PROJECT REPORT II

A project report submitted in fulfillment of

the requirements for the subject

CAS702 – FOSS LAB

MSC - Semester II

By

SHRESTHA BHATTACHARJEE

(205322019)

DEPARTMENT OF COMPUTER APPLICATIONS

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPPALLI-620015

April 2023

FOSS LAB

PROJECT II ON :

MOVIE RECOMMENDATION SYSTEM

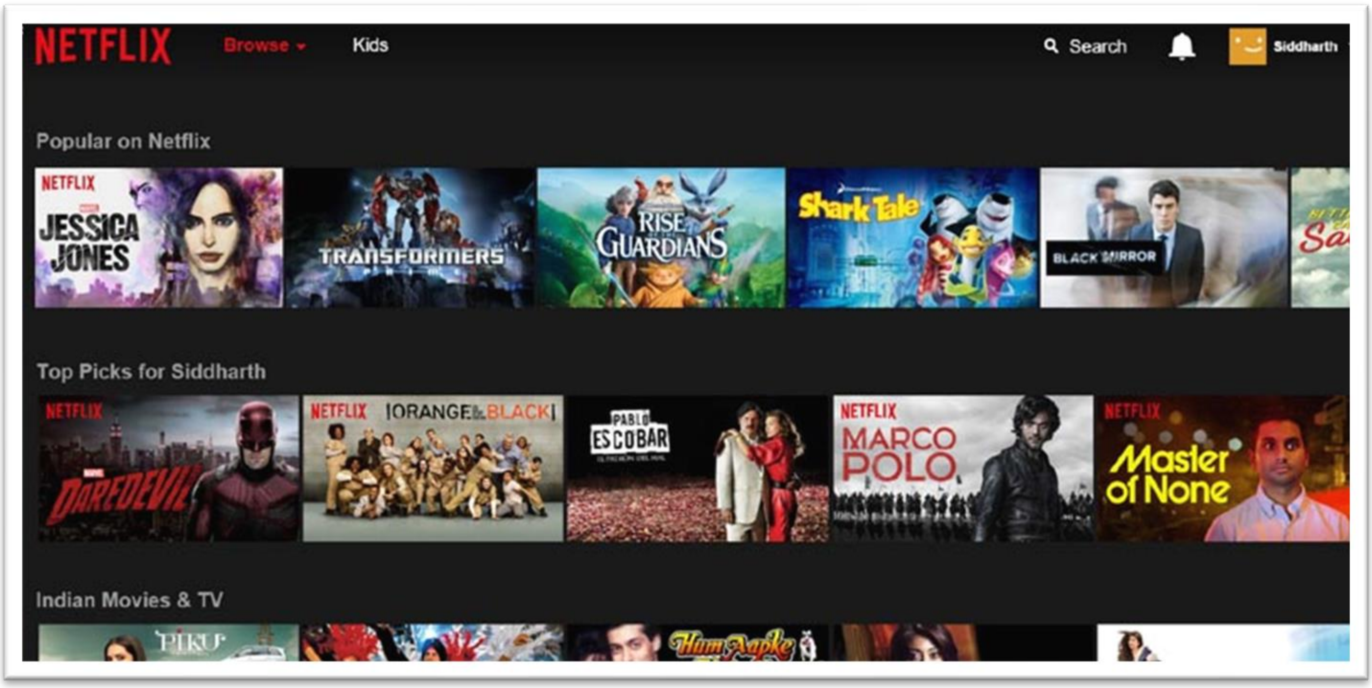
CONTENTS :

INTRODUCTION:

Recommender systems are among the most popular applications of data science today. They are used to predict the "rating" or "preference" that a user would give to an item. Almost every major tech company has applied them in some form. Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on autoplay, and Facebook uses it to recommend pages to like and people to follow.

What's more, for some companies like Netflix, Amazon Prime, Hulu, and Hotstar, the business model and its success revolves around the potency of their recommendations. Netflix even offered a million dollars in 2009 to anyone who could improve its system by 10%.

