Project Name: Movie Recommendation System: Enhancing Personalized and Tailored Movie Suggestions

Objective : The main objective of a movie recommendation system is to provide personalized and relevant movie suggestions to users based on their preferences and historical behavior. The system aims to enhance user experience and satisfaction by offering tailored recommendations that align with their individual tastes, thereby assisting users in discovering new movies that they are likely to enjoy. The primary goals of a movie recommendation system include:

Personalization: Delivering movie recommendations that are customized to each user's unique preferences, taking into account their past behavior, ratings, and interactions with movies.

Accuracy: Providing accurate and reliable recommendations that accurately reflect the user's preferences and increase the likelihood of user satisfaction with the suggested movies.

Diversity: Ensuring that the recommended movies cover a diverse range of genres, directors, actors, and other relevant factors, to offer users a variety of options and broaden their movie-watching horizons.

Serendipity: Introducing users to movies they may not have discovered otherwise, by suggesting relevant yet unexpected choices that go beyond their usual preferences, thereby enhancing their movie discovery experience.

User Engagement: Encouraging users to explore the platform, spend more time watching movies, and actively participate by rating, reviewing, and sharing their movie experiences.

By achieving these objectives, a movie recommendation system aims to enhance user satisfaction, increase user engagement and retention, and drive overall user enjoyment of the movie-watching experience.

There are 2 kinds of recommendation systems

1. Content-based recommendation system : Once your interest or past activity was collected. In simple, after you watched a video or a movie them the recommendation system finds other videos that are similar to the one you've already watched and recommend a similar video back to you.
2. Collaborative Filtering System : This is the recommendation system made popular by Netflix. It's basically a peer to peer recommendations. If a person 1 watched a movie and Person 2 also watches the movie then the movie watched by person 1 later will be suggested to the person 2 vice-versa. This matches the people with similar interests and recommends to each other.

In this project, Google Colab was utilized as the primary platform for developing a recommendation system. Python libraries, including Pandas and NumPy, were employed for data manipulation and analysis. Additionally, machine learning libraries such as scikit-learn were leveraged to facilitate the implementation of various algorithms, with cosine similarity employed to determine the similarity between two movies during the recommendation process. Moreover, TfidfVectorizer, a machine learning tool, was employed for feature extraction and transformation

Import Library

```
import pandas as pd
```

```
import numpy as np
```

Import Dataset

```
df = pd.read_csv(r'https://github.com/YBIFoundation/Dataset/raw/main/Movies%20Recommendati
```

Describe Data

```
df.head()
```

ie_Revenue	Movie_Runtime	Movie_Vote	...	Movie_Homepage	Movie_Keywor
4300000	98.0	6.5	...	NaN	hotel new yea eve witch b hotel roc
775398007	121.0	8.1	...	http://www.starwars.com/films/star-wars-episod...	android gala hermit death st lightsab

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4760 entries, 0 to 4759
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Movie_ID                4760 non-null   int64
 1   Movie_Title             4760 non-null   object
 2   Movie_Genre             4760 non-null   object
 3   Movie_Language          4760 non-null   object
 4   Movie_Budget            4760 non-null   int64
 5   Movie_Popularity        4760 non-null   float64
 6   Movie_Release_Date      4760 non-null   object
 7   Movie_Revenue           4760 non-null   int64
 8   Movie_Runtime           4758 non-null   float64
 9   Movie_Vote              4760 non-null   float64
 10  Movie_Vote_Count        4760 non-null   int64
 11  Movie_Homepage          1699 non-null   object
 12  Movie_Keywords          4373 non-null   object
 13  Movie_Overview          4757 non-null   object
 14  Movie_Production_House  4760 non-null   object
 15  Movie_Production_Country  4760 non-null   object
 16  Movie_Spoken_Language   4760 non-null   object
 17  Movie_Tagline           3942 non-null   object
 18  Movie_Cast              4733 non-null   object
 19  Movie_Crew              4760 non-null   object
 20  Movie_Director          4738 non-null   object
dtypes: float64(3), int64(4), object(14)
memory usage: 781.1+ KB
```

```
df.shape
```

```
(4760, 21)
```

```
df.columns
```

```
Index(['Movie_ID', 'Movie_Title', 'Movie_Genre', 'Movie_Language',
       'Movie_Budget', 'Movie_Popularity', 'Movie_Release_Date',
       'Movie_Revenue', 'Movie_Runtime', 'Movie_Vote', 'Movie_Vote_Count',
       'Movie_Homepage', 'Movie_Keywords', 'Movie_Overview',
       'Movie_Production_House', 'Movie_Production_Country',
       'Movie_Spoken_Language', 'Movie_Tagline', 'Movie_Cast', 'Movie_Crew',
```

```
       'Movie_Director'],
      dtype='object')
```

Get Feature Selection

```
df_features = df[['Movie_Genre','Movie_Keywords','Movie_Tagline','Movie_Cast','Movie_Direc
```

