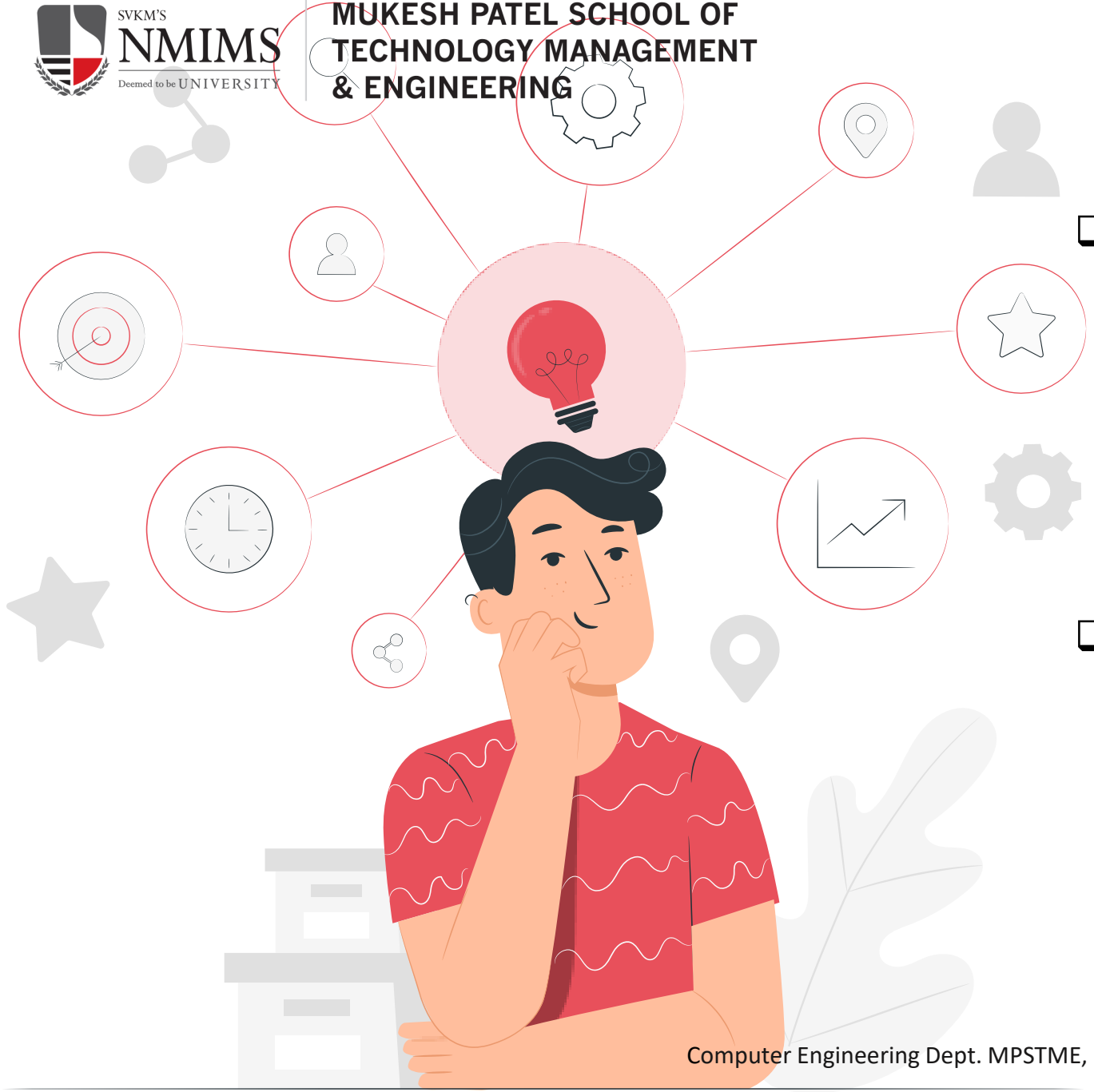
SVKM'S NMIMS
Deemed to be UNIVERSITY
MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

