Movie Recommendations System Using Machine Learning

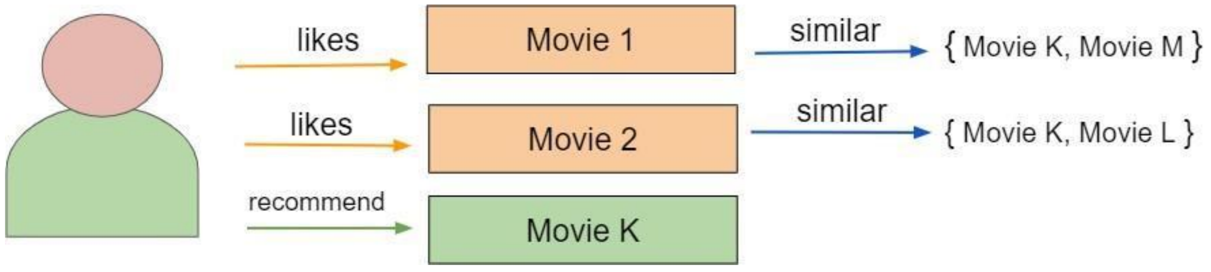
Golla Bhavana Durga

1.1 Objective:

We have discussed about a recommender system and its types . Now , our objective is to acquire more knowledge about content based filetring and to know how to built a content based movie recommender system

Content-Based Recommendations systems are the systems that look for similarity before recommending something. We all have seen whenever we are looking for a movie or web series on Netflix, we get the same genre movie recommended by Netflix. But how does this work? How does Netflix compute what I like? This is all done through content-based systems. The similarity of different movies is computed to the one you are currently watching and all the similar movies are recommended to us. In the case of e-commerce website similarity in terms of products is calculated. Considering I am looking for a MacBook then the website will look for all similar products that are similar to MacBook and straight away will recommend us.

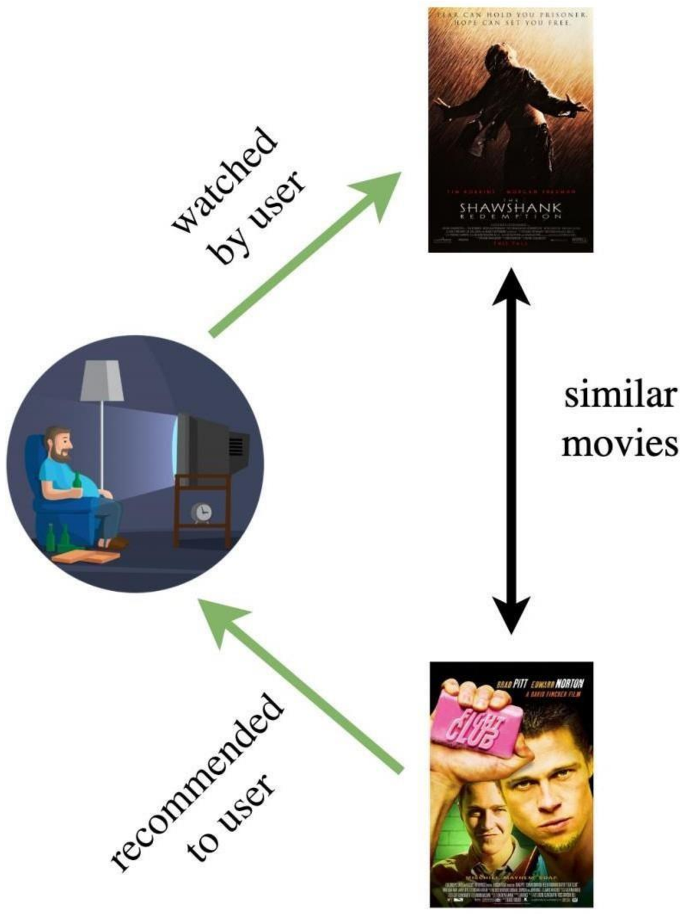
Before knowing about the overview of the project let us know the applications of content based filtering.



1.2Applications

Content-based filtering. It relies on similarities between features of the items. It recommends items to a customer based on previously rated highest items by the same customer. List of features about these items needs to be generated.

- Each item will have an item profile
- A table structure will list these properties
- Comparing what and how many features match and collect scores
- Recommend highest scored item
- Code will be based on an algorithm, by given some item, the most similar item will be found.
- Best scoring match will be provided to the user
- This method relies on item features only, and not the user preferences.



Overview of the project:

We will work with two datasets. One is Credits, and the other is Movies The dataset is taken from Kaggle, it is called TMDB data i.e. The Movie Database. We are going to build a recommender based on the following metadata: the 3 top actors, the director, related genres and the movie plot keywords. From the cast, crew and keywords features, we need to extract the three most important actors, the director and the keywords associated with that movie.

The similarity is the main key fundamental in the case of content-based recommendation systems. A most similar thing to what we are currently watching gets recommended to us. We have used cosine similarity as a simlilarity matrix.

Cosine Similarity is the type of metric is used to compute the similarity textual data.

Consider an example where we have to find similar news or similar movies. How is it done? We convert these textual data in the form of vectors and check for cosine angle between those two vectors if the angle between them is 0. It means they are similar or else they are not. Most used similarity measures when we talk about the similarity between any textual content. There are other different metrics as well like Jaccard Similarity that is used when we have categorical data.

We have used the algorithm Count Vectorizer to train model.In CountVectorizer we only count the number of times a word appears in the document which results in biasing in favour of most frequent words. this ends up in ignoring rare words which could have helped is in processing our data more efficiently.

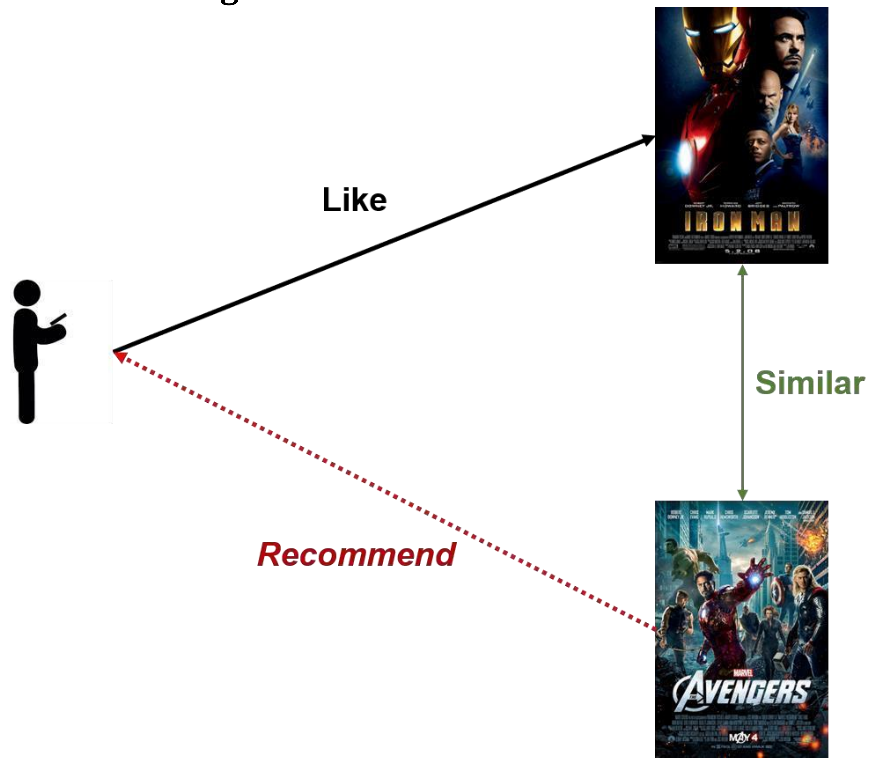
Once the model is trained we can make use of it to get recommendations and later on I have converted it as a web app using flask web framework

Get the more understanding about various similarity matrics and different machine learning algorithms in the upcoming chapters.

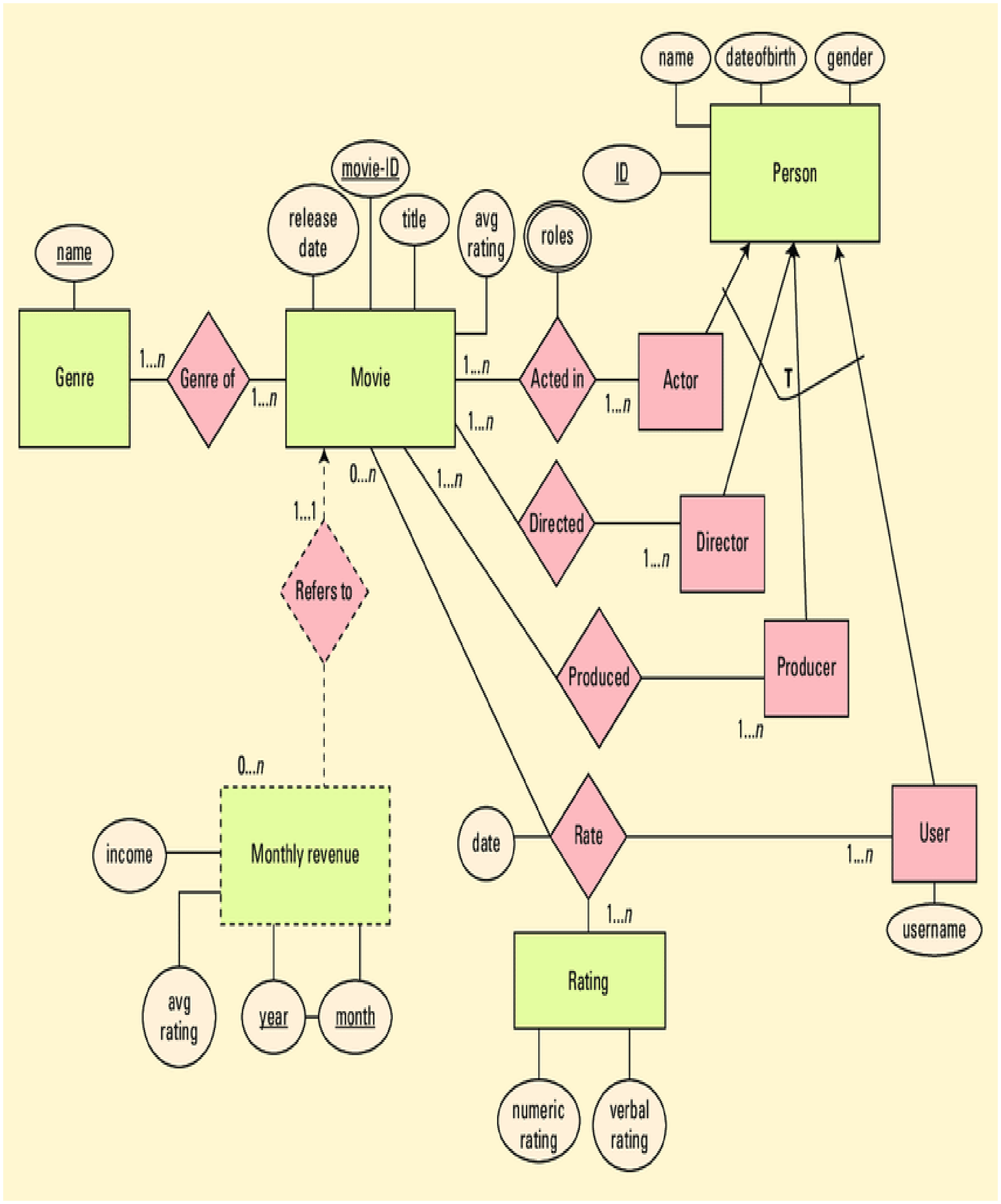
2.1 BENEFITS OF CONTENT-BASED RECOMMERDER SYSTEMS:

- **User independence:** The content-based method only has to analyze the items and a single user’s profile for the recommendation, which makes the process less cumbersome. Content-based filtering would thus produce more reliable results with fewer users in the system.
- **Transparency:** Collaborative filtering gives recommendations based on other unknown users who have the same taste as a given user, but with content-based filtering, items are recommended on a feature-level basis.
- **No cold start:** As opposed to collaborative filtering, new items can be suggested before being rated by a substantial number of users.

2.2 Use-case Diagram:



2.3 Architecture:



3. IMPLEMENTATION:

3.1 MODULE DESCRIPTION:

3.1.1 Pandas and Numpy :

NumPy provides the essential multi-dimensional array-oriented computing functionalities designed for high-level mathematical functions and scientific computation , Pandas DataFrame will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, and Excel file. Pandas DataFrame can be created from the lists, dictionary, and from a list of dictionary etc

Importing pandas and numpy

```
import numpy as np
import pandas as pd
```

3.1.2 Matplotlib Library :

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

Importing matplotlib

```
import matplotlib.pyplot as plt
```

3.1.2 Cosine Similarity:

Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified BSD license, encouraging its use in both academic and commercial settings

From scikit learn module I have imported Cosine Similarity as it performs used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space

```
from sklearn.metrics.pairwise import cosine_similarity
```

3.1.3 Count Vectorizer:

Scikit-learn’s count vectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

```
from sklearn.feature_extraction.text import CountVectorizer
```

3.2 IMPLEMENTATION DETAILS:

3.2.1 DATASET:

We will work with two datasets. One is Credits, and the other is Movies The dataset is taken from Kaggle, it is called TMDB data i.e. The Movie Database. Load the basic libraries pandas , numpy and mathplotlib , load the datasets and explore the basic details like shape of the data, null value and data types of each column. The first credit dataset contains 4803 rows and 4 columns.The second movies dataset contains 4803 rows and 20 columns, to perform further analysis we need to merge the datasets on the 'id' column.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df1 = pd.read_csv('credits.csv')
df2 = pd.read_csv('movies.csv')
df1.columns = ['id', 'title', 'cast', 'crew']
df2 = df2.merge(df1, on='id')
```

3.2.2 IMDB's weighted rating:

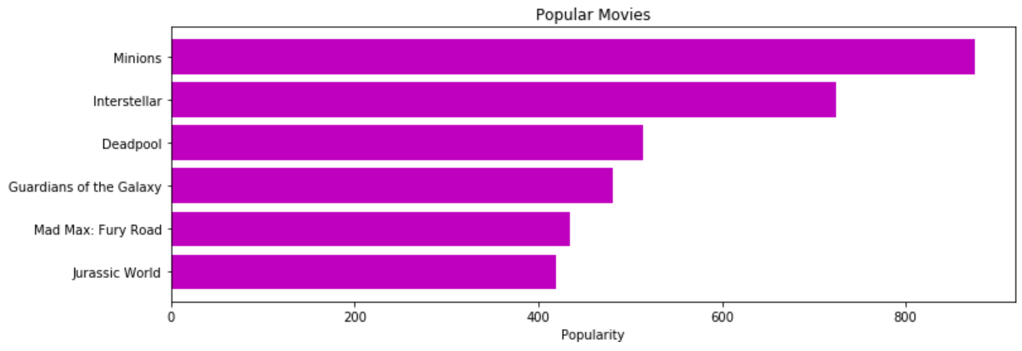
We need a metric for rating the movie so that we need to Calculate the score for every movie**.** Sort the scores and recommend the best rated movie to the users.
A simple way out to find the score is to go with the average ratings, but using this won't be fair enough since a movie with 8.7 average rating and only 8 votes cannot be considered better than the movie with 7.9 as as average rating but 40 votes. So,let us use IMDB's weighted rating (wr) which is given as :-