```
df_features.shape
```

```
(4760, 5)
```

```
df_features
```

	Movie_Genre	Movie_Keywords	Movie_Tagline	Movie_Cas
0	Crime Comedy	hotel new year's eve witch bet hotel room	Twelve outrageous guests. Four scandalous requ...	Tim Roth Antoni Banderas Jennife Beals Madon.
1	Adventure Action Science Fiction	android galaxy hermit death star lightsaber	A long time ago in a galaxy far, far away...	Mark Hamill Harriso Ford Carrie Fishe Peter .
2	Animation Family	father son relationship harbor underwater fish...	There are 3.7 trillion fish in the ocean, they...	Albert Brooks Elle DeGeneres Alexande Gould .
3	Comedy Drama Romance	vietnam veteran hippie mentally disabled runni...	The world will never be the same, once you've ...	Tom Hanks Robi Wright Gary Sinis Mykelti Wil.
4	Drama	male nudity female nudity adultery midlife cri...	Look closer.	Kevin Spacey Annett Bening Thora Birc Wes Be.
...	...	...	...	.
4755	Horror		The hot spot where Satan's waitin'.	Lisa Hart Carrc Michael Des Barre Paul Drak.

Get Feature Selection

```
X = df_features['Movie_Genre'] + ' ' + df_features['Movie_Tagline'] + ' ' + df_features['M
```

```
X
```

```
0       Crime Comedy Twelve outrageous guests. Four sc...
1       Adventure Action Science Fiction A long time a...
2       Animation Family There are 3.7 trillion fish i...
3       Comedy Drama Romance The world will never be t...
4       Drama Look closer. Kevin Spacey Annette Bening...
                              ...
4755    Horror The hot spot where Satan's waitin'. Lis...
```

```
4756    Comedy Family Drama It’s better to stand out t...
4757    Thriller Drama She never knew it could happen ...
4758                                              Family
4759       Documentary  Tony Oppedisano Simon Napier-Bell
Length: 4760, dtype: object
```

```
X.shape
```

```
(4760,)
```

Get Feature Text Conversion to Tokens

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf = TfidfVectorizer()
```

```
X = tfidf.fit_transform(X)
```

```
X.shape
```

```
(4760, 14522)
```

```
print(X)
```

```
  (0, 536)      0.18650035534391993
  (0, 425)      0.163573988034563
  (0, 12942)    0.1673343571078569
  (0, 8260)     0.16243380336877533
  (0, 8073)     0.1895214821424654
  (0, 1162)     0.19294655321876156
  (0, 6582)     0.11189186354263483
  (0, 1012)     0.15832102537496356
  (0, 635)      0.15317383577872903
  (0, 11059)    0.163573988034563
  (0, 12888)    0.11963620447432914
  (0, 7696)     0.11052960016040947
  (0, 9522)     0.0817193034473646
  (0, 4238)     0.1813531657476854
  (0, 14325)    0.15477747865376898
  (0, 9299)     0.13177600168161205
  (0, 14080)    0.20157704304573848
  (0, 4683)     0.098789221275497
  (0, 14047)    0.12205988813713547
  (0, 6652)     0.17174230442934296
  (0, 12768)    0.11431048024187192
  (0, 9596)     0.1084085633118925
  (0, 3252)     0.1387089301293658
  (0, 4594)     0.13827045908436408
  (0, 5989)     0.2263333132749165
  :     :
  (4757, 1894)  0.30757798308075385
  (4757, 11986) 0.30757798308075385
  (4757, 699)   0.29336587999264124
```

```
  (4757, 4852)  0.258986481674795
  (4757, 12273) 0.258986481674795
  (4757, 990)   0.2017079859012847
  (4757, 2910)  0.2126571431460345
  (4757, 5624)  0.23256575558502307
  (4757, 11699) 0.17574280735214792
  (4757, 9341)  0.19618287718072355
  (4757, 7169)  0.2511649969371266
  (4757, 5877)  0.1824610108818817
  (4757, 7089)  0.1866512308804554
  (4757, 12845) 0.08173029845757258
  (4757, 12916) 0.10610980394127228
  (4757, 6421)  0.12266242201117997
  (4757, 9297)  0.15532169957924374
  (4757, 3766)  0.060644484319855856
  (4758, 4374)  1.0
  (4759, 9616)  0.5572028145970158
  (4759, 9188)  0.5131890638764602
  (4759, 3647)  0.30216883908425024
  (4759, 11875) 0.329655024339566
  (4759, 12960) 0.32296479807592227
  (4759, 1247)  0.34910930228325543
```

Get Similarity Score using Cosine Similarity

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

```
Similarity_Score = cosine_similarity(X)
```

```
Similarity_Score
```

```
array([[1.        , 0.02078195, 0.05449681, ..., 0.        , 0.        ,
        0.        ],
       [0.02078195, 1.        , 0.01356195, ..., 0.        , 0.        ,
        0.        ],
       [0.05449681, 0.01356195, 1.        , ..., 0.        , 0.10459462,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.10459462, ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

```
Similarity_Score.shape
```

```
(4760, 4760)
```

Get Movie Name as Input from User and Validate for Closest Spelling