MOVIE RECOMMENDATION SYSTEM IN PYTHON USING MACHINE LEARNING

OUTLINE

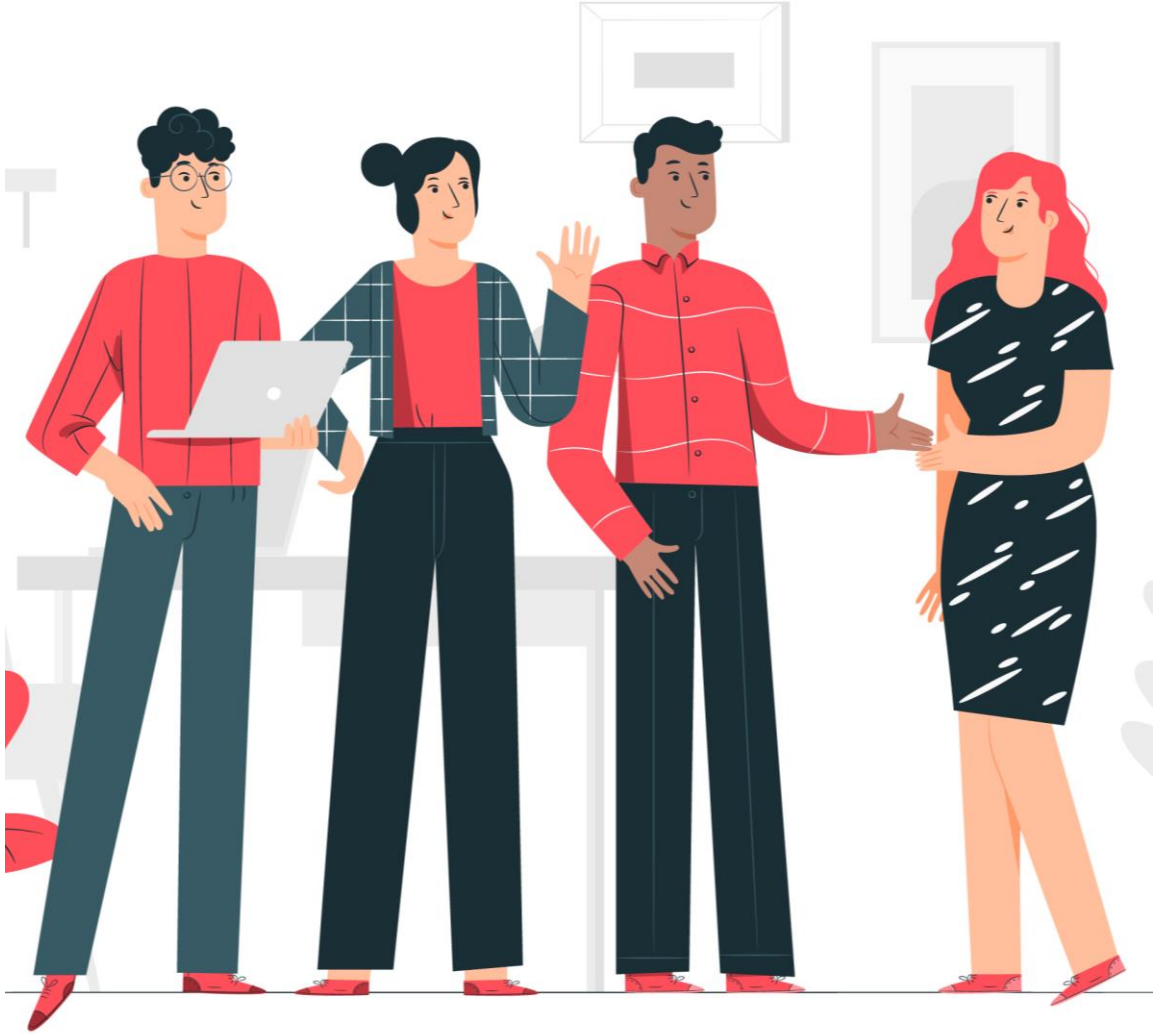
- Motivation
- Introduction
- Tools & Techniques
- Simulation & Results
- Conclusion
- Future Work
- Reference

