In [1]:
```
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-process

In [2]:
```
movies_data=pd.read_csv(r"D:\ML_WORK\movies.csv")
```

In [3]:
```
movies_data.head()
```

Out[3]:

	index	budget	genres	homepage	id	keywords	original_language	original_title	overview
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	Avatar	In the 22nd century, a paraplegic Marine is di...
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6	en	Spectre	A cryptic message from Bond's past sends him o...
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc comics crime fighter terrorist secret ident...	en	The Dark Knight Rises	Following the death of District Attorney Harve...
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based on novel mars medallion space travel pri...	en	John Carter	John Carter is a war-weary, former military ca...

5 rows × 24 columns

In [4]:
```
movies_data.shape
```

Out[4]: (4803, 24)

In [5]:
```
selected_features=["genres",'keywords','tagline','cast','director']
print(selected_features)
```

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

In [6]:
```
#replacing null values with null string
for feature in selected_features:
    movies_data[feature]=movies_data[feature].fillna('')
```

In [7]:
```
#combining all the features(concating all the features all together)

combined_features=movies_data['genres']+" "+movies_data['keywords']+" "+movies_data['tagline']+" "+movies
```

In [8]:
```
combined_features.head
```

Out[8]:
```
<bound method NDFrame.head of 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>
```

In [9]:
```
#creating instance of vectorizer
vectorizer=TfidfVectorizer() #-->loading this TfidVec.. that we have imported from Sklearn into vectorize
```

In [10]:
```
#now fitting and transforming all the data
feature_vectors=vectorizer.fit_transform(combined_features)
```

In [11]:
```
print(feature_vectors) # now we can fit this data into the ML model this will give numeric value to text
```

```
  (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
```

In [12]:
```
#Now finding the similarity score-> using cosine similarity, also imported from sklearn
similarity=cosine_similarity(feature_vectors)
```

In [13]:
```
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.        ]]
```

In [14]:
```
print(similarity[0][:9]) # so basically it will pick 1st movie, and compare with all to give a similarity
```

```
[1.         0.07219487 0.037733   0.0125202  0.10702574 0.077869
 0.00823714 0.03613473 0.02960931]
```

In [15]:
```
similarity.shape #1. is movie index, 2nd is similarity score
```

Out[15]:
```
(4803, 4803)
```

In [16]:
```
# now we can ask user to give input fav movie name,

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

```
Enter your favourite movie name : IronMan
```

In [17]:
```
#create a list that contain all the movie names given in that 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 Carter',
'Spider-Man 3', 'Tangled', 'Avengers: Age of Ultron', 'Harry Potter and the Half-Blood Prince', 'Batman v
Superman: Dawn of Justice', 'Superman Returns', 'Quantum of Solace', "Pirates of the Caribbean: Dead Man's
Chest", 'The Lone Ranger', 'Man of Steel', 'The Chronicles of Narnia: Prince Caspian', 'The Avengers', 'Pi
rates of the Caribbean: On Stranger Tides', 'Men in Black 3', 'The Hobbit: The Battle of the Five Armies',
'The Amazing Spider-Man', 'Robin Hood', 'The Hobbit: The Desolation of Smaug', 'The Golden Compass', 'King
Kong', 'Titanic', 'Captain America: Civil War', 'Battleship', 'Jurassic World', 'Skyfall', 'Spider-Man 2',
'Iron Man 3', 'Alice in Wonderland', 'X-Men: The Last Stand', 'Monsters University', 'Transformers: Reveng
e of the Fallen', 'Transformers: Age of Extinction', 'Oz: The Great and Powerful', 'The Amazing Spider-Man
2', 'TRON: Legacy', 'Cars 2', 'Green Lantern', 'Toy Story 3', 'Terminator Salvation', 'Furious 7', 'World
War Z', 'X-Men: Days of Future Past', 'Star Trek Into Darkness', 'Jack the Giant Slayer', 'The Great Gatsb
y', 'Prince of Persia: The Sands of Time', 'Pacific Rim', 'Transformers: Dark of the Moon', 'Indiana Jones
and the Kingdom of the Crystal Skull', 'The Good Dinosaur', 'Brave', 'Star Trek Beyond', 'WALL·E', 'Rush H
our 3', '2012', 'A Christmas Carol', 'Jupiter Ascending', 'The Legend of Tarzan', 'The Chronicles of Narni
a: The Lion, the Witch and the Wardrobe', 'X-Men: Apocalypse', 'The Dark Knight', 'Up', 'Monsters vs Alien
s', 'Iron Man', 'Hugo', 'Wild Wild West', 'The Mummy: Tomb of the Dragon Emperor', 'Suicide Squad', 'Evan
Almighty', 'Edge of Tomorrow', 'Waterworld', 'G.I. Joe: The Rise of Cobra', 'Inside Out', 'The Jungle Boo
k', 'Iron Man 2', 'Snow White and the Huntsman', 'Maleficent', 'Dawn of the Planet of the Apes', 'The Love
rs', '47 Ronin', 'Captain America: The Winter Soldier', 'Shrek Forever After', 'Tomorrowland', 'Big Hero
```

In [18]:
```
#finding the close match for the movie name given by user

find_close_match= difflib.get_close_matches(movie_name,list_of_all_titles)
print(find_close_match)
```

```
['Iron Man', 'Iron Man 3', 'Iron Man 2']
```

In [19]:
```
# we want only 1 value, which will match exactly, not all, only the most similar

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

```
Iron Man
```

In [20]:
```
#finding the index of this movie with title, based on title

index_of_the_movie=movies_data[movies_data.title==close_match]['index'].values[0]
print(index_of_the_movie)
```

```
68
```

In [21]:
```
#getting list of similar movie->now we will take this index and find the similar movie to this

similarity_score= list(enumerate(similarity[index_of_the_movie]))  #enumerate is used to carry a loop in

print(similarity_score) # 1st value index of movie, 2nd is similarity score compare to input
```

```
[(0, 0.033570748780675445), (1, 0.0546448279236134), (2, 0.013735500604224323), (3, 0.006468756104392058),
(4, 0.03268943310073386), (5, 0.013907256685755473), (6, 0.07692837576335507), (7, 0.23944423963486405),
(8, 0.007882387851851008), (9, 0.07599206098164225), (10, 0.07536074882460438), (11, 0.01192606921174529),
(12, 0.013707618139948929), (13, 0.012376074925089967), (14, 0.09657127116284188), (15, 0.0072862713838167
43), (16, 0.22704403782296803), (17, 0.013112928084103857), (18, 0.04140526820609594), (19, 0.078832825468
34255), (20, 0.07981173664799915), (21, 0.011266873271064948), (22, 0.006892575895462364), (23, 0.00659909
7891242659), (24, 0.012665208122549737), (25, 0.0), (26, 0.21566241096831154), (27, 0.030581282093826635),
(28, 0.061074402219665376), (29, 0.014046184258938898), (30, 0.0807734659476981), (31, 0.3146705244947750
6), (32, 0.02878209913426701), (33, 0.13089810941050173), (34, 0.0), (35, 0.035350090674865595), (36, 0.03
185325269937555), (37, 0.008024326882532318), (38, 0.1126182690487113), (39, 0.08902766481306311), (40, 0.
008086007019135987), (41, 0.06454289714171595), (42, 0.0), (43, 0.054316692518940446), (44, 0.006244741632
576977), (45, 0.023530724758699103), (46, 0.14216268867232237), (47, 0.03716851751705058), (48, 0.01375572
5647812333), (49, 0.0), (50, 0.012978759995781826), (51, 0.027557058720715163), (52, 0.030326407086366649),
(53, 0.022454895898373586), (54, 0.04976759996785291), (55, 0.0141234086767521), (56, 0.0376540797713692
4), (57, 0.03611189350591964), (58, 0.006559522468571584), (59, 0.031705778955831605), (60, 0.013840362984
553644), (61, 0.0334730220502558), (62, 0.013292388095397085), (63, 0.007052604148534629), (64, 0.15299924
139445145), (65, 0.005907393049485784), (66, 0.007215870794201356), (67, 0.028674707838967486), (68, 1.000
0000000000002), (69, 0.01303791167475042), (70, 0.03296111942451571), (71, 0.017647877890961963), (72, 0.0
7116913464035789), (73, 0.0), (74, 0.02605515302790224), (75, 0.012706074782984477), (76, 0.05139282303706
```

In [22]:
```
len(similarity_score)
```

Out[22]: 4803

In [23]:
```
# we want only which has highest similarity value, we will sort this from high similarity value to lowest

sorted_similar_movies=sorted(similarity_score,key= lambda x:x[1],reverse=True)
```

In [24]:
```
#we will print the name of similar movie based on 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<16):
        print(i,'.',title_from_index)
        i+=1
```

```
Movies suggested for you :

1 . Iron Man
2 . Iron Man 2
3 . Iron Man 3
4 . Avengers: Age of Ultron
5 . The Avengers
6 . Captain America: Civil War
7 . Captain America: The Winter Soldier
8 . Ant-Man
9 . X-Men
10 . Made
11 . X-Men: Apocalypse
12 . X2
13 . The Incredible Hulk
14 . The Helix... Loaded
15 . X-Men: First Class
```

In [25]:
```
# Collecting code at one place
```

In [26]:

```
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')

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<16):
        print(i,'.',title_from_index)
        i+=1
```

```
Enter your favourite movie name : IronMan
Movies suggested for you :

1 . Iron Man
2 . Iron Man 2
3 . Iron Man 3
4 . Avengers: Age of Ultron
5 . The Avengers
6 . Captain America: Civil War
7 . Captain America: The Winter Soldier
8 . Ant-Man
9 . X-Men
10 . Made
11 . X-Men: Apocalypse
12 . X2
13 . The Incredible Hulk
14 . The Helix... Loaded
15 . X-Men: First Class
```