggested films based on what other users who are similar to you liked.

- **Movie Recommendation Generation**: Taking into account the collaborative filtering process we used, we systematically generated movie suggestions for user. This stage is essential for providing tailored movie

- **User Interface:** A user-friendly interface was thoughtfully crafted, offering a seamless experience for users. Through this interface, users can input their preferences, thereby initiating the recommendation process.

- **Feedback Mechanism**: Our recommendation system includes a valuable feedback mechanism. This allows users to rate and provide feedback on recommended movies.

- **Deployment:** The referral system is now work on the stream lit so it is available to users and we are ready to deploy it to serve a wider audience.

This systematic approach forms the backbone of our movie recommendation system, each step is carefully designed to deliver accurate, personalized and user-friendly movie recommendations.

4.TECHNOLOGIES

- Programming Languages-Python
- Data Preprocessing-Pandas
- Machine Learning and Recommendation Algorithms-Scikit-Learn, NumPy
- Deployment-Stream lit

5.BACKEND IMPLEMENTATION

1. Importing Libraries
2. Load and read the data set for environment
   (Movies.csv gives details of Movies likes genres, id, title overview etc.

```
#Import Libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import os
```

```
#Movie Details,including genres,id,voting,title etc Pandas library to read a CSV file
movies = pd.read_csv('/Users/randika/Desktop/Movie/data/tmdb_5000_movies.csv')
```

```
#listing of individuals or groups who contributed to the making of a film knowns as credits
#,Pandas library to read a  credit CSV file
credits = pd.read_csv('/Users/randika/Desktop/Movie/data/tmdb_5000_credits.csv')
```

3.To get clearly understand about dataset we lookup first few Rows
4.Looup No of Rows and Columns in Data frame
5.3rd and 4th step apply for credit csv also

```
#first two rows of a DataFrame
movies.head(2)
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_comp
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ing Film Partners
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Pictures", "id": :

```
#dimensions of a DataFrame.  The number of rows and columns in the DataFrame.
movies.shape
```

```
(4803, 20)
```

```
credits.head()
```

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

```
credits.head()
```

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

```
#dimensions of a DataFrame.  The number of rows and columns in the DataFrame.
credits.shape
```

```
(4803, 4)
```

6.Merge two Movies and credits data frame

```
#A merge operation between two pandas DataFrames, namely "movies" and "credits,"
#using the "title" column as the key for merging.
movies = movies.merge(credits,on='title')
```

```
movies.head(2)
```

click to scroll output; double click to hide

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_comp
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ing Film Partners
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Pictures", "id": :

2 rows × 23 columns

5.Keep important columns for recommendation

- **Movie_id** – In here in **collaborative filtering**, a user movie interaction data represented as a user movie matrix. This is foundation of recommendation system. Each row in this matrix contains information's how user interact with a specific movie likes voting, reviews. Movie IDs are used to uniquely identify each movie in this matrix, enabling the system to make user-specific recommendations based on historical interactions.

- Further clarifying selected columns, In here content-based recommendations **Content-based recommendations**, on the other hand, focus on the characteristics of movies themselves, such as genre, actors, director, and keywords.

- Movie IDs are used to link the content-based features and descriptions to the corresponding movies in the database. In a **hybrid recommendation** system, movie IDs serve as the bridge between these two different recommendation methods.

- **titles** - a fundamental part of user recommendation systems, particularly in **content-based and search-based** approaches. They are used for feature extraction, keyword analysis, user engagement, search functionality, categorization, and enhancing the overall user experience. Accurate and relevant movie titles are essential for providing users with personalized and appealing recommendations.

- **Overview**-Movie overviews can be analysed using natural language processing techniques to extract **keywords** and understand the movie's content, themes, and genre. This information is then used to recommend other movies with similar content. For example, if a user enjoys movies with "action and thriller" themes, a **content-based recommendation** system might suggest other films with similar keywords in their overviews

```
# Keep important columns for recommendation
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
```

```
movies.shape
```

```
(4809, 7)
```

5.Then count the total number of Null values
6.Drop them and again get sum of null values

```
#count the number of missing (null) values in each column of a DataFrame
movies.isnull().sum()
```