MOTIVATION



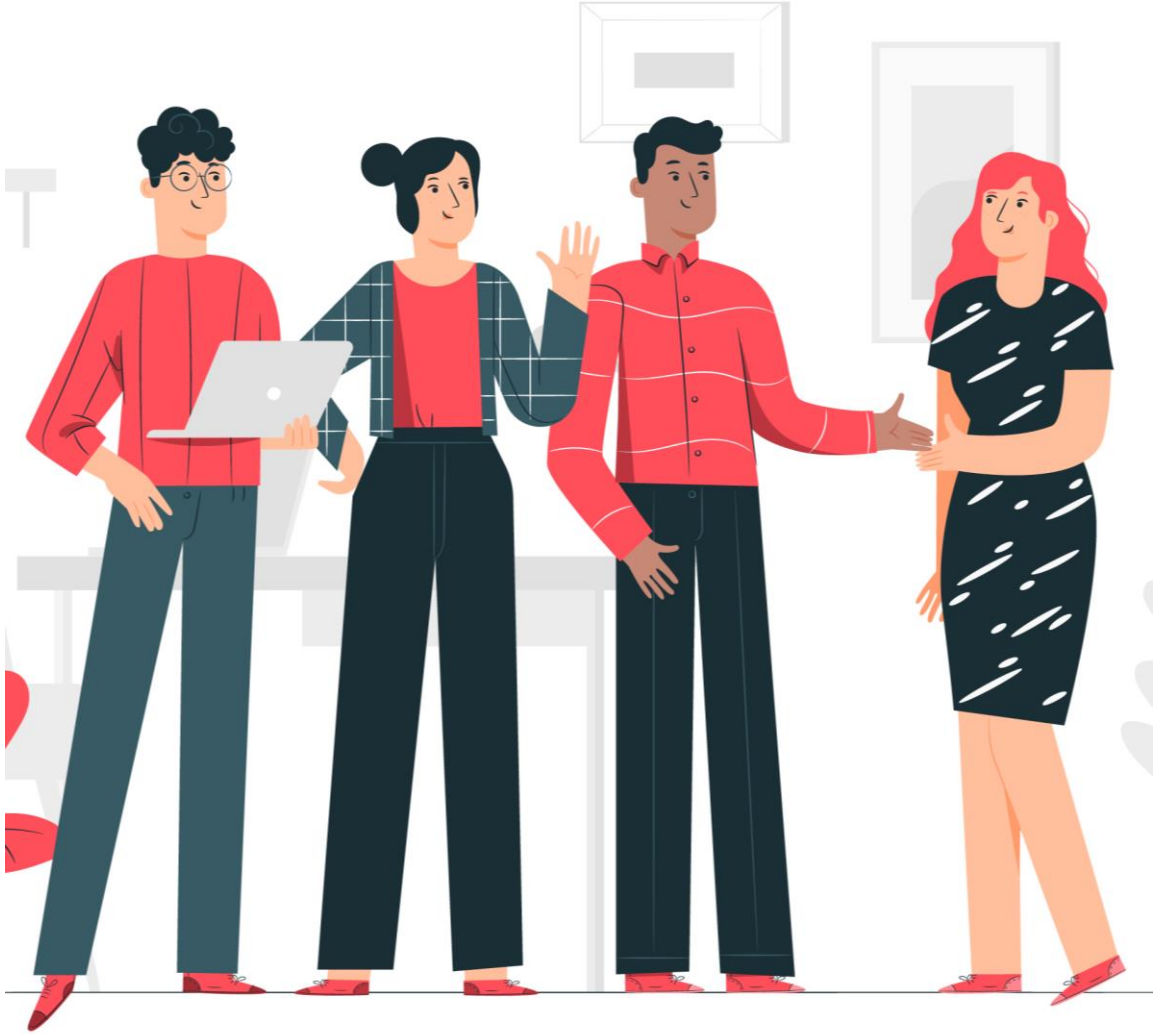
- In this current situation where there have been emerging economic crises across the entire world due to the Coronavirus have led to the shutdown of Theatres and people have shifted their focus to OTT (Over the top) platforms.
- Having ample choices it made difficult to decide and often lead to frustration and not watching anything hence this movie reccomendation could really help viewers to continue use the platforms

INTRODUCTION



- What is a Recommendation System?
- There are 3 types of recommendation systems.
- **Demographic Filtering**: They are generalized, not personalized. "Top Trending".
- **Content-based Filtering:** These suggest recommendations
- **Collaboration-based Filtering:** These systems make recommendations by grouping the users with similar interests

INTRODUCTION



- In this project, we are building a Content-based recommendation engine for movies.
- The approach to build the movie recommendation engine consists of the following steps.
- Perform Exploratory Data Analysis (EDA) on the data
- Build the recommendation system
- Get recommendations

TOOLS AND TECHNIQUES

- We have used GoogleColab as we were already familiar with it .
- Colaboratory, or "Colab" for short, is a free product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis and education.
- We chose Colab as it allows to use and share Jupyter notebooks with others without having to download, install, or run anything.

TOOLS AND TECHNIQUES

- Tools and Libraries used:
- o Python – 3.x
- o Pandas – 1.2.4
- o Scikit-learn – 0.24.1

SIMULATION AND RESULT

EXPLORATORY DATA ANALYSIS(EDA)

Loading the movie and credits dataset as data frames using pandas.

```
import pandas as pd
import numpy as np
from ast import literal_eval
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

```
[3] tmdb_5000_credits_dataframe = pd.read_csv("tmdb_5000_credits.csv")
    tmdb_5000_movies_dataframe = pd.read_csv("tmdb_5000_movies.csv")
```

The dataset contains two CSV files : credits and movies.

EXPLORATORY DATA ANALYSIS(EDA)

```
[5] tmdb_5000_movies_dataframe.head(2)
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...

Snippet of movies dataset as a data frame

Snippet of credits dataset as a data frame

```
tmdb_5000_credits_dataframe.head(2)
```

	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...

BUILD THE MOVIE RECOMMENDER SYSTEM

```
tmdb_5000_credits_dataframe.columns = ['id','title','cast','crew']
```

+ Code + Text

```
[9] tmdb_5000_movies_dataframe = tmdb_5000_movies_dataframe.merge(tmdb_5000_credits_dataframe, on="id")
```

```
[10] tmdb_5000_movies_dataframe.head(2)
```

	budget	genres	homepage	id	keywords	original_language	overview	popularity	production_companies	production_countries
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "US", "name": "United States o...
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": "US", "name": "United States o...

Merging the credits and movies data frame into a single data frame using “id” column as index

Application of literal_eval to object datatype columns in the data

```
[13] eval_column_list = ["cast", "crew", "keywords", "genres"]

     for i in eval_column_list:
         tmdb_5000_movies_dataframe[i] = tmdb_5000_movies_dataframe[i].apply(literal_eval)
```

```
[14] tmdb_5000_movies_dataframe[eval_column_list].head(2)
```

	cast	crew	keywords	genres
0	[{'cast_id': 242, 'character': 'Jake Sully', '...	[{'credit_id': '52fe48009251416c750aca23', 'de...	[{'id': 1463, 'name': 'culture clash'}, {'id':...	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...
1	[{'cast_id': 4, 'character': 'Captain Jack Spa...	[{'credit_id': '52fe4232c3a36847f800b579', 'de...	[{'id': 270, 'name': 'ocean'}, {'id': 726, 'na...	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...

BUILD THE MOVIE RECOMMENDER SYSTEM

```
[15] def get_director(x):
         for i in x:
             if i["job"] == "Director":
                 return i["name"]
         return np.nan
```

get_director is used for extracting the name of director from the dataset

```
[16] def get_list(x):
         if isinstance(x, list):
             names = [i["name"] for i in x]
             print(names)
             if len(names) > 3:
                 names = names[:3]

             return names

         return []
```

get_list is used for extracting the top 3 actors, keywords and genres for the corresponding movies

