Name: Nishendu Mishra

Topic: Movie Recommendation System

Email: nishendumishra73@gmail.

Importing the dependencies

```
import numpy as np
import pandas as pd
import difflib   #close match of the movie name given by the user, for spelling mistake and all
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

+ Code + Text

Data Collection and Pre-Processing

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

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

	index	budget	genres	homepage	id	keyword
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	cultur clas futur spac wa spac color so.
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocea dru abus exot islan east indi trad.
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	sp based c nov secr age sequ mi
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	c comic crim fight terrori secr ident.
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based c nov mai medallic spac trav pri.

5 rows × 24 columns

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

```
(4803, 24)
```

```
#selecting the relevant features for recommendations
```

```
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 five selected features

combined_features = movies_data['genres']+ ' '+ movies_data['keywords'] + ' '+ movies_data['tagline']+' '+  movies_data['cast'] +' '+ mov
```

```
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)
```

```
  (0, 2432)	0.17272411194153
  (0, 7755)	0.1128035714854756
  (0, 13024)	0.1942362060108871
  (0, 10229)	0.16058685400095302
  (0, 8756)	0.22709015857011816
  (0, 14608)	0.15150672398763912
  (0, 16668)	0.19843263965100372
  (0, 14064)	0.20596090415084142
  (0, 13319)	0.2177470539412484
  (0, 17290)	0.20197912553916567
  (0, 17007)	0.23643326319898797
  (0, 13349)	0.15021264094167086
  (0, 11503)	0.27211310056983656
  (0, 11192)	0.09049319826481456
  (0, 16998)	0.1282126322850579
  (0, 15261)	0.07095833561276566
  (0, 4945)	0.24025852494110758
  (0, 14271)	0.21392179219912877
  (0, 3225)	0.24960162956997736
  (0, 16587)	0.12549432354918996
  (0, 14378)	0.33962752210959823
  (0, 5836)	0.1646750903586285
  (0, 3065)	0.22208377802661425
  (0, 3678)	0.21392179219912877
  (0, 5437)	0.1036413987316636
  :	:
  (4801, 17266)	0.2886098184932947
  (4801, 4835)	0.24713765026963996
  (4801, 403)	0.17727585190343226
  (4801, 6935)	0.2886098184932947
  (4801, 11663)	0.21557500762727902
  (4801, 1672)	0.1564793427630879
  (4801, 10929)	0.13504166990041588
  (4801, 7474)	0.11307961713172225
  (4801, 3796)	0.3342808988877418
  (4802, 6996)	0.5700048226105303
  (4802, 5367)	0.22969114490410403
  (4802, 3654)	0.262512960498006
  (4802, 2425)	0.24002350969074696
  (4802, 4608)	0.24002350969074696
  (4802, 6417)	0.21753405888348784
  (4802, 4371)	0.1538239182675544
  (4802, 12989)	0.1696476532191718
  (4802, 1316)	0.1960747079005741
  (4802, 4528)	0.19504460807622875
  (4802, 3436)	0.21753405888348784
  (4802, 6155)	0.18056463596934083
```

```
  (4802, 4980)  0.16078053641367315
  (4802, 2129)  0.3099656128577656
  (4802, 4518)  0.16784466610624255
  (4802, 11161) 0.17867407682173203
```

Cosine Similarity

```
#getting a 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 a movie name from the user

movie_name = input('Enter the name of the movie : ')
```

```
Enter the name of the movie : avatar
```

```
# creating a list with all the movies given in the dataset

list_of_all_titles = movies_data['title'].tolist()
# print(list_of_all_titles)
```

```
#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
```

we found the index because similarity is having index value and the similarity.

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

# we are taking the iron man movie and find the similarity of all the movies. So the movies which are similar to iron man have higher val
```

```
# print(similarity_score)
```

the first value is the index and second is the similarity between movie at that index

```
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)

# lamba x:x[1] means x is representing similarity score and x[1] means second element of each set eg.  0.033570748780675445


#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
30 . Sunshine
```

Movies Recommendation system

```
#incorporating at one place
movie_name = input('Enter the name of the movie : ')

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')

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
```

```
Enter the name of the movie :  batman
Movies suggested for you:

1 . Batman
2 . Batman Returns
3 . Batman & Robin
4 . The Dark Knight Rises
5 . Batman Begins
6 . The Dark Knight
7 . A History of Violence
8 . Superman
9 . Beetlejuice
10 . Bedazzled
11 . Mars Attacks!
12 . The Sentinel
13 . Planet of the Apes
14 . Man of Steel
15 . Suicide Squad
16 . The Mask
17 . Salton Sea
18 . Spider-Man 3
19 . The Postman Always Rings Twice
20 . Hang 'em High
21 . Spider-Man 2
22 . Dungeons & Dragons: Wrath of the Dragon God
23 . Superman Returns
24 . Jonah Hex
25 . Exorcist II: The Heretic
26 . Superman II
27 . Green Lantern
28 . Superman III
29 . Something's Gotta Give
30 . Reds
```

14s completed at 10:58 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.