Movie Recommendation System

Recommender system is a system that seeks to predict or filter preferences according to the user's choice. Recommender system are utilized in a variety of areas including movies,music,news,books,research articales,search queries,social tags,and products in genral.

Recommender systemms product a list of recommendation in any of the two ways-

collaborative filtering: collaborative filtering apporaches build a model from user's past behavior (i.e. items purchased or searched by the user) as well as similar decision made by other users. this model is then used to predict items(or rating for items) that users may have an interse in.

Content-based filtering: contact-based filteing approaches uses a series of discrete characteristics of an item in oredr to remcomend additional items items with similar properties. content- based filtering methods are totally based on a description of the item and a profile of the user's prefernces. it recommends items based on the user's past preferences. let's develop a basic recommendation system using python and pandas.

let's develop a basic recommendation system by suggesting items that are most similar items that are most similar to a particular item, in this case, movies. it just tells what movies/items are most similar to user's movie choice

Import Libary

```
import pandas as pd
```

```
import numpy as np
```

Import Dataset

```
df = pd.read_csv(r'https://raw.githubusercontent.com/YBI-Foundation/Dataset/main/Movies%20Re
```

```
df.head()
```

	Movie_ID	Movie_Title	Movie_Genre	Movie_Language	Movie_Budget	Movie_Popularity	I
0	1	Four Rooms	Crime Comedy	en	4000000	22.876230	

```
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_Directo
```

```
df_features.shape
```

```
(4760, 5)
```

```
df_features
```

	Movie_Genre	Movie_Keywords	Movie_Tagline	Mo
0	Crime Comedy	hotel new year's eve witch bet hotel room	Twelve outrageous guests. Four scandalous requ...	Tim Roth Antonic Jennifer Beal
1	Adventure Action Science Fiction	android galaxy hermit death star lightsaber	A long time ago in a galaxy far, far away...	Mark Hamill Ha Carrie Fish
2	Animation Family	father son relationship harbor underwater fish...	There are 3.7 trillion fish in the ocean, they...	Albert Br DeGeneres
3	Comedy Drama Romance	vietnam veteran hippie mentally disabled runni...	The world will never be the same, once you've ...	Tom Hanks Rc Gary Sinise M
4	Drama	male nudity female nudity adultery midlife cri...	Look closer.	Kevin Spac Bening Thora
...	...	...	...	
4755	Horror		The hot spot where Satan's waitin'.	Lisa Hart Carr Des Barres F
4756	Comedy Family Drama		It's better to stand out than to fit in.	Roni Akura Sharbino Jason I
4757	Thriller Drama	christian film sex trafficking	She never knew it could happen to her...	Nicole Smolen K Ariana Steph
4758	Family			
4759	Documentary	music actors legendary perfomer classic hollyw...		Tony C

4760 rows × 5 columns

```
X = df_features['Movie_Genre'] + ' '+ df_features['Movie_Keywords'] + ' ' + df_features['Mov
```

```
X
```

```
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
```

```
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, 17258)
```

```
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

```
from sklearn.metrics.pairwise import cosine_similarity

Similarity_Score = cosine_similarity(X)

Similarity_Score

    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.        ]])
```

```
Similarity_Score.shape
```

```
(4760, 4760)
```

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

```
Favouritre_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(Favouritre_Movie_Name, All_Movies_Title_Lis
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
```

```
# getting a list of similar movies
```

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

```
len(Recommendation_Score)
```

```
4760
```

✓ 0 से° 12:48 pm पर पूरा किया गया