Movie Recommendation System

Recommender System is a system that seeks to predict or filter prefernnces according to the user's choices.Recommender systems are utilized in a variety of areas including movies,music,news,books,research articles,search queries,social tags and products in general
Collaborative filtering:Collaborative filtering approaches build a model from the user's past behaviour(i.e. items purchesed or searched by the user) as well as similar decisions made by other users.this model is the used to predict items(or rating for items) that users may have an interest in.
Content-based filtering: Content-based filtering approaches users a series of discrete characteristics of an item in order to recommend additional items with similar properties. Content-based filtering methods are totally based on a description of the item and a profile of the user's preferences it recommends items based on the user's past preference. Let's develop a basic recommenation system using python and pandas
Let's develop a basic recommendations system by suggesting items that are most similar to a particular item.In this case,movies it just tells what movies/items are most similar to the user's moviechoice

Import Library

In [1]:

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

Import Dataset

In [2]:

```
df = pd.read_csv("Dataset-main/Dataset-main/Movies Recommendation.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	Movie_ID	Movie_Title	Movie_Genre	Movie_Language	Movie_Budget	Movie_Popularity	Mov
0	1	Four Rooms	Crime Comedy	en	4000000	22.876230	
1	2	Star Wars	Adventure Action Science Fiction	en	11000000	126.393695	
2	3	Finding Nemo	Animation Family	en	94000000	85.688789	
3	4	Forrest Gump	Comedy Drama Romance	en	55000000	138.133331	
4	5	American Beauty	Drama	en	15000000	80.878605	

5 rows × 21 columns

In [4]:

```
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
```

In [5]:

```
df.shape
```

Out[5]:

```
(4760, 21)
```

In [6]:

```
df.columns
```

Out[6]:

```
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

In [7]:

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

selected five existing features to recommend movies it may vary from one project to another Like one can add vote counts,budget,language etc.

In [8]:

```
df_features.shape
```

Out[8]:

```
(4760, 5)
```

In [9]:

```
df_features
```

Out[9]:

	Movie_Genre	Movie_Keywords	Movie_Tagline	Movie_Cast	Movie_Director
0	Crime Comedy	hotel new year's eve witch bet hotel room	Twelve outrageous guests. Four scandalous requ...	Tim Roth Antonio Banderas Jennifer Beals Madon...	Allison Anders
1	Adventure Action Science Fiction	android galaxy hermit death star lightsaber	A long time ago in a galaxy far, far away...	Mark Hamill Harrison Ford Carrie Fisher Peter ...	George Lucas
2	Animation Family	father son relationship harbor underwater fish...	There are 3.7 trillion fish in the ocean, they...	Albert Brooks Ellen DeGeneres Alexander Gould ...	Andrew Stanton
3	Comedy Drama Romance	vietnam veteran hippie mentally disabled runni...	The world will never be the same, once you've ...	Tom Hanks Robin Wright Gary Sinise Mykelti Wil...	Robert Zemeckis
4	Drama	male nudity female nudity adultery midlife cri...	Look closer.	Kevin Spacey Annette Bening Thora Birch Wes Be...	Sam Mendes
...	...	...	...	...	...
4755	Horror		The hot spot where Satan's waitin'.	Lisa Hart Carroll Michael Des Barres Paul Drak...	Pece Dingo
4756	Comedy Family Drama		It's better to stand out than to fit in.	Roni Akurati Brighton Sharbino Jason Lee Anjul...	Frank Lotito
4757	Thriller Drama	christian film sex trafficking	She never knew it could happen to her...	Nicole Smolen Kim Baldwin Ariana Stephens Brys...	Jaco Booyens
4758	Family				
4759	Documentary	music actors legendary perfomer classic hollyw...		Tony Oppedisano	Simon Napier-Bell

4760 rows × 5 columns

In [10]:

```
atures['Movie_Keywords']+' '+df_features['Movie_Tagline']+' '+df_features['Movie_Cast']+' '+
```

In [11]:

```
X
```

Out[11]:

```
0       Crime Comedy hotel new year's eve witch bet ho...
1       Adventure Action Science Fiction android galax...
2       Animation Family father son relationship harbo...
3       Comedy Drama Romance vietnam veteran hippie me...
4       Drama male nudity female nudity adultery midli...
                              ...
4755    Horror  The hot spot where Satan's waitin'. Li...
4756    Comedy Family Drama  It’s better to stand out ...
4757    Thriller Drama christian film sex trafficking ...
4758                                             Family
4759    Documentary music actors legendary perfomer cl...
Length: 4760, dtype: object
```

In [12]:

```
X.shape
```

Out[12]:

```
(4760,)
```

Get Feature Text Conversion to Tokens

In [13]:

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

In [14]:

```
tfidf = TfidfVectorizer()
```

In [15]:

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

In [16]:

```
X.shape
```

Out[16]:

```
(4760, 17258)
```

In [17]:

```
print(X)
```

```
  (0, 617)	0.1633382144407513
  (0, 492)	0.1432591540388685
  (0, 15413)	0.1465525095337543
  (0, 9675)	0.14226057295252661
  (0, 9465)	0.1659841367820977
  (0, 1390)	0.16898383612799558
  (0, 7825)	0.09799561597509843
  (0, 1214)	0.13865857545144072
  (0, 729)	0.13415063359531618
  (0, 13093)	0.1432591540388685
  (0, 15355)	0.10477815972666779
  (0, 9048)	0.0866842116160778
  (0, 11161)	0.06250380151644369
  (0, 16773)	0.17654247479915475
  (0, 5612)	0.08603537588547631
  (0, 16735)	0.10690083751525419
  (0, 7904)	0.13348000542112332
  (0, 15219)	0.09800472886453934
  (0, 11242)	0.07277788238484746
  (0, 3878)	0.11998399582562203
  (0, 5499)	0.11454057510303811
  (0, 7071)	0.19822417598406614
  (0, 7454)	0.14745635785412262
  (0, 1495)	0.19712637387361423
  (0, 9206)	0.15186283580984414
  :	:
  (4757, 5455)	0.12491480594769522
  (4757, 2967)	0.16273475835631626
  (4757, 8464)	0.23522565554066333
  (4757, 6938)	0.17088173678136628
  (4757, 8379)	0.17480603856721913
  (4757, 15303)	0.07654356007668191
  (4757, 15384)	0.09754322497537371
  (4757, 7649)	0.11479421494340192
  (4757, 10896)	0.14546473055066447
  (4757, 4494)	0.05675298448720501
  (4758, 5238)	1.0
  (4759, 11264)	0.33947721804318337
  (4759, 11708)	0.33947721804318337
  (4759, 205)	0.3237911628497312
  (4759, 8902)	0.3040290704566037
  (4759, 14062)	0.3237911628497312
  (4759, 3058)	0.2812896191863103
  (4759, 7130)	0.26419662449963793
  (4759, 10761)	0.3126617295732147
  (4759, 4358)	0.18306542312175342
  (4759, 14051)	0.20084315377640435
  (4759, 5690)	0.19534291014627303
  (4759, 15431)	0.19628653185946862
  (4759, 1490)	0.21197258705292082
  (4759, 10666)	0.15888268987343043
```

Get Similarity Score using Cosine Similarity

cosine_similarity computers the L2-normalized dot product of vectors Euclidean(L2)normalization projects the vectors onto the unit sphere. and their dot product is then the cosine of the angle between the points denoted by the vectors

In [18]:

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

In [19]:

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

In [20]:

```
Similarity_Score
```

Out[20]:

```
array([[1.        , 0.01351235, 0.03570468, ..., 0.        , 0.        ,
        0.        ],
       [0.01351235, 1.        , 0.00806674, ..., 0.        , 0.        ,
        0.        ],
       [0.03570468, 0.00806674, 1.        , ..., 0.        , 0.08014876,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.08014876, ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

In [21]:

```
Similarity_Score.shape
```

Out[21]:

```
(4760, 4760)
```

Get Movie Name as input from user and Validate for Closest Spelling

In [22]:

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

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

In [26]:

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

In [27]:

```
import difflib
```

In [30]:

```
ovie_Recommendation = difflib.get_close_matches(Favourite_Movie_Name,All_Movies_Title_List)
```

In [31]:

```
print(Movie_Recommendation)
```

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

In [32]:

```
1 Close_Match = Movie_Recommendation[0]
```

In [33]:

```
print(Close_Match)
```

```
Avatar
```

In [34]:

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

```
2692
```

In [35]:

```
#getting a lis of similar movies
Recommendation_Score = list(enumerate(Similarity_Score[Index_of_Close_Match_Movie]))
```

In [36]:

```
print(Recommendation_Score)
```

```
[(0, 0.009805093506053453), (1, 0.0), (2, 0.0), (3, 0.00800429043895183),
(4, 0.0026759665928032302), (5, 0.009639835665946627), (6, 0.0049636657561
850026), (7, 0.012848827437220958), (8, 0.0027543335470164663), (9, 0.0060
7882290416431), (10, 0.007539724639541887), (11, 0.0026263170118314906),
(12, 0.002708340354961457), (13, 0.012904730427356216), (14, 0.0), (15, 0.
022556564866386044), (16, 0.005959078936688496), (17, 0.0), (18, 0.0136398
24714195078), (19, 0.008784739948684396), (20, 0.0026527570934446066), (2
1, 0.015211614027840471), (22, 0.006522322352622825), (23, 0.0026429172195
160193), (24, 0.0016564482636435309), (25, 0.025600660315408176), (26, 0.0
024815199490618002), (27, 0.0047922703978129), (28, 0.0), (29, 0.023288277
583204436), (30, 0.004648836119227042), (31, 0.006723965537835127), (32,
0.007984548069367697), (33, 0.018612326068635436), (34, 0.0074396222674798
48), (35, 0.0060612328203774185), (36, 0.0), (37, 0.0), (38, 0.00808542827
4959462), (39, 0.0046323263203813065), (40, 0.015305064222782005), (41, 0.
0028220612513682524), (42, 0.007236825272071698), (43, 0.01485128947451648
9), (44, 0.03961780430399104), (45, 0.08999324643162435), (46, 0.018554995
96172605), (47, 0.010374759033888029), (48, 0.015673410180680997), (49, 0.
0), (50, 0.006986992676753986), (51, 0.014965979411782002), (52, 0.0136008
04094978335), (53, 0.0), (54, 0.0), (55, 0.0), (56, 0.006687995450791239),
```

In [37]:

```
len(Recommendation_Score)
```

Out[37]:

```
4760
```

Get All Movies Sort Based on Recommendation Score wrt Favourite Movie

In [38]:

```
# sorting the movies based on their similarity score

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

```
[(2692, 1.0000000000000002), (3276, 0.11904275527845871), (3779, 0.1018580
5797079382), (62, 0.10153560702418994), (2903, 0.10063787314386034), (164
7, 0.09397055536069451), (4614, 0.09362226751043302), (4375, 0.09117512301
489193), (45, 0.08999324643162435), (1383, 0.08425242441722802), (110, 0.0
8361784775029485), (628, 0.08334515876919323), (1994, 0.0828783534525221
6), (2558, 0.08267633224298852), (1070, 0.08104448918225104), (4378, 0.078
94345402700793), (1341, 0.07732693809361939), (1977, 0.07510309168081497),
(3465, 0.07411089841255805), (3053, 0.0732438108456325), (4116, 0.07264153
003988619), (1982, 0.07246569778553744), (2538, 0.06802035746289192), (324
8, 0.06683400770968473), (3946, 0.06577120166835922), (3480, 0.06560363079
666712), (254, 0.06351452702158421), (590, 0.06275727122098754), (3450, 0.
06274272831079739), (1886, 0.06267985852941994), (4594, 0.062469952104989
4), (2112, 0.06218435141221765), (84, 0.0618237599684129), (675, 0.0617699
1517572303), (3854, 0.06161566270378365), (1134, 0.06151448371353247), (34
63, 0.06070604565602541 5), (4252, 0.06059815508412411), (4137, 0.060477037
09769184), (1118, 0.05998954734066491), (4389, 0.059627372790876695), (338
5, 0.05898328865604495), (4062, 0.05895899420588936), (282, 0.058792850178
83316), (4398, 0.05848106495843603), (424, 0.05839654732699123), (2358, 0.
05826769637272555), (3462, 0.057434079728437545), (2985, 0.057173552958398
```

In [40]:

```
# print the name of similar movies based on the index

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
1 , Caravans
1 , My Week with Marilyn
1 , Brokeback Mountain
1 , Harry Brown
1 , Night of the Living Dead
1 , The Curse of Downers Grove
1 , The Boy Next Door
1 , Back to the Future
1 , The Juror
1 , Some Like It Hot
1 , Enough
1 , The Kentucky Fried Movie
1 , Eye for an Eye
1 , Welcome to the Sticks
1 , Alice Through the Looking Glass
1 , Superman III
```

Top 10 Movie Recommendation System

In [*]:

```
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[i],reverse=True)
print('Top 10 Movie 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<11):
        print(i,',',title_from_index)
        i++1
```

Enter your favourite movie name:

avtaar

In [ ]:

```

```

In [ ]:

```

```