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('/content/movies.csv', sep=',',
                header='infer',
                index_col=None,
                usecols=None,
                squeeze=False,
                engine='python',
                quotechar='"',
                error_bad_lines=False)
```

```
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326: FutureWarning: The error_bad_lines argument has been depr

  exec(code_obj, self.user_global_ns, self.user_ns)
Skipping line 2868: unexpected end of data
```

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

| index | budget | genres | homepage | id | keywords |
|---|---|---|---|---|---|

```
# number of rows and columns in the data frame

movies_data.shape
```

    (2866, 24)

colony

```
# selecting the relevant features for recommendation

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

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

```
# replacing the null valuess with null string

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

Crime agent

```
# combining all the 5 selected features

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

Action comics

```
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...
                               ...
2861    Thriller Horror vampire dracula bite blood vla...
2862    Family Comedy based on novel job interview bad...
2863    Drama Romance bachelor beautiful prejudice sui...
2864    Science Fiction Drama Thriller artificial inte...
2865    Adventure scotland biography 18th century high...
Length: 2866, dtype: object
```

```
# converting the text data to feature vectors

vectorizer = TfidfVectorizer()


feature_vectors = vectorizer.fit_transform(combined_features)


print(feature_vectors)
```

```
  (0, 1160)     0.164314481344402
  (0, 3832)     0.11640168863502841
  (0, 6403)     0.1925488727149144
  (0, 4991)     0.1600885757983711
  (0, 4292)     0.23685245608367106
  (0, 7188)     0.15633324481850083
  (0, 8188)     0.20689427986446918
  (0, 6899)     0.21112018541050012
  (0, 6535)     0.22480997625975965
  (0, 8510)     0.22480997625975965
  (0, 8384)     0.22159200163941709
  (0, 6552)     0.15062469049514243
  (0, 5620)     0.25574686386273904
  (0, 5462)     0.09212531261319962
  (0, 8380)     0.1201410067245958
  (0, 7514)     0.06809811150660568
  (0, 2420)     0.24821282829280872
  (0, 7011)     0.21112018541050012
  (0, 1540)     0.23685245608367106
  (0, 8143)     0.12153830316809083
  (0, 7062)     0.31538603933367
  (0, 2897)     0.16058878018527561
  (0, 1462)     0.24205707301347945
  (0, 1777)     0.25574686386273904
  (0, 2663)     0.09424689905578607
  :     :
  (2865, 7217)  0.2000265312588519
  (2865, 1279)  0.18172252845298603
```

```
  (2865, 4293)	0.18172252845298603
  (2865, 1308)	0.1710153731984597
  (2865, 2434)	0.12895862198915717
  (2865, 6455)	0.15888045974539744
  (2865, 4643)	0.4766413792361924
  (2865, 3467)	0.4083349271578864
  (2865, 815)	0.13386287212977616
  (2865, 3894)	0.12764891089858607
  (2865, 3538)	0.15059767508092572
  (2865, 1692)	0.1516333801429095
  (2865, 5309)	0.14864069888844816
  (2865, 4435)	0.13670508520461372
  (2865, 6675)	0.17331309081634544
  (2865, 3487)	0.10228590182314887
  (2865, 3934)	0.07802420392865589
  (2865, 7586)	0.1181026939801353
  (2865, 3430)	0.11665932413817903
  (2865, 3621)	0.14200421513806746
  (2865, 3945)	0.11156529887913935
  (2865, 4696)	0.20863643855330283
  (2865, 4987)	0.08663768441385027
  (2865, 4377)	0.13922195008914895
  (2865, 126)	0.05801999176796022
```

Cosine Similarity

```
# getting the similarity scores using cosine similarity

similarity = cosine_similarity(feature_vectors)
```

```
print(similarity)
```

```
[[1.         0.06233862 0.0340665  ... 0.00867704 0.02835473 0.00430406]
 [0.06233862 1.         0.02687679 ... 0.09525828 0.         0.00769072]
 [0.0340665  0.02687679 1.         ... 0.         0.         0.0045218 ]
 ...
 [0.00867704 0.09525828 0.         ... 1.         0.00826616 0.        ]
 [0.02835473 0.         0.         ... 0.00826616 1.         0.13818532]
 [0.00430406 0.00769072 0.0045218  ... 0.         0.13818532 1.        ]]
```

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

```
(2866, 2866)
```

Getting the movie name from the user

```
# getting the movie name from the user

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

```
 Enter your favourite movie name : Batman
```

```
# 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', 'A
```

```
# 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)
```

```
['Batman', 'Catwoman', 'Catwoman']
```

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

```
Batman
```

```
# 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)
```

```
1359
```

```
# getting a list of similar movies

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

```
[(0, 0.011852582301577534), (1, 0.12492215685455316), (2, 0.0039214815032333475), (3, 0.0042860245238468975), (4, 0.03989485320753945),
```

```
len(similarity_score)
```

```
2866
```

```
# 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)
```

```
[(1359, 1.0), (1835, 1.0), (402, 0.1576604605019644), (738, 0.1576604605019644), (2791, 0.1448235067789417), (1691, 0.14124966612128542
```

```
# 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 . Bulletproof Monk
2 . Bulletproof Monk
3 . The Rundown
4 . The Rundown
5 . Dragonball Evolution
6 . Curse of the Golden Flower
7 . Curse of the Golden Flower
8 . Pirates of the Caribbean: At World's End
9 . Anna and the King
10 . Anna and the King
11 . The Replacement Killers
12 . Priest
13 . Priest
14 . The Hunted
15 . The Hunted
16 . Drillbit Taylor
17 . Drillbit Taylor
18 . Million Dollar Baby
19 . The Pursuit of Happyness
20 . The Pursuit of Happyness
21 . DOA: Dead or Alive
22 . Role Models
23 . The Dukes of Hazzard
24 . The Dukes of Hazzard
25 . Cop Out
26 . Cop Out
27 . American Reunion
28 . American Reunion
29 . The Hobbit: The Desolation of Smaug
```

## Movie Recommendation Sytem

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

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

1 . Avatar
2 . Guardians of the Galaxy
3 . Star Trek Into Darkness
4 . Star Trek Beyond
5 . Galaxy Quest
6 . Galaxy Quest
7 . Pocahontas
8 . Pocahontas
9 . Alien³
10 . Alien³
11 . Gravity
12 . Gravity
13 . Clash of the Titans
14 . Clash of the Titans
15 . Space Cowboys
16 . Space Cowboys
17 . Moonraker
18 . Event Horizon
19 . Event Horizon
20 . The Book of Life
21 . The Book of Life
22 . Colombiana
23 . Colombiana
24 . Shadow Conspiracy
25 . Shadow Conspiracy
26 . Terminator Salvation
27 . Star Trek
28 . Alien: Resurrection
29 . Alien: Resurrection
```

6s completed at 21:13