```
movie_id    0
title       0
overview    3
genres      0
keywords    0
cast        0
crew        0
dtype: int64
```

```
#remove rows with missing (NaN) values from the DataFrame called "movies" in place
movies.dropna(inplace=True)
```

```
#get a count of missing (NaN) values
movies.isnull().sum()
```

```
movie_id    0
title       0
overview    0
genres      0
keywords    0
cast        0
crew        0
dtype: int64
```

7.Remove Duplicate values

```
#count the number of duplicated rows in the "movies" DataFrame
movies.duplicated().sum()
```

```
0
```

```
# handle genres
#access the value of the "genres" column
movies.iloc[0]['genres']
```

```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name":
"Science Fiction"}]'
```

8.Access the relevant columns and convert these strings to list

```
0
```

```
# handle genres
#access the value of the "genres" column
movies.iloc[0]['genres']
```

```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name":
"Science Fiction"}]'
```

```
#converts this string into a list of names extracted from dictionaries within that list.
import ast #for converting str to list

def convert(text):
    L = []
    for i in ast.literal_eval(text):
        L.append(i['name'])
    return L
```

```
#convert each entry in the 'genres' column from a string representation of a list of dictionaries
#to a list of names.
movies['genres'] = movies['genres'].apply(convert)
```

```
movies.head()
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
			Following the death of		[{"id": 849, "name":	[{"cast_id": 2	

```
# handle keywords
#access a specific row and the 'keywords' column of a DataFrame named movies
#iloc is used for integer-location based indexing.
movies.iloc[0]['keywords']
```

```
'[{"id": 1463, "name": "culture clash"}, {"id": 2964, "name": "future"}, {"id": 3386, "name": "space war"}, {"id":
3388, "name": "space colony"}, {"id": 3679, "name": "society"}, {"id": 3801, "name": "space travel"}, {"id": 9685,
"name": "futuristic"}, {"id": 9840, "name": "romance"}, {"id": 9882, "name": "space"}, {"id": 9951, "name": "alie
n"}, {"id": 10148, "name": "tribe"}, {"id": 10158, "name": "alien planet"}, {"id": 10987, "name": "cgi"}, {"id": 11
399, "name": "marine"}, {"id": 13065, "name": "soldier"}, {"id": 14643, "name": "battle"}, {"id": 14720, "name": "l
ove affair"}, {"id": 165431, "name": "anti war"}, {"id": 193554, "name": "power relations"}, {"id": 206690, "name":
"mind and soul"}, {"id": 209714, "name": "3d"}]'
```

```
movies['keywords'] = movies['keywords'].apply(convert)
movies.head()
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

```
# handle overview (converting to list)

movies.iloc[0]['overview']
```

```
['In',
 'the',
 '22nd',
 'century,',
 'a',
 'paraplegic',
 'Marine',
 'is',
 'dispatched',
 'to',
 'the',
 'moon',
 'Pandora',
 'on',
 'a',
 'unique',
 'mission,',
 'but',
 'becomes',
 'torn',
 'between',
 'following',
 'orders',
 'and',
 'protecting',
 'an',
 'alien',
 'civilization.']
```

In [113]:
```
#string representation of a list of dictionaries into a list of actor names.
movies['cast'] = movies['cast'].apply(convert_cast)
movies.head()
```

```
#extract the name of the director from a list of crew members,
def fetch_director(text):
    L = []
    for i in ast.literal_eval(text):
        if i['job'] == 'Director':
            L.append(i['name'])
            break
    return L
```

```
#transform the 'crew' column by extracting the names of directors from the list of crew members for each movie.
movies['crew'] = movies['crew'].apply(fetch_director)
```

9.Removing spaces between words
10.Concatinate all the columns want for recommendations in two data frames and get into to single one

```
# now removing space like that
'Anna Kendrick'
'AnnaKendrick'

def remove_space(L):
    L1 = []
    for i in L:
        L1.append(i.replace(" ",""))
    return L1
```

```
movies['cast'] = movies['cast'].apply(remove_space)
movies['crew'] = movies['crew'].apply(remove_space)
movies['genres'] = movies['genres'].apply(remove_space)
movies['keywords'] = movies['keywords'].apply(remove_space)
```