Creating a new column named director

```
[17] tmdb_5000_movies_dataframe["director"] = tmdb_5000_movies_dataframe["crew"].apply(get_director)
```

```
tmdb_5000_movies_dataframe["director"].head(5)
```

```
0        James Cameron
1       Gore Verbinski
2           Sam Mendes
3    Christopher Nolan
4       Andrew Stanton
Name: director, dtype: object
```

BUILD THE MOVIE RECOMMENDER SYSTEM

```python
eval_column_list = ["cast", "keywords", "genres"]
for i in eval_column_list:
    tmdb_5000_movies_dataframe[i] = tmdb_5000_movies_dataframe[i].apply(get_list)
```

```python
tmdb_5000_movies_dataframe["cast"].head()
```

```
0    [Sam Worthington, Zoe Saldana, Sigourney Weaver]
1       [Johnny Depp, Orlando Bloom, Keira Knightley]
2        [Daniel Craig, Christoph Waltz, Léa Seydoux]
3        [Christian Bale, Michael Caine, Gary Oldman]
4      [Taylor Kitsch, Lynn Collins, Samantha Morton]
Name: cast, dtype: object
```

```python
tmdb_5000_movies_dataframe["keywords"].head()
```

```
0       [culture clash, future, space war]
1       [ocean, drug abuse, exotic island]
2      [spy, based on novel, secret agent]
3    [dc comics, crime fighter, terrorist]
4        [based on novel, mars, medallion]
Name: keywords, dtype: object
```

```python
tmdb_5000_movies_dataframe["genres"].head()
```

```
0            [Action, Adventure, Fantasy]
1            [Adventure, Fantasy, Action]
2              [Action, Adventure, Crime]
3                  [Action, Crime, Drama]
4    [Action, Adventure, Science Fiction]
Name: genres, dtype: object
```

Application of get_list function for cast, keywords and genres columns

```
tmdb_5000_movies_dataframe[['title_x',"cast", "keywords", "genres"]].head(5)
```

	title_x	cast	keywords	genres
0	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley]	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[spy, based on novel, secret agent]	[Action, Adventure, Crime]
3	The Dark Knight Rises	[Christian Bale, Michael Caine, Gary Oldman]	[dc comics, crime fighter, terrorist]	[Action, Crime, Drama]
4	John Carter	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[based on novel, mars, medallion]	[Action, Adventure, Science Fiction]

Clean_data function is used to clean the data present in the data frame

```
[22] def clean_data(row):
         if isinstance(row, list):
             return [str.lower(i.replace(" ", "")) for i in row]
         else:
             if isinstance(row, str):
                 return str.lower(row.replace(" ", ""))
             else:
                 return ""
```

```
[23] features = ['cast', 'keywords', 'director', 'genres']
     for feature in features:
         tmdb_5000_movies_dataframe[feature] = tmdb_5000_movies_dataframe[feature].apply(clean_data)
```

```
[24] tmdb_5000_movies_dataframe[['title_x', 'cast', 'director', 'keywords', 'genres']].head(2)
```

	title_x	cast	director	keywords	genres
0	Avatar	[samworthington, zoesaldana, sigourneyweaver]	jamescameron	[cultureclash, future, spacewar]	[action, adventure, fantasy]
1	Pirates of the Caribbean: At World's End	[johnnydepp, orlandobloom, keiraknightley]	goreverbinski	[ocean, drugabuse, exoticisland]	[adventure, fantasy, action]

BUILD THE MOVIE RECOMMENDER SYSTEM

```
[57] def determinents(features):
         return ' '.join(features['keywords']) + ' ' + ' '.join(features['cast']) + ' ' + features['director'] + ' ' + ' '.join(features['genres'])
```

```
[58]
     tmdb_5000_movies_dataframe["determinents"] = tmdb_5000_movies_dataframe.apply(determinents, axis=1)
```

```
[59] print(tmdb_5000_movies_dataframe[["title_x","determinents"]].head(5))
```

```
                                  title_x                                      determinents
0                                  Avatar  cultureclash future spacewar samworthington zo...
1  Pirates of the Caribbean: At World's End  ocean drugabuse exoticisland johnnydepp orland...
2                                 Spectre  spy basedonnovel secretagent danielcraig chris...
3                    The Dark Knight Rises  dccomics crimefighter terrorist christianbale ...
4                              John Carter  basedonnovel mars medallion taylorkitsch lynnc...
```