```
Favourite_Movie_Name = input('Enter your favourite movie name: ' )
```

```
Enter your favourite movie name: Avatar
```

```
All_Movies_Title_List = df['Movie_Title'].tolist()
```

```
import difflib
```

```
Movie_Recommendation = difflib.get_close_matches(Favourite_Movie_Name, All_Movies_Title_Li
print(Movie_Recommendation)
```

```
['Avatar']
```

```
Close_Match  = Movie_Recommendation[0]
print(Close_Match)
```

```
Avatar
```

```
Index_of_Close_Match_Movie = df[df.Movie_Title == Close_Match]['Movie_ID'].values[0]
print(Index_of_Close_Match_Movie)
```

```
2692
```

```
Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Close_Match_Movie]))
print(Recommendation_Score)
```

```
[(0, 0.015132661710742871), (1, 0.0), (2, 0.0), (3, 0.013160861066208182), (4, 0.0038
```

```
len(Recommendation_Score)
```

```
4760
```

Get All Movies Based on Recommendation Score wrt Favourite Movie

```
Sorted_Similar_Movies = sorted(Recommendation_Score, key = lambda x:x[1], reverse = True)
print(Sorted_Similar_Movies)
```

```
[(2692, 1.0000000000000002), (62, 0.1411679395649748), (3276, 0.13918763868476203), (
```

Prediction: The prediction process of the recommendation system involves analyzing user preferences and leveraging machine learning algorithms to generate accurate movie recommendations. By utilizing techniques such as cosine similarity and TfidfVectorizer, the system calculates the similarity between movies based on their features and user preferences. It then ranks the movies according to their similarity scores and presents the top recommendations to the user. The prediction phase takes into account the user's input, historical data, and the underlying algorithms to generate personalized and relevant movie suggestions. This ensures that the recommendations are tailored to the specific tastes and preferences of each user, enhancing their movie-watching experience by providing them with a curated list of movies they are likely to enjoy.

```
print('Top 30 Movies Suggested for You : \n')
i = 1
for movie in Sorted_Similar_Movies:
  index = movie[0]
  title_from_index = df[df.index==index]['Movie_Title'].values[0]
  if (i<31):
    print(i, '.',title_from_index)
    i+=1
```

```
Top 30 Movies Suggested for You :

1 . Niagara
2 . Brokeback Mountain
3 . Caravans
4 . Night of the Living Dead
5 . Mad Hot Ballroom
6 . Some Like It Hot
7 . The Kentucky Fried Movie
8 . The Misfits
9 . Superman III
10 . Tora! Tora! Tora!
11 . To Kill a Mockingbird
12 . Beyond the Black Rainbow
13 . Duel in the Sun
14 . The Change-Up
15 . Man of Steel
16 . Running with Scissors
17 . All That Jazz
18 . Butch Cassidy and the Sundance Kid
19 . The Boy Next Door
20 . The Odd Life of Timothy Green
21 . The Lazarus Effect
22 . Mad Max 2: The Road Warrior
23 . Dallas Buyers Club
24 . The Dark Knight Rises
25 . Source Code
26 . Camping Sauvage
27 . Twister
28 . Mission: Impossible
29 . I Spit on Your Grave
30 . The Longest Yard
```

Top 10 Movie Recommendation System

```
Movie_Name = input('Enter your favourite movie name : ')
list_of_all_titles = df['Movie_Title'].tolist()
Find_Close_Match = difflib.get_close_matches(Movie_Name, list_of_all_titles)
Close_Match = Find_Close_Match[0]
```

```
Index_of_Movie = df[df.Movie_Title == Close_Match]['Movie_ID'].values[0]
Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Movie]))
sorted_similar_movies = sorted(Recommendation_Score, key = lambda x:x[1], reverse = True)
print('Top 10 Movies suggested for you : \n')
i = 1
for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = df[df.Movie_ID==index]['Movie_Title'].values
  if (i<11):
    print(i, '.',title_from_index)
    i+=1
```

```
Enter your favourite movie name : Avatar
Top 10 Movies suggested for you :

1 . ['Avatar']
2 . ['Donnie Darko']
3 . ['The Girl on the Train']
4 . ['Freaky Friday']
5 . ['Repo! The Genetic Opera']
6 . ['The Godfather']
7 . ['New Nightmare']
8 . ['Rollerball']
9 . ['A Prairie Home Companion']
10 . ['The Merchant of Venice']
```

Explanation or Final Outcome: The project aimed to develop a recommendation system using Google Colab, Python libraries such as Pandas and NumPy, and machine learning tools like scikit-learn. The final outcome of the project was a resounding success, as it enabled users to receive movie recommendations tailored to their preferences. By employing cosine similarity and TfidfVectorizer, the system accurately identified and suggested movies that closely matched the user's choices. With this achievement, users can now effortlessly discover and explore movies that align with their interests, enhancing their overall movie-watching experience. The project's goal of delivering personalized and relevant movie recommendations has been accomplished, providing users with an enhanced entertainment selection.

6s completed at 9:34 PM