```
movies.head()
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes]
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dccomics, crimefighter, terrorist, secretiden...	[ChristianBale, MichaelCaine, GaryOldman]	[ChristopherNolan]

```
# Concatinate all
#The contents of multiple columns into a new column named 'tags' in the movies DataFrame
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']
```

```
# first few rows of your DataFrame
movies.head()
```

	movie_id	title	overview	genres	keywords	cast	crew	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes]	[A, cryptic, message, from, Bond's, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dccomics, crimefighter, terrorist, secretiden...	[ChristianBale, MichaelCaine, GaryOldman]	[ChristopherNolan]	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Action, Adventure, ScienceFiction]	[basedonnovel, mars, medallion, spacetravel, p...	[TaylorKitsch, LynnCollins, SamanthaMorton]	[AndrewStanton]	[John, Carter, is, a, war-weary,, former, mili...

```
# droping those extra columns
new_df = movies[['movie_id','title','tags']]
```

```
new_df.head()
```

	movie_id	title	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...

```
# Converting list to str
new_df['tags'] = new_df['tags'].apply(lambda x: " ".join(x))
new_df.head()
```

11.Apply Stemming algorithm

```
#stemming algorithm in NLP
import nltk
from nltk.stem import PorterStemmer
```

```
#created an instance of the Porter Stemmer
ps = PorterStemmer()
```

```
#splits the input text into individual words, applies the Porter Stemmer to each word, and then joins the stem

def stems(text):
    T = []

    for i in text.split():
        T.append(ps.stem(i))

    return " ".join(T)
```

```
#stemming the words within the 'tags' column and replacing the original text with the stemmed version
new_df['tags'] = new_df['tags'].apply(stems)
```

```
/var/folders/py/l9nrq9zd6js76qm5j_bpkcqr0000gn/T/ipykernel_4227/3973021881.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retur
ng-a-view-versus-a-copy
  new_df['tags'] = new_df['tags'].apply(stems)
```

```
new_df.iloc[0]['tags']
```

```
'in the 22nd century, a parapleg marin is dispatch to the moon pandora on a uniqu mission, but becom torn between
ollow order and protect an alien civilization. action adventur fantasi sciencefict cultureclash futur spacewar sp
ecoloni societi spacetravel futurist romanc space alien tribe alienplanet cgi marin soldier battl loveaffair anti
r powerrel mindandsoul 3d samworthington zoesaldana sigourneyweav jamescameron'
```

```
#processing to convert text data into numerical format
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
```

```
#transform the 'tags' column of the DataFrame new_df into a matrix of word counts and then converting it to a Num
vector = cv.fit_transform(new_df['tags']).toarray()
```

```
#word count representation of the first document
vector[0]
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
vector.shape
```

```
(4806, 5000)
```

12.Vectorizing and cosine similarities

```
# Get feature names
#words that correspond to the columns in the matrix produced by the vectorization process.
feature_names = cv.get_feature_names_out()
```

```
#number of unique words  in the vocabulary used for vectorization by the CountVectorize
len(cv.get_feature_names_out())
```

```
5000
```

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
#the cosine similarity between vectors
similarity = cosine_similarity(vector)
```

```
similarity.shape
```

```
(4806, 4806)
```

```
#find the index of the first occurrence of a specific value in the 'title' column of the DataFrame new_df
new_df[new_df['title'] == 'The Lego Movie'].index[0]
```

```
744
```

13.Build Recommendation algorithm

```
def recommend(movie):
    #This line finds the index of the movie in the DataFrame new_df that matches the given movie title.
    #It uses the 'title' column to locate the movie.
    index = new_df[new_df['title'] == movie].index[0]
    #his line calculates the cosine similarity between the selected movie (indexed by index) and all other movies i
    #It sorts the movies by their similarity in descending order.
    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])
    #the top 5 movies in distances (excluding the first element, which is the movie itself),
    #indicating the most similar movies.
    for i in distances[1:6]:
    #retrieves the title of that movie from the DataFrame new_df.
        print(new_df.iloc[i[0]].title)
```

