Importing the dependencies

In [4]:
```
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

In [5]:
```
# loading the data from the csv file to apandas dataframe
movies_data = pd.read_csv('movies.csv')
```

In [6]:
```python
# printing the first 5 rows of the dataframe
movies_data.head()
```

Out[6]:

	index	budget	genres	homepage	id	keywords
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...

5 rows × 24 columns

In [7]:
```python
# number of rows and columns in the data frame

movies_data.shape
```

Out[7]: (4803, 24)

In [8]:
```python
# selecting the relevant features for recommendation

selected_features = ['genres','keywords','tagline','cast','director']
print(selected_features)
```

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

In [10]:
```
# replacing the null valuess with null string

for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('')
```

In [11]:
```
# combining all the 5 selected features

combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+
```

In [12]:
```
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
```

In [13]:
```
# converting the text data to feature vectors

vectorizer = TfidfVectorizer()
```

In [14]:
```
feature_vectors = vectorizer.fit_transform(combined_features)
```

In [16]:
```
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

In [17]:
```
# getting the similarity scores using cosine similarity

similarity = cosine_similarity(feature_vectors)
```

In [18]:
```
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 [19]:
```
print(similarity.shape)
```

```
(4803, 4803)
```

Getting the movie name from the user

In [20]:
```
# getting the movie name from the user

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

```
 Enter your favourite movie name : avtar
```

In [21]:
```
# 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 Carter', 'Spider-Man 3', 'Tangled', 'Avenger
s: 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',
'Pirates of the Caribbean: On Stranger Tides', 'Men in Black 3', 'The
Hobbit: The Battle of the Five Armies', 'The Amazing Spider-Man', 'Rob
in Hood', 'The Hobbit: The Desolation of Smaug', 'The Golden Compass',
'King Kong', 'Titanic', 'Captain America: Civil War', 'Battleship', 'J
urassic World', 'Skyfall', 'Spider-Man 2', 'Iron Man 3', 'Alice in Won
derland', 'X-Men: The Last Stand', 'Monsters University', 'Transformer
s: Revenge 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 Darkn
ess', 'Jack the Giant Slayer', 'The Great Gatsby', '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 Dinosa
```

In [22]:

```
# finding the close match for the movie name given by the user

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

```
['Avatar', 'Salvador', 'Water']
```

In [23]:

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

```
Avatar
```

In [25]:

```
# finding the index of the movie with title

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

```
0
```

In [26]:

```
# getting a list of similar movies

similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)
```

```
[(0, 1.0000000000000002), (1, 0.07219486822992491), (2, 0.037732999577
17928), (3, 0.012520204623868908), (4, 0.10702574467235304), (5, 0.077
86899789424222), (6, 0.008237143013608844), (7, 0.03613473061484884),
(8, 0.02960930964063025), (9, 0.02628716743995174), (10, 0.09261074046
755373), (11, 0.012717759249124133), (12, 0.027217360083100117), (13,
0.02956975252334751), (14, 0.06915925473724742), (15, 0.01955159449930
903), (16, 0.03426340578061641), (17, 0.026036564614294145), (18, 0.05
714759266724124), (19, 0.0389505953521203), (20, 0.0397148021541549
5), (21, 0.01201480380565613), (22, 0.03043869426198959), (23, 0.04592
6535588179496), (24, 0.04623989017965259), (25, 0.042849260959502256),
(26, 0.07010711150614286), (27, 0.037198076232328854), (28, 0.04083909
796927843), (29, 0.03858648330156397), (30, 0.07893753610792024), (31,
0.060555221380551486), (32, 0.030362745635800832), (33, 0.035910214700
68869), (34, 0.0), (35, 0.03769674103474844), (36, 0.04891087950911423
4), (37, 0.0857517399841944), (38, 0.02720463661843483), (39, 0.038456
29888205785), (40, 0.013957672280167852), (41, 0.03448587045127904),
(42, 0.0), (43, 0.1348209130228474), (44, 0.08883923145894224), (45,
0.03663594218725408), (46, 0.09425361782127925), (47, 0.20115287461144
912), (48, 0.02731288064333537), (49, 0.01106610661886975), (50, 0.056
```

In [28]:

```
len(similarity_score)
```

Out[28]:

```
4803
```

In [27]:

```
# sorting the movies based on their similarity score

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reve
print(sorted_similar_movies)
```

```
[(0, 1.0000000000000002), (3158, 0.24946766307532411), (2403, 0.248414
62595906275), (94, 0.24505931974059814), (56, 0.2037806964828543), (4
7, 0.20115287461144912), (1053, 0.19702752258651426), (838, 0.18017023
369312357), (3730, 0.17646241185313413), (4593, 0.1744884579741517),
(239, 0.17441748680810654), (1531, 0.1682605817219649), (2696, 0.16503
460259176522), (812, 0.1606230190749179), (643, 0.15644455512484967),
(4401, 0.15468923545220403), (2198, 0.15217161971893786), (770, 0.1502
5726727753488), (1951, 0.14933372705282924), (2229, 0.1466180128549225
7), (1922, 0.1448197430191312), (206, 0.14226144606175545), (3208, 0.1
401230206493547), (1759, 0.13899056016968867), (43, 0.134820913022847
4), (1473, 0.13476547670086914), (278, 0.13291021545504), (158, 0.1325
2892131627672), (1650, 0.13024318650645417), (1275, 0.1260221630479115
3), (3439, 0.12480340331169382), (661, 0.12153002734138185), (3202, 0.
12144749322246053), (4332, 0.12002556168548507), (3105, 0.119484664942
12534), (775, 0.11847062758014923), (1099, 0.11781613451227567), (461,
0.11712791118789065), (108, 0.11622780359046815), (539, 0.115779694665
67734), (4108, 0.11562098658293778), (1354, 0.11321938517508143), (74
0, 0.11313014687818856), (942, 0.11302774792781604), (977, 0.112550773
90714462), (305, 0.11247355316980151), (855, 0.11241708754892185), (20
```

In [28]:

```
# 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
  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 Sytem

In [29]:

```
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_title

close_match = find_close_match[0]

index_of_the_movie = movies_data[movies_data.title == close_match]['index'

similarity_score = list(enumerate(similarity[index_of_the_movie]))

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reve

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
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

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

1 . Cars
2 . Cars 2
3 . The Fast and the Furious: Tokyo Drift
4 . 2 Fast 2 Furious
5 . The Final Destination
6 . Death Race
7 . Days of Thunder
8 . Furious 7
9 . Herbie Fully Loaded
10 . Larry the Cable Guy: Health Inspector
11 . The Fast and the Furious
12 . The Cable Guy
13 . Back to the Future Part II
14 . Witless Protection
15 . Gone in Sixty Seconds
16 . Turbo
17 . The Transporter
18 . Cheaper by the Dozen
19 . Vacation
20 . Back to the Future
21 . The Siege
22 . The Woman Chaser
23 . Toy Story
24 . Speed Racer
25 . American Graffiti
26 . Bottle Rocket
27 . Bride of Chucky
28 . Life or Something Like It
29 . Cheaper by the Dozen 2
```

In [ ]: