Movie Recommender System

In [1]:

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")
```

Lode Dataset

In [2]:

```
movies = pd.read_csv('data/tmdb_5000_movies.csv')
credits = pd.read_csv('data/tmdb_5000_credits.csv')
```

In [3]:

```
movies.head(2)
```

Out[3]:

	budget	genres	homepage	id	keywords	origin
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	

In [4]:

```
credits.head()
```

Out[4]:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

EDA

In [5]:

```
print(f'movies shape {movies.shape}')
print(f'credits shape {credits.shape}')
```

```
movies shape (4803, 20)
credits shape (4803, 4)
```

Merge credits dataset with movies

In [6]:

```
movies = movies.merge(credits, on='title')
print(f'movies shape {movies.shape}')
```

```
movies shape (4809, 23)
```

Take important columns from movies dataset

In [7]:

```
# Keeping important features from dataset
movies = movies[['movie_id','title','overview','genres'
                 ,'keywords','cast','crew']]
```

In [8]:

```
movies.head(2)
```

Out[8]:

	movie_id	title	overview	genres	keywords	cast	
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"c "52fe48009251416c75
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"c "52fe4232c3a36847f8

Check null values in dataset

In [9]:

```
movies.isnull().sum()
```

Out[9]:

```
movie_id    0
title       0
overview    3
genres      0
keywords    0
cast        0
crew        0
dtype: int64
```

- **as we can see only three null values are there in overview**
- **we can remove theree records from dataset**

In [10]:

```
movies.dropna(inplace=True)
```

In [11]:

```
movies.isnull().sum()
```

Out[11]:

```
movie_id    0
title       0
overview    0
genres      0
keywords    0
cast        0
crew        0
dtype: int64
```

now there is no null values in data frame

In [12]:

```
print(f'movies shape {movies.shape}')
```

```
movies shape (4806, 7)
```

check duplicate records are there or not

In [13]:

```
movies.duplicated().sum()
```

Out[13]:

```
0
```

There is no duplicate records in dataset

In [14]:

```
# handle genres

movies.iloc[0]['genres']
```

Out[14]:

```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 1
4, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]'
```

In [15]:

```
import ast #for converting str to list


'''
as we can see there are lot of dictionary in dataset
Create function to remove annassery data frome dataset
'''

def convert(text):
    L = []
    for i in ast.literal_eval(text):
        L.append(i['name'])
    return L
```

In [16]:

```
movies['genres'] = movies['genres'].apply(convert)
```

In [17]:

```
movies['keywords'] = movies['keywords'].apply(convert)
```

In [18]:

```
def convert_cast(text):
    L = []
    counter = 0
    for i in ast.literal_eval(text):
        if counter < 3:
            L.append(i['name'])
        counter+=1
    return L
```

In [19]:

```
movies['cast'] = movies['cast'].apply(convert_cast)
```

In [20]:

```
def  fetch_director(text):
    L = []
    for i in ast.literal_eval(text):
        if i['job'] == 'Director':
            L.append(i['name'])
    return L
```

In [21]:

```
movies['crew'] = movies['crew'].apply(fetch_director)
```

In [22]:

```
movies.head()
```

Out[22]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

In [23]:

```
movies['overview'] = movies['overview'].apply(lambda x : x.split())
```

In [24]:

```
# now removing space like that
'Anna Kendrick'
'AnnaKendrick'

def remove_space(L):
    L1 = []
    for i in L:
        L1.append(i.replace(" ", ""))
    return L1
```

In [25]:

```
movies['cast'] = movies['cast'].apply(remove_space)
movies['crew'] = movies['crew'].apply(remove_space)
movies['genres'] = movies['genres'].apply(remove_space)
movies['keywords'] = movies['keywords'].apply(remove_space)
```

Create new columns called tages using concating all columns and drop all the columns

In [26]:

```
# Concatinate all
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['ca
```

In [27]:

```
# droping those extra columns
new_df = movies[['movie_id','title','tags']]
```

In [28]:

```
new_df.head()
```

Out[28]:

	movie_id	title	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond’s, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...

In [29]:

```
# Converting list to str
new_df['tags'] = new_df['tags'].apply(lambda x:" ".join(x))
```

In [30]:

```
# Converting to lower case
new_df['tags'] = new_df['tags'].apply(lambda x: x.lower())
```

In [31]:

```
new_df.iloc[0]['tags']
```

Out[31]:

```
'in the 22nd century, a paraplegic marine is dispatched to the moon pando
ra on a unique mission, but becomes torn between following orders and pro
tecting an alien civilization. action adventure fantasy sciencefiction cu
ltureclash future spacewar spacecolony society spacetravel futuristic rom
ance space alien tribe alienplanet cgi marine soldier battle loveaffair a
ntiwar powerrelations mindandsoul 3d samworthington zoesaldana sigourneyw
eaver jamescameron'
```

now we will reduce words to their base forms, the feature space is simplified, making it easier for machine learning models to learn patterns and make accurate predictions using PorterStemmer

In [32]:

```
import nltk
from nltk.stem import PorterStemmer

ps = PorterStemmer()
```

In [33]:

```
def stems(text):
    t = []
    for i in text.split():
        t.append(ps.stem(i))
    return " ".join(t)
```

Create new dataframe called new_df and apply PorterStemmer

In [34]:

```
new_df['tags'] = new_df['tags'].apply(stems)
```

convert word to numeric value using CountVectorizer

In [35]:

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
```

In [36]:

```
vector = cv.fit_transform(new_df['tags']).toarray()
```

In [37]:

```
len(vector[0])
```

Out[37]:

```
5000
```

In [38]:

```
print(f'vector shape {vector.shape}')
```

```
vector shape (4806, 5000)
```

Model Building

In [39]:

```
from sklearn.metrics.pairwise import cosine_similarity

similarity = cosine_similarity(vector)
```

In [40]:

```
new_df[new_df['title'] == 'The Lego Movie'].index[0]
```

Out[40]:

```
744
```

In [41]:

```
def recommend(movie):
    index = new_df[new_df['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x
    for i in distances[1:6]:
        print(new_df.iloc[i[0]].title)
```

Predict recommend movies

In [42]:

```
recommend('The Dark Knight Rises')
```

```
The Dark Knight
Batman Returns
Batman
Batman Forever
Batman Begins
```

Save model in pickle formate so we can use in future

In [43]:

```
import pickle

pickle.dump(new_df,open('movie_list.pkl','wb'))
pickle.dump(similarity,open('similarity.pkl','wb'))
```