```
recommend('Spider-Man 2')
```

```
Spider-Man 3
Spider-Man
The Amazing Spider-Man
Iron Man 2
Superman
```

14.Build model save to the file for further UI Implement

```
#save and load Python objects to/from files
import pickle
```

```
import os#working with the file system.
import pickle#The process of converting a Python object into a format that can be easily saved to a file

# Define the directory path where you want to save the files
directory = '/Users/randika/Desktop/Movie/artifacts'

# Create the directory if it doesn't exist
if not os.path.exists(directory):
    os.makedirs(directory)

# Now you can save the pickle files
#opens a file for writing in binary mode ('wb').
#It constructs the full file path by joining the directory path specified in the directory variable with
#the filename 'movie_list.pkl'. The resulting file will be created in the specified directory.
pickle.dump(new_df, open(os.path.join(directory, 'movie_list.pkl'), 'wb'))
pickle.dump(similarity, open(os.path.join(directory, 'similarity.pkl'), 'wb'))
```

6.UI Implementation
6.1. Background

The artifact files are used to obtain the recommendation algorithm to work in this system all the recommendations are based on the data that have been preprocessed in the file.
This is a simple user interface to navigate for the user to have the interaction and the system to give the recommendations according to the user.
For this process, the system is configured to a single user and the extensions can be done in the future by using a DB to fetch the data from the User DB as well. This is the UI when a single user gets into the site and for more understanding this code will be shown,

url = "https://api.themoviedb.org/3/movie/{}?api_key=e29500185edb84e31a6780b0c7c0f7dc&language=en-US&page=1"