There are also popular recommender systems for domains like restaurants, movies, and online dating. Recommender systems have also been developed to explore research articles and experts, collaborators, and financial services. YouTube uses the recommendation system at a large scale to suggest you videos based on your history. For example, if you watch a lot of educational videos, it would suggest those types of videos.



Types of Movie Recommendation Systems :

Demographic filtering

Here we offer generalized recommendation to every user just based on popularity and ratings of the movie and genre. This method simply assumes that a movie is most likely interested by user if average users got interested in that one.

Content based filtering

In this system we filter based on the content of the movie. That is movies with similar plot line or we can also suggest based on similar genre, director and important crew of that movie. Here also we are assuming that user will watch movies similar to what he searched for, so it is not personalized based recommender system.

Collaborative filtering

Here when metadata is not available, we match users with similar interests to one other and recommend systems. That is, since content based filtering, is just recommending similar movies but going across various genres and based on the users taste so here we map users with similar tastes and recommend movies based on that. Here we are personalizing the movie interests.

DATASET OVERVIEW:

The data set has been downloaded from the given link :

https://drive.google.com/file/d/1cCkwiVv4mgfl20ntgY3n4yApcWqqZQe6/view

- This movies Data Set consists of 4803 rows and 24 columns.
- Each row of the 4803 rows holds the information related to a particular movie.
- With the help of this data set our aim will lie in

  Finding the best recommendations for the user when the user inputs a particular movies's name.

WORK FLOW :

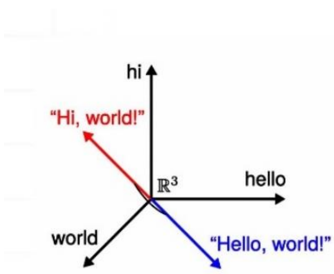
DATA

DATA

PRE PROCESSING

FEATURE

EXTRACTION



Cosine Similarity

USER INPUT

LIST OF MOVIES

DEPENDENCIES :

Importing libraries

```
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
```

- **Numpy** : A python package that provides support for multi-dimensional arrays.
- **Pandas** : It is an open source python package that is used for the data analysis or manipulation. It provides the tools like DataFrame and Series.
- **Sklearn**: It is the useful library for machine learning in python. It provides the tools for machine learning and statistical modelling.
- **Difflib**: This module provides classes and functions for comparing sequences. It can be used for example, for comparing files, and can produce information about file differences in various formats, including HTML and context and unified diffs.
- **TfidfVectorizer :** Used to convert the textual data to the numerical values (feature vectors)
- **Cosine similarity** : Cosine Similarity is a method of calculating the similarity of two vectors by taking the dot product and dividing it by the magnitudes of each vector.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

IMPLEMENTATIONS :

Data Collection and Data Pre-processing

In [2]:
```
# loading the data from the csv file to a pandas dataframe
data = pd.read_csv('movies.csv')
```

In [3]:
```
# printing the first 5 rows of the dataframe
data.head()
```

Out[3]:

	index	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	...	runtime
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...	150.437577	...	162.0
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...	139.082615	...	169.0
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...	107.376788	...	148.0
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...	112.312950	...	165.0

In [5]:
```
# selecting the relevant features for recommendation

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

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

In [6]:
```
# replacing the null valuess with null string

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

In [7]:
```
# combining all the 5 selected features

combined_features = data['genres']+' '+data['keywords']+' '+data['tagline']+' '
+data['cast']+' '+data['director']
```

In [8]:
```
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
```

CONVERTING THE TEXT DATA TO THE FEATURE VECTORS :

In [9]:
```
vectorizer = TfidfVectorizer()
```

In [10]:
```
feature_vectors = vectorizer.fit_transform(combined_features)
```

In [11]:
```
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
```

COSINE – SIMILARITY :

In [12]:
```
# getting the similarity scores using cosine similarity

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.shape)
```

```
(4803, 4803)
```

Getting the movie name from the users

```
]: # getting the movie name from the user

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

