movie recommender system

importing the dependencies

```
import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

data collection and pre processing

```
# loading the data from the csv file to apandas dataframe
movies_data = pd.read_csv('/movies.csv')
```

```
#printing the first 5 rows of the dataframe
movies_data.head()
```

	index	budget	genres	homepage	id	keywords	original_
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...	
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6	
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc comics crime fighter terrorist secret ident...	
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based on novel mars medallion space travel pri...	

```
# number of rows and columns in the data frame
movies_data.shape
```

```
(4803, 24)
```

```
# selecting the relevant features for recommdation
selected_features = ['genres','keywords','tagline','cast','director']
print(selected_features)
```

```
['genres', 'keywords', 'tagline', 'cast', 'director']
```

```
#replacing the null values with null string
for feature in selected_features:
```

```
  movies_data[feature] = movies_data[feature].fillna('')


# combining all the selected features
combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+movies_data['tagline']+' '


print(combined_features)
```

```
0       Action Adventure Fantasy Science Fiction cultu...
1       Adventure Fantasy Action ocean drug abuse exot...
2       Action Adventure Crime spy based on novel secr...
3       Action Crime Drama Thriller dc comics crime fi...
4       Action Adventure Science Fiction based on nove...
                              ...
4798    Action Crime Thriller united states\u2013mexic...
4799    Comedy Romance  A newlywed couple's honeymoon ...
4800    Comedy Drama Romance TV Movie date love at fir...
4801      A New Yorker in Shanghai Daniel Henney Eliza...
4802    Documentary obsession camcorder crush dream gi...
Length: 4803, dtype: object
```

```
#converting the text data to feature vectors
vectorizer = TfidfVectorizer()


feature_vectors = vectorizer.fit_transform(combined_features)


print(feature_vectors)
```

```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 124266 stored elements and shape (4803, 17318)>
  Coords        Values
  (0, 201)      0.07860022416510505
  (0, 274)      0.09021200873707368
  (0, 5274)     0.11108562744414445
  (0, 13599)    0.1036413987316636
  (0, 5437)     0.1036413987316636
  (0, 3678)     0.21392179219912877
  (0, 3065)     0.22208377802661425
  (0, 5836)     0.1646750903586285
  (0, 14378)    0.33962752210959823
  (0, 16587)    0.12549432354918996
  (0, 3225)     0.24960162956997736
  (0, 14271)    0.21392179219912877
  (0, 4945)     0.24025852494110758
  (0, 15261)    0.07095833561276566
  (0, 16998)    0.1282126322850579
  (0, 11192)    0.09049319826481456
  (0, 11503)    0.27211310056983656
  (0, 13349)    0.15021264094167086
  (0, 17007)    0.23643326319898797
  (0, 17290)    0.20197912553916567
  (0, 13319)    0.2177470539412484
  (0, 14064)    0.20596090415084142
  (0, 16668)    0.19843263965100372
  (0, 14608)    0.15150672398763912
  (0, 8756)     0.22709015857011816
  :     :
```

```
        (4801, 403)   0.17727585190343229
        (4801, 4835)  0.24713765026964
        (4801, 17266) 0.28860981849329476
        (4801, 13835) 0.27870029291200094
        (4801, 13175) 0.28860981849329476
        (4801, 17150) 0.3025765103586468
        (4801, 3511)  0.3025765103586468
        (4801, 13948) 0.3025765103586468
        (4801, 7269)  0.3025765103586468
        (4802, 11161) 0.17867407682173203
        (4802, 4518)  0.16784466610624255
        (4802, 2129)  0.3099656128577656
        (4802, 4980)  0.16078053641367315
        (4802, 6155)  0.18056463596934083
        (4802, 3436)  0.21753405888348784
        (4802, 4528)  0.19504460807622875
        (4802, 1316)  0.1960747079005741
        (4802, 12989) 0.1696476532191718
        (4802, 4371)  0.1538239182675544
        (4802, 6417)  0.21753405888348784
        (4802, 4608)  0.24002350969074696
        (4802, 2425)  0.24002350969074696
        (4802, 3654)  0.262512960498006
        (4802, 5367)  0.22969114490410403
        (4802, 6996)  0.5700048226105303
```

Start coding or generate with AI.

cosine similarity

```
# getting the similarity scores using cosine similarity
similarity=cosine_similarity(feature_vectors)
```

```
print(similarity)
```

```
[[1.         0.07219487 0.037733   ... 0.         0.         0.        ]
 [0.07219487 1.         0.03281499 ... 0.03575545 0.         0.        ]
 [0.037733   0.03281499 1.         ... 0.         0.05389661 0.        ]
 ...
 [0.         0.03575545 0.         ... 1.         0.         0.02651502]
 [0.         0.         0.05389661 ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.02651502 0.         1.        ]]
```

```
print(similarity.shape)
```

```
(4803, 4803)
```

```
# getting the movie name form the user
movie_name = input('enter your favourite movie name : ')
```

```
enter your favourite movie name : avatar
```

```
# creating a list with all the movie names given in the dataset
list_of_all_titles = movies_data['title'].tolist()
```

```
print(list_of_all_titles)
```

```
['Avatar', "Pirates of the Caribbean: At World's End", 'Spectre', 'The Dark Knight Rises', 'John
```

```
# finding the close match for the movie name given by the user
find_close_match = difflib.get_close_matches(movie_name,list_of_all_titles)
print(find_close_match)
```

```
['Avatar']
```

```
close_match = find_close_match[0]
print(close_match)
```

```
Avatar
```

```
# finding the index of the movie with title
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)
```

```
0
```

```
# getting a list of similar movies
similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)
```

```
[(0, np.float64(1.0)), (1, np.float64(0.07219486822992488)), (2, np.float64(0.037732999577179294
```

```
len(similarity_score)
```

```
4803
```

```
# sorting the movies based on their similarity score
sorted_similar_movies = sorted(similarity_score,key=lambda x:x[1],reverse=True)
print(sorted_similar_movies)
```

```
[(0, np.float64(1.0)), (3158, np.float64(0.2494676630753241)), (2403, np.float64(0.2484146259590
```

```
# print the name of similar movies based on the index
print('movies suggested for you:\n')
i = 1
for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<30):
    print (i,'.',title_from_index)
    i+=1
```

```
movies suggested for you:

 1 . Avatar
```

```
2 . Alien
3 . Aliens
4 . Guardians of the Galaxy
5 . Star Trek Beyond
6 . Star Trek Into Darkness
7 . Galaxy Quest
8 . Alien³
9 . Cargo
10 . Trekkies
11 . Gravity
12 . Moonraker
13 . Jason X
14 . Pocahontas
15 . Space Cowboys
16 . The Helix... Loaded
17 . Lockout
18 . Event Horizon
19 . Space Dogs
20 . Machete Kills
21 . Gettysburg
22 . Clash of the Titans
23 . Star Wars: Clone Wars: Volume 1
24 . The Right Stuff
25 . Terminator Salvation
26 . The Astronaut's Wife
27 . Planet of the Apes
28 . Star Trek
29 . Wing Commander
```

MOVIE RECOMMENDATION SYSTEM

```
movie_name = input(' Enter your favourite movie name : ')
list_of_all_titles = movies_data['title']. tolist()
find_close_match = difflib.get_close_matches (movie_name, list_of_all_titles)
close_match = find_close_match[0]
index_of_the_movie = movies_data[movies_data.title == close_match][ 'index']. values [0]
similarity_score = list (enumerate(similarity[index_of_the_movie]))
sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)
print ( 'Movies suggested for you : \n')
```