In the above code, **api_key=e29500185edb84e31a6780b0c7c0f7dc** is used to verify that the user's login key for the system. According to this user, the recommendations will be given.
All the URLs used for this system is got by the TMDB movie API website which provides various kind of API to work with as a developer.

A key function in the system is retrieving movie posters using the TMDB API. The code snippet below demonstrates this process:

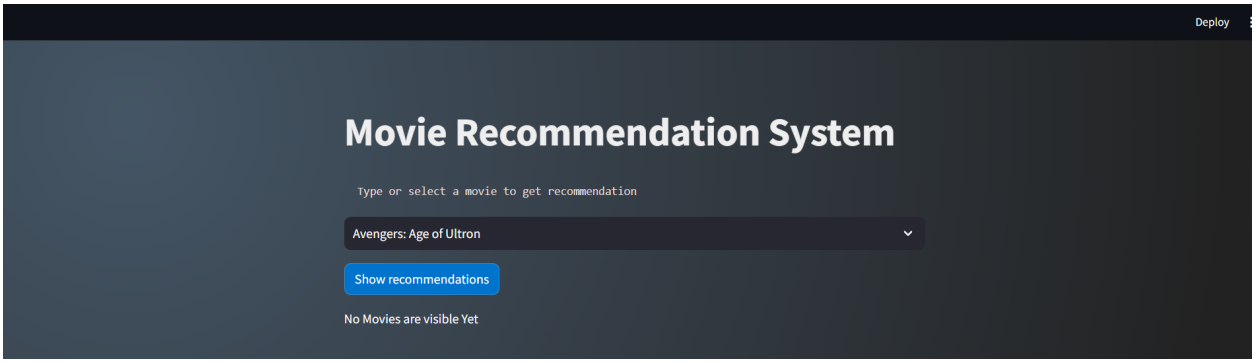
```
def fetch_poster(movie_id):
    url = "https://api.themoviedb.org/3/movie/{}?api_key=e29500185edb84e31a6780b0c7c0f7dc&language=en-US&page=1".format(
        movie_id)
    data = requests.get(url)
    data = data.json()
    poster_path = data['poster_path']  # this will only a part from the path
    # to define the full path
    full_path = "https://image.tmdb.org/t/p/w500/" + poster_path
    return full_path
```

This code retrieves the poster image, ensuring a seamless user experience by providing a full path to the image.

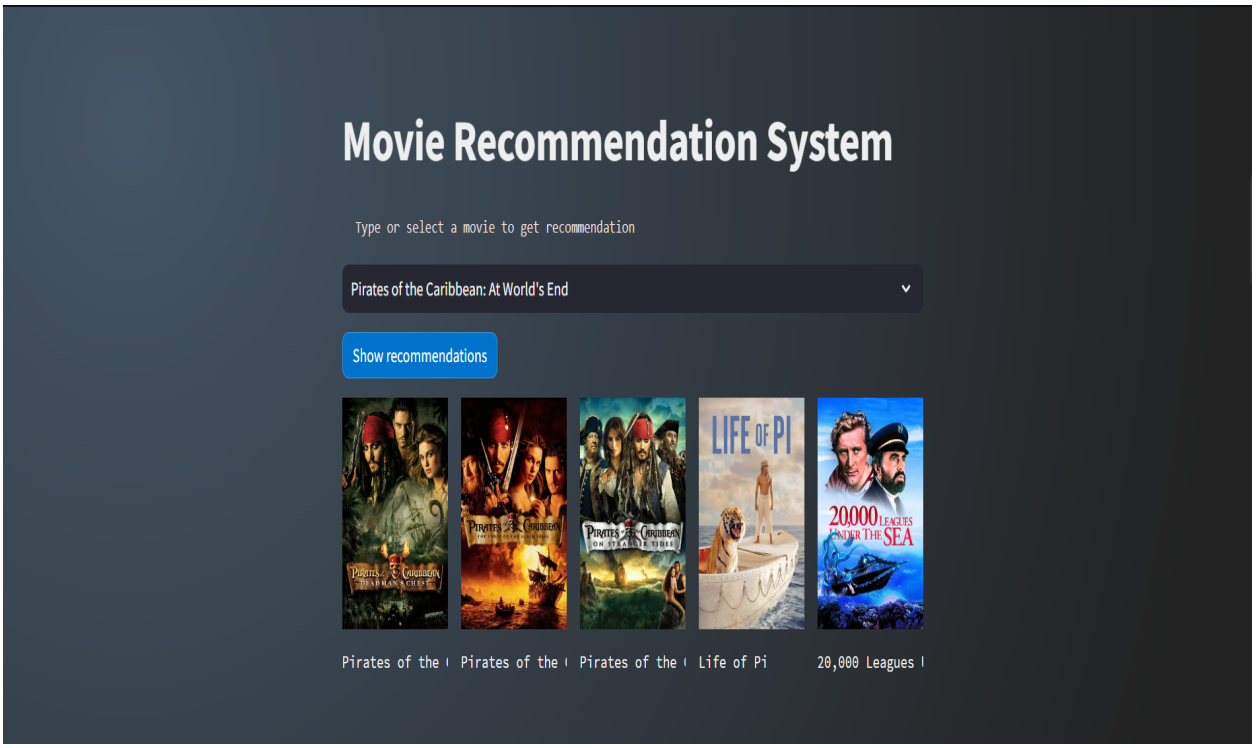
After receiving recommendations, users can rate the suggested movies and provide feedback. This feedback can be incorporated into the system to enhance the quality of recommendations further. A message is displayed to confirm the successful addition of reviews.

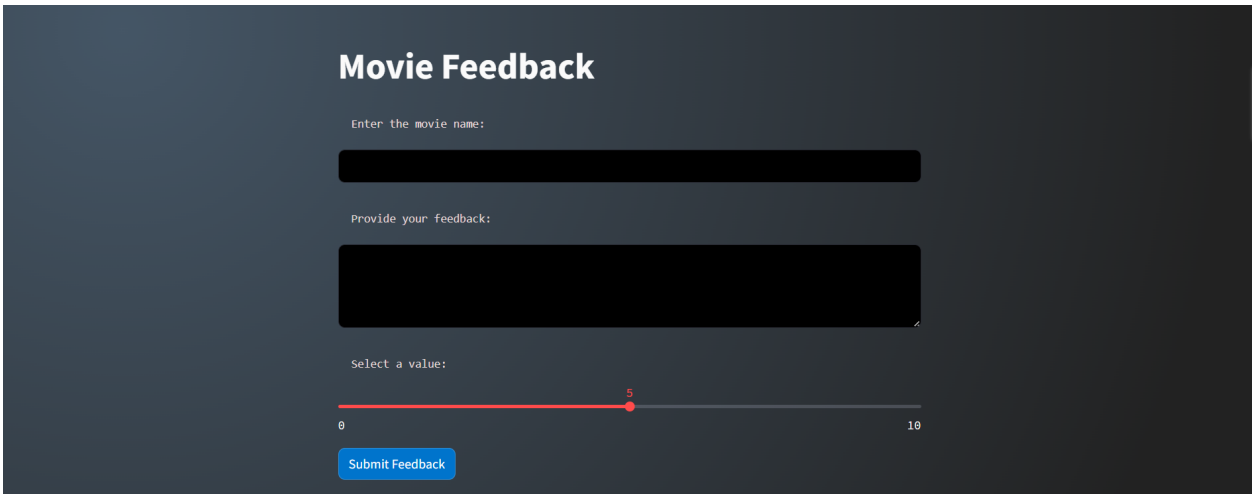
6.2. UI Deployement

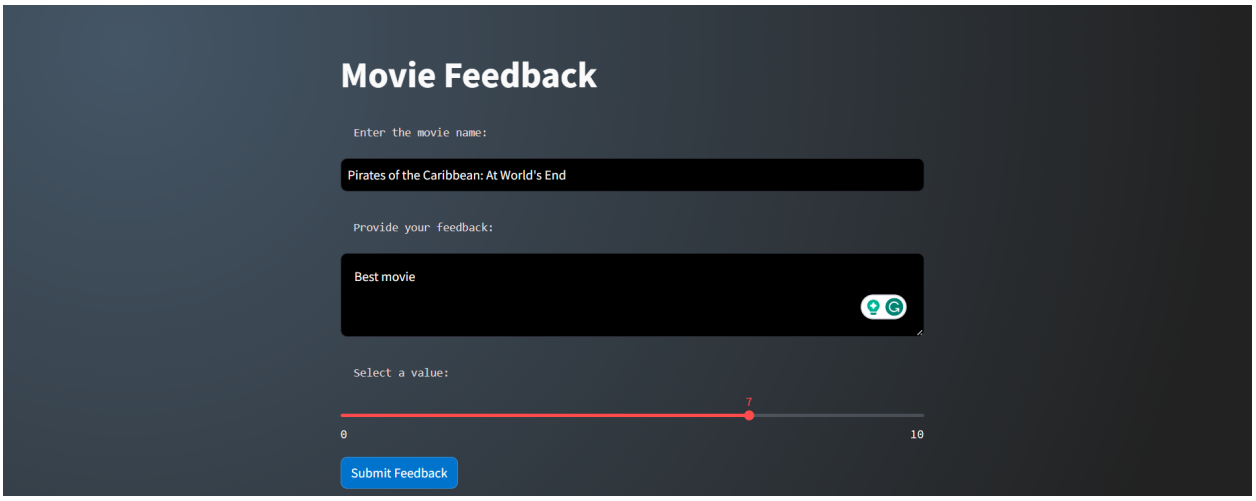
The user interface presents a clean and user-friendly input section to users. Here, they can easily express their movie preferences by choosing genres, showing their favorite actors or directors, and even rating the movies they enjoyed. The user interface allows users to simply communicate their movie tastes and thus start the recommendation process. Get