```
 Enter your favourite movie name : Jurassic Park
```

```
]: # creating a list with all the movie names given in the dataset

list_of_all_titles = data['title'].tolist()
print(list_of_all_titles)
```

```
['Avatar', "Pirates of the Caribbean: At World's End", 'Spectre', 'The Dark Knight Rises', 'John Carte
ngled', 'Avengers: Age of Ultron', 'Harry Potter and the Half-Blood Prince', 'Batman v Superman: Dawn
Returns', 'Quantum of Solace', "Pirates of the Caribbean: Dead Man's Chest", 'The Lone Ranger', 'Man o
es of Narnia: Prince Caspian', 'The Avengers', 'Pirates of the Caribbean: On Stranger Tides', 'Men in
The Battle of the Five Armies', 'The Amazing Spider-Man', 'Robin Hood', 'The Hobbit: The Desolation of
mpass', 'King Kong', 'Titanic', 'Captain America: Civil War', 'Battleship', 'Jurassic World', 'Skyfall
n Man 3', 'Alice in Wonderland', 'X-Men: The Last Stand', 'Monsters University', 'Transformers: Reveng
sformers: Age of Extinction', 'Oz: The Great and Powerful', 'The Amazing Spider-Man 2', 'TRON: Legacy'
ern', 'Toy Story 3', 'Terminator Salvation', 'Furious 7', 'World War Z', 'X-Men: Days of Future Past',
ss', 'Jack the Giant Slayer', 'The Great Gatsby', 'Prince of Persia: The Sands of Time', 'Pacific Rim'
f the Moon', 'Indiana Jones and the Kingdom of the Crystal Skull', 'The Good Dinosaur', 'Brave', 'Star
E', 'Rush Hour 3', '2012', 'A Christmas Carol', 'Jupiter Ascending', 'The Legend of Tarzan', 'The Chro
ion, the Witch and the Wardrobe', 'X-Men: Apocalypse', 'The Dark Knight', 'Up', 'Monsters vs Aliens',
ld Wild West', 'The Mummy: Tomb of the Dragon Emperor', 'Suicide Squad', 'Evan Almighty', 'Edge of Tom
'G.I. Joe: The Rise of Cobra', 'Inside Out', 'The Jungle Book', 'Iron Man 2', 'Snow White and the Hunt
awn of the Planet of the Apes', 'The Lovers', '47 Ronin', 'Captain America: The Winter Soldier', 'Shre
rrowland', 'Big Hero 6', 'Wreck-It Ralph', 'The Polar Express', 'Independence Day: Resurgence', 'How t
'Terminator 3: Rise of the Machines', 'Guardians of the Galaxy', 'Interstellar', 'Inception', 'Shin Go
n Unexpected Journey', 'The Fast and the Furious', 'The Curious Case of Benjamin Button', 'X-Men: Firs
```

```
In [29]: # 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)
```

```
['Jurassic Park', 'Jurassic Park III', 'Jurassic World']
```

```
In [30]: close_match = find_close_match[0]
print(close_match)
```

```
Jurassic Park
```

```
In [31]: # finding the index of the movie with title

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

```
675
```

```
In [32]: # getting a list of similar movies

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

```
[(0, 0.06153287404767126), (1, 0.10515988647647775), (2, 0.03741258077508364 4), (3, 0.005292964862215528), (4, 0.
65081), (5, 0.042791063241028174), (6, 0.015510017604065693), (7, 0.035882471791977524), (8, 0.015827297252339 5),
1572001641456), (10, 0.045316678601410934), (11, 0.013612791106472314), (12, 0.09285492693450617), (13, 0.0184964
4), (14, 0.0400272033594872), (15, 0.014630336037803796), (16, 0.03402421079389001), (17, 0.014967509213119633),
76377169 2615), (19, 0.0353519586148899), (20, 0.01998082493664874), (21, 0.012860364093062184), (22, 0.0325367265
(23, 0.03981770556932683), (24, 0.06275633915482347), (25, 0.0), (26, 0.03703383145551456), (27, 0.0367019460576
0.228547730594 15237), (29, 0.03825881693803299), (30, 0.04763955666366072), (31, 0.04425323823484437), (32, 0.024
244), (33, 0.035659523237 50867), (34, 0.0), (35, 0.037433577719076655), (36, 0.033730640792252964), (37, 0.049701
7), (38, 0.0156389929031746 8), (39, 0.061429130120352046), (40, 0.02125878093451468), (41, 0.04455415557671798),
```

In [33]:
```python
len(similarity_score)
```

Out[33]:
```
4803
```

In [34]:
```python
# 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)
```

