ASSIGNMENT14

Develop a movie recommendation model using the scikit-learn library in python. Refer dataset https://github.com/rashida048/SomeNLPProjects/blob/master/movie_dataset.csv

In [1]:
```
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

1. pandas (import pandas as pd) Handles data loading and manipulation.

Stores movie details (titles, genres, descriptions) in a DataFrame.

2. numpy (import numpy as np) Used for numerical operations.

3. CountVectorizer (from sklearn.feature_extraction.text import CountVectorizer) Converts movie descriptions or genres into numerical vectors (Bag-of-Words model).

Converts text into a matrix of token counts.

4. cosine_similarity (from sklearn.metrics.pairwise import cosine_similarity) Measures the similarity between movies based on their vectorized descriptions/genres.

Returns a similarity score between 0 (no similarity) and 1 (identical).

In [2]:
```
df = pd.read_csv("https://raw.githubusercontent.com/rashida048/Some-NLP-Projects/ma
```

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

Out[3]:

	index	budget	genres	homepage	id	keywor
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	cultu cla futu space v spa colo s
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	oce dr abu exc isla east ind tra
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy bas on no sec age sequ n
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc com cri figh terroi sec ider
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based no m medalli spa tra p

5 rows × 24 columns

In [4]:
```
df.tail()
```

Out[4]:

	index	budget	genres	homepage	id
4798	4798	220000	Action Crime Thriller	NaN	9367
4799	4799	9000	Comedy Romance	NaN	72766
4800	4800	0	Comedy Drama Romance TV Movie	http://www.hallmarkchannel.com/signedsealeddel...	231617
4801	4801	0	NaN	http://shanghaicalling.com/	126186
4802	4802	0	Documentary	NaN	25975

5 rows × 24 columns

In [5]:
```
df.sample(5)
```

Out[5]:

	index	budget	genres	homepage	id	keywords	original_language
4383	4383	1000000	Documentary	NaN	39183	new york beckenbauer pele	en
1954	1954	25000000	History Action Drama	NaN	33157	biography napoleon bonaparte waterloo	en
3897	3897	3000000	Comedy	NaN	20337	daily life scandal growing up divorce	en
4549	4549	0	Action Drama Thriller	NaN	253626	pilot suspicion drone u.s. military air force ...	en
1595	1595	0	Drama	NaN	9918	basketball racial segregation teachers and stu...	en

5 rows × 24 columns

In [6]:
```
features = ['keywords','cast','genres','director']
```

This function is used in a content-based recommendation system to combine multiple movie attributes (features) into a single string. This combined text is then vectorized using CountVectorizer, allowing us to compute similarity between movies.

In [7]:
```
def combine_features(row):
    return row['keywords']+" "+row['cast']+" "+row['genres']+" "+row['director']
```

In [8]:
```
for feature in features:
    df[feature] = df[feature].fillna('')

df["combined_features"] = df.apply(combine_features,axis=1)
```

This code converts the combined text features of movies into a numerical matrix using the Bag-of-Words (BoW) model

In [9]:
```
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])
```

This computes the similarity between movies based on their combined features.

In [10]:
```
cosine_sim = cosine_similarity(count_matrix)
```

In [11]:
```
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]
def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
```

In [12]:
```
movie_user_likes = "Avatar"
movie_index = get_index_from_title(movie_user_likes)
similar_movies = list(enumerate(cosine_sim[movie_index]))
```

In [13]:
```
sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)[1:]
```

This code prints the top 5 recommended movies based on the similarity to the movie the user likes .

In [14]:
```
i=0
print("Top 5 similar movies to "+movie_user_likes+" are:\n")
for element in sorted_similar_movies:
    print(get_title_from_index(element[0]))
    i=i+1
    if i>5:
        break
```

```
Top 5 similar movies to Avatar are:

Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
Alien
```