recommendations: Once users have made their selections, they will immediately receive a curated list of movie recommendations. These recommendations are intelligently generated using our implemented collaborative filtering algorithms that consider both user behavior and movie attributes. Recommendations are presented in a visually appealing and organized manner, making it easy for users to find new movies that suit their tastes. Integrating user feedback: Our user interface also seamlessly integrates feedback. This feature allows users to rate recommended movies and provide feedback on their viewing experience. This feedback mechanism not only improves the learning ability of the system, but is also a valuable tool for further improvement of the system. Overall, our user-centric approach to user interface design ensures that users can easily enter their preferences, receive personalized recommendations and actively interact with the system. The user interface is a key part of providing a satisfying and user-friendly movie recommendation experience.

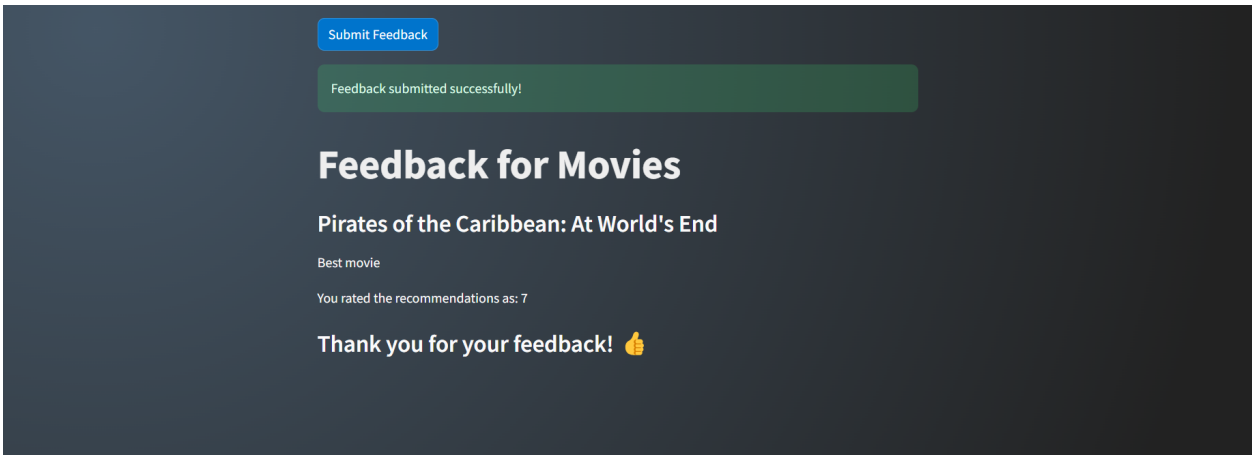

Deploy
Movie Recommendation System
Type or select a movie to get recommendation
Avengers: Age of Ultron
Show recommendations
No Movies are visible Yet


Deploy
Movie Recommendation System
Type or select a movie to get recommendation
Pirates of the Caribbean: At World's End
Avatar
Pirates of the Caribbean: At World's End
Spectre
The Dark Knight Rises
John Carter
Spider-Man 3
Tangled


Movie Recommendation System
Type or select a movie to get recommendation
Pirates of the Caribbean: At World's End
Show recommendations
Pirates of the
Pirates of the
Pirates of the
Life of Pi
20,000 Leagues


Movie Feedback
Enter the movie name:
Provide your feedback:
Select a value:
5
0
10