```
[(675, 1.0), (508, 0.5332976711190278), (334, 0.3822696390233404), (28, 0.22854773059415237)
1, 0.13909970924754214), (1259, 0.1339279253375984), (479, 0.13231309368084204), (2805, 0.12
64381694999), (2296, 0.1164062999355315), (1999, 0.11594497823704528), (2929, 0.114474125873
3), (2128, 0.11098619593859205), (2809, 0.11056652664140285), (363, 0.10843199076205949), (5
0798010867721915), (1331, 0.10526109061325896), (1, 0.10515988647647775), (507, 0.1046575942
2), (4332, 0.10170359461020914), (572, 0.10063084035155911), (384, 0.09816328761835266), (18
09360837113390841), (12, 0.09285492693450617), (3753, 0.09190752819564361), (2163, 0.0915459
620897), (2838, 0.09087287050153421), (770, 0.09034426814747631), (375, 0.09009678115388704)
351, 0.08899120337969034), (340, 0.08892021195842709), (3698, 0.08780129955823582), (2849, 0
4866161380894), (3616, 0.08597287632454298), (483, 0.08596047682052393), (2157, 0.0855580460
838), (199, 0.0851340930241899), (3488, 0.08489817649521779), (1435, 0.08384406201213929), (
0, 0.08186313509917485), (4211, 0.08153578516088843), (644, 0.08152893284733467), (275, 0.08
74176442923), (2085, 0.08076356911563094), (2289, 0.0801097545953252), (1006, 0.079895650182
5), (1546, 0.07964973308808969), (961, 0.07924364666682096), (3130, 0.0787773192507216), (13
0.0779845130790397), (175, 0.07775347922879063), (1687, 0.077634120767682), (4199, 0.0776066
754606), (4399, 0.07691107119319168), (1691, 0.07690861915637404), (1352, 0.0765005494509883
(1187, 0.07627436699490961), (2077, 0.0749540495551866), (495, 0.0747639111331035), (3633, 0
31666187581978), (2783, 0.07384009947958989), (1213, 0.07345072447831655), (2851, 0.07312654
03334), (57, 0.0725543918328409), (789, 0.07218896938033634), (1161, 0.07182507040471331), (
```

In [35]:
```python
# 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 = data[data.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

```
Movies suggested for you :

1 . Jurassic Park
2 . The Lost World: Jurassic Park
3 . Jurassic Park III
4 . Jurassic World
5 . E.T. the Extra-Terrestrial
6 . Independence Day: Resurgence
7 . Memoirs of an Invisible Man
8 . Walking With Dinosaurs
9 . The Land Before Time
10 . Close Encounters of the Third Kind
11 . The Bounty
12 . The Adventurer: The Curse of the Midas Box
13 . History of the World: Part I
14 . The Helix... Loaded
15 . Man of the Year
16 . Jaws
```

CONCLUSIONS:

```
 Enter your favourite movie name : Jurassic Park
Movies suggested for you :

1 . Jurassic Park
2 . The Lost World: Jurassic Park
3 . Jurassic Park III
4 . Jurassic World
5 . Walking With Dinosaurs
6 . The Land Before Time
7 . The Good Dinosaur
8 . The Bounty
9 . History of the World: Part I
10 . Journey to Saturn
11 . Pirates of the Caribbean: At World's End
12 . Nim's Island
13 . Delgo
14 . Return to the Blue Lagoon
15 . Space Battleship Yamato
16 . Pirates of the Caribbean: Dead Man's Chest
17 . Cast Away
18 . Hard to Be a God
19 . Mississippi Mermaid
20 . Krrish
21 . Rapa Nui
22 . The Man with the Golden Gun
23 . E.T. the Extra-Terrestrial
24 . Rockaway
25 . Rotor DR1
26 . Species
27 . The Beach
28 . The Helix... Loaded
29 . Vessel
```

When the user entered the movie of their choice
They got 29 different movie suggestions that were related to their current search.