Combining keywords, cast, director, and genres columns together into a single determinents column

Count vectorizer is used to encode words as integer

To use textual data for predictive modeling, words need to then be encoded as integers, or floating-point values

```
[61] count_vectorizer = CountVectorizer(stop_words="english")
     count_matrix = count_vectorizer.fit_transform(tmdb_5000_movies_dataframe["determinents"])
     print(count_matrix.shape)
```

```
(4803, 11520)
```

```
[62] cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
     print(cosine_sim2.shape)
     movies_df = tmdb_5000_movies_dataframe.reset_index(drop=True)
     indices = pd.Series(movies_df.index, index=movies_df['title_x'])
```

```
(4803, 4803)
```

BUILD THE MOVIE RECOMMENDER SYSTEM

```
[63] indices = pd.Series(movies_df.index, index=movies_df["title_x"]).drop_duplicates()
     print(indices.head())
```

```
title_x
Avatar                                      0
Pirates of the Caribbean: At World's End    1
Spectre                                     2
The Dark Knight Rises                       3
John Carter                                 4
dtype: int64
```

- Getting the list of the indices of the top 10 movies from the above sorted list. Exclude the first element because it is the title itself.

- Mapping those indices to their respective titles and return the movies list.

Creating a function (get_recommendation) that takes in the movie title and the cosine similarity as input and outputs the top 10 movies similar to it.

```
[64] def get_recommendations(title, cosine_sim=cosine_sim2):
         idx = indices[title]
         similarity_scores = list(enumerate(cosine_sim[idx]))
         similarity_scores= sorted(similarity_scores, key=lambda x: x[1], reverse=True)
         similarity_scores=  similarity_scores[1:11]
         movies_indices = [ind[0] for ind in similarity_scores]
         movies = movies_df["title_x"].iloc[movies_indices]
         return movies
```

GET RECOMMENDATIONS FOR THE MOVIES

```python
x=input("Enter Movie: -")
print("Recommendations for ",x)
print(get_recommendations(x, cosine_sim2))
print()
y=input('Enter Movie: -')
print("Recommendations for ",y)
print(get_recommendations(y, cosine_sim2))
```

```
Enter Movie: -Spider-Man 3
Recommendations for  Spider-Man 3
30                                Spider-Man 2
159                                 Spider-Man
37                 Oz: The Great and Powerful
50        Prince of Persia: The Sands of Time
71      The Mummy: Tomb of the Dragon Emperor
786                         The Monkey King 2
103                 The Sorcerer's Apprentice
131                                   G-Force
215    Fantastic 4: Rise of the Silver Surfer
715                         The Scorpion King
Name: title_x, dtype: object

Enter Movie: -Tangled
Recommendations for  Tangled
1108                                 Pinocchio
1481             Thunder and the House of Magic
3670                            Running Forever
42                                  Toy Story 3
254                                  The Smurfs
390                          Hotel Transylvania
578     Alvin and the Chipmunks: The Squeakquel
1695                                   Aladdin
1426                                   Valiant
358                   Atlantis: The Lost Empire
Name: title_x, dtype: object
```

We ask the user for the name of the movie of choice based on which recommended movies are displayed

FUTURE WORK

- In this project, we are built a Content-based recommendation engine for movies.
- We could make it better by adding Demographic filtering and Collaboration based filtering.
- Demographic Filtering: The recommendations are the same for every user. They are generalized, not personalized. These types of systems are behind sections like “Top Trending”.
- Collaboration-based Filtering: These systems make recommendations by grouping the users with similar interests. For this system, metadata of the item is not required.

CONCLUSION

- In the end we would like to conclude by saying that the code we built a content-based recommendation engine that makes recommendations given the title of the movie as input.

REFERENCE

- https://www.kaggle.com/tmdb/tmdb-movie-metadata
- https://towardsdatascience.com/machine-learning-for-building-recommender-system-in-python-9e4922dd7e97
- https://en.wikipedia.org/wiki/Collaborative_filtering


Thank You