Submit Feedback





7.CONCLUSION

In conclusion, our movie recommendation system was designed and implemented with a strong focus on the accuracy and relevance of the recommendations it provides. Both user-based and object-based collaborative filtering algorithms have been diligently used and optimized to create highly personalized movie recommendations.

In addition, the recommendation engine not only increased user engagement, but also promoted active interaction with the streaming platform. User feedback, ratings and reviews have played a key role in the continuous learning and improvement of the system, ensuring it stays in line with changing user preferences. The user interface is thoughtfully designed to be intuitive, ensuring that users can easily enter their preferences, view recommendations and provide feedback. This

approach made the recommendation system easy to use and use, improving the overall experience of watching movies.

In conclusion, our movie recommendation system has proven its ability to provide accurate and relevant recommendations, increase user engagement, efficiently implement recommendation algorithms, and provide an intuitive interface for smooth communication. This project demonstrates the effectiveness of recommendation systems to improve user experience and user satisfaction in the cinema world.has context menu

8.RESPONSIBILITIES

Name	Registration Number	Responsibilities
Maddumage P.W.	IT21007538	• Analyze dataset • Deployment • Documentation
Hewage R.P.	IT21054686	• Preprocessing Data • UI Implementation • Documentation
Kiriella K.G.A.K.	IT21035876	• Visualize Movie Dataset • UI Deployment • Documentation

9.REFERENCES

- https://www.kaggle.com/
- https://www.geeksforgeeks.org/
- https://discuss.streamlit.io/t/css-styling