Movie Recommendation System

Objective : Is to create a recommendation system that seeks to predict or fiter preferences according to the users choices. Recommender systems produce a list of recommendations in any of the two ways - Collaborative filtering: Collaborative filtering approaches build a model from the user's past behavior (Le, Items purchased or searched by the user) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that users may have an interestin Content-based filtering: Content based filtering approaches uses a series of discrete characteristics of an item in order to recommend additionalitems with similar properties. Content-based filtering methods are totally based on a description of the item and a profile of the users preferences, It recommends items based on the user's past preferences

IMPORT LIBRARY

```
import pandas as pd
```

```
import numpy as np
```

IMPORT DATASET

```
df=pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Movies%20Recommendation.csv')
```

DESCRIBE DATA

```
df.head()
```

Show hidden output

```
df.info()
```

Show hidden output

```
df.shape
```

Show hidden output

```
df.columns
```

Show hidden output

GET FEATURE SELECTION

```
df_features=df[['Movie_Genre','Movie_Keywords','Movie_Tagline','Movie_Cast','Movie_Director']].fillna('')
```

```
df_features.shape
```

Show hidden output

```
df_features
```

Show hidden output

Next steps: Generate code with df_features View recommended plots

```
x = df_features['Movie_Genre'] + ' ' + df_features['Movie_Keywords'] + ' ' + df_features['Movie_Tagline'] + ' ' + df_features['Movie_Cast']
```

```
x
```

Show hidden output

```
x.shape
```

Show hidden output

GET FEATURE TEXT CONVERSION TO TOKENS

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

```
tfidf = TfidfVectorizer()
```

```
x = tfidf.fit_transform(x)
```

```
x.shape
```

Show hidden output

```
print(x)
```

Show hidden output

GET SIMILARITY SCORE USING COSINE SIMILARITY (cosine_similarity computes the L2-Normalised dost product to vector)

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

```
Similarity_Score = cosine_similarity(x)
```

```
Similarity_Score
```

Show hidden output

```
Similarity_Score.shape
```

Show hidden output

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 : avtaar
```

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

```
import difflib
```

Suggested code may be subject to a license | Naga-Himaja/YBI-Foundation-Internship

```
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)
```

Show hidden output

```
len(Recommendation_Score)
```

Show hidden output

GET ALL MOVIES SORTED BASED ON RECOMMENDATION SCORE WRT FAVIOURITE MOVIES

Suggested code may be subject to a license | Naga-Himaja/YBI-Foundation-Internship

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

```
[(2692, 1.0000000000000002), (3276, 0.11904275527845871), (3779, 0.10185805797079382), (62, 0.10153560702418994), (2903, 0.1006378731438
```

Suggested code may be subject to a license | Naga-Himaja/YBI-Foundation-Internship

```
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
```

Show hidden output

TOP 10 MOVIE RECOMMENDATION SYSTEM

Suggested code may be subject to a license | Naga-Himaja/YBI-Foundation-Internship

```
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
```

Show hidden output

Start coding or generate with AI.

Start coding or generate with AI.