Movie Recommendation System

Objective :

To develop a personalized Movie Recommendation System that leverages machine learning algorithms and user preferences to suggest movies tailored to individual tastes. The primary goal is to enhance user engagement and satisfaction by providing relevant movie recommendations based on historical viewing behavior and movie attributes. The system will strive to achieve a high level of accuracy and user-friendliness, making it a valuable tool for movie enthusiasts seeking personalized movie suggestions.

Data Source :

This Data source is taken from github of ybi repositories. It's an a csv file.

Import Library :

```
import pandas as pd
import numpy as np
```

Import DataSet :

```
df = pd.read_csv(r"https://raw.githubusercontent.com/Naveen1131/MovieRecommendation/main/Movies%20Recommendation.csv")
```

```
df.head()
```

	Movie_ID	Movie_Title	Movie_Genre	Movie_Language	Movie_Budget	Movie_Popu
0	1	Four Rooms	Crime Comedy	en	4000000	22.8
1	2	Star Wars	Adventure Action Science Fiction	en	11000000	126.3
2	3	Finding Nemo	Animation Family	en	94000000	85.6
3	4	Forrest Gump	Comedy Drama Romance	en	55000000	138.1
4	5	American Beauty	Drama	en	15000000	80.8

5 rows × 21 columns

```
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')
```

Feature Selection :

```
df_features = df[['Movie_Title','Movie_Budget','Movie_Production_Country','Movie_Cast','Movie_Director']].fillna('')
```

Selected five existing features from recommended movies. It can vary for each movie.

```
df_features.shape
```

```
(4760, 5)
```

```
df_features
```

	Movie_Title	Movie_Budget	Movie_Production_Country	Movie_Cast	Movie_Di
0	Four Rooms	4000000	[{"iso_3166_1": "US", "name": "United States o...	Tim Roth Antonio Banderas Jennifer Beals Madon...	Allison
1	Star Wars	11000000	[{"iso_3166_1": "US", "name": "United States o...	Mark Hamill Harrison Ford Carrie Fisher Peter ...	George

```
import numpy as np

# Assuming df_features contains your data
X = np.array(df_features[['Movie_Title', 'Movie_Budget', 'Movie_Production_Country', 'Movie_Cast', 'Movie_Director']])
shape = X.shape
print(shape)
```

```
(4760, 5)
```

```
X.shape
```

```
(4760, 5)
```

Get Feature Text Conversion to Tokens

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# Sample text data
df = pd.read_csv(r"https://raw.githubusercontent.com/Naveen1131/MovieRecommendation/main/Movies%20Recommendation.csv")

# Create a DataFrame from the sample data
df = pd.DataFrame(df)

# Combine the text data from multiple columns into a single Series
combined_text = str(df['Movie_Title']) + ' ' + str(df['Movie_Budget']) + ' ' + str(df['Movie_Production_Country']) + ' ' + str(df['

# Create and fit the TF-IDF vectorizer
tfidf = TfidfVectorizer()
X_tfidf = tfidf.fit_transform(df['Movie_Title'])

# Now, X_tfidf contains the TF-IDF vectorized representation of the combined text data
# You can use X_tfidf for further machine learning tasks
```

```
X_tfidf.shape
```

```
(4760, 4644)
```

```
print(X_tfidf)
```

```
  (0, 3463)     0.7824646116611985
  (0, 1603)     0.6226950549810797
  (1, 4455)     0.7635238578264928
  (1, 3876)     0.6457796207141794
  (2, 2844)     0.7355609287565832
  (2, 1514)     0.6774585744433034
  (3, 1839)     0.7071067811865476
  (3, 1592)     0.7071067811865476
  (4, 398)      0.7915614414287505
  (4, 176)      0.6110895879028205
  (5, 1047)     0.5659826012260144
  (5, 4091)     0.19510206158205726
  (5, 2092)     0.41151705900516006
  (5, 1039)     0.68720636698419
  (6, 1335)     0.7116814919236066
  (6, 1504)     0.6787971787233847
  (6, 4091)     0.1809526021105048
  (7, 2660)     1.0
  (8, 2622)     0.44548787228860987
  (8, 4548)     0.6087089961049882
  (8, 2427)     0.483775893665036
  (8, 2803)     0.44381842955562406
  (9, 3062)     0.4240723693456447
```

```
  (9, 476)      0.32196706943146974
  (9, 1012)     0.3877017276743954
  :     :
  (4749, 4091)  0.31341548959890053
  (4750, 3372)  1.0
  (4751, 391)   0.7997350442113305
  (4751, 176)   0.6003531119768609
  (4752, 791)   0.7071067811865476
  (4752, 2951)  0.7071067811865476
  (4753, 951)   0.7639104548225393
  (4753, 3074)  0.6453222582654509
  (4754, 529)   0.6570767286559245
  (4754, 3258)  0.6570767286559245
  (4754, 2092)  0.36945953136663995
  (4755, 643)   0.7700120936544849
  (4755, 2672)  0.6380292905704539
  (4756, 1822)  0.6476407451819303
  (4756, 4327)  0.4740982950749016
  (4756, 3764)  0.5964832535681666
  (4757, 1067)  1.0
  (4758, 3494)  0.7071067811865476
  (4758, 1587)  0.7071067811865476
  (4759, 2)     0.4782455763614687
  (4759, 3712)  0.4782455763614687
  (4759, 1611)  0.45614752056240154
  (4759, 386)   0.3805937687515619
  (4759, 286)   0.34755905740092197
  (4759, 4157)  0.26237923254749423
```

Get Similarity Score using Cosine Similarity

Cosine similarity is a metric used to measure the similarity between two non-zero vectors in an inner product space. In the context of text data or document similarity, cosine similarity is often used to determine how similar two documents are based on the angle between their vector representations in a multi-dimensional space.

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

```
Similarity_Score = cosine_similarity(X_tfidf)
```

```
Similarity_Score
```

```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 1.]])
```

```
Similarity_Score.shape
```

```
(4760, 4760)
```

Get Movie as Input from user and validate for closest spelling

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

```
Enter your favourite movie name : avtaar
```

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

```
import difflib
```

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

```
['Avatar', 'Gattaca']
```

```
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, 0.0), (1, 0.0), (2, 0.0), (3, 0.0), (4, 0.0), (5, 0.0), (6, 0.0), (7, 0.0), (8, 0.0), (9, 0.0), (10, 0.0), (11, 0.0), (12
```

```
len(Recommendation_Score)
```

```
4760
```

Get All Movies Sort 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, 1.0), (0, 0.0), (1, 0.0), (2, 0.0), (3, 0.0), (4, 0.0), (5, 0.0), (6, 0.0), (7, 0.0), (8, 0.0), (9, 0.0), (10, 0.0), (
```

```
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 . Four Rooms
3 . Star Wars
4 . Finding Nemo
5 . Forrest Gump
6 . American Beauty
7 . Dancer in the Dark
8 . The Fifth Element
9 . Metropolis
10 . My Life Without Me
11 . Pirates of the Caribbean: The Curse of the Black Pearl
12 . Kill Bill: Vol. 1
13 . Jarhead
14 . Apocalypse Now
15 . Unforgiven
16 . The Simpsons Movie
17 . Eternal Sunshine of the Spotless Mind
18 . Amores perros
19 . Pirates of the Caribbean: Dead Man's Chest
20 . A History of Violence
21 . 2001: A Space Odyssey
22 . 8 Mile
23 . Absolute Power
24 . Brazil
25 . Walk the Line
26 . Million Dollar Baby
27 . Billy Elliot
28 . American History X
29 . War of the Worlds
30 . Mars Attacks!
```

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
```

```
Top 10 Movies Suggested for you :

1 . ['Avatar']
2 . []
3 . ['Four Rooms']
4 . ['Star Wars']
5 . ['Finding Nemo']
6 . ['Forrest Gump']
7 . ['American Beauty']
8 . ['Dancer in the Dark']
9 . ['The Fifth Element']
10 . ['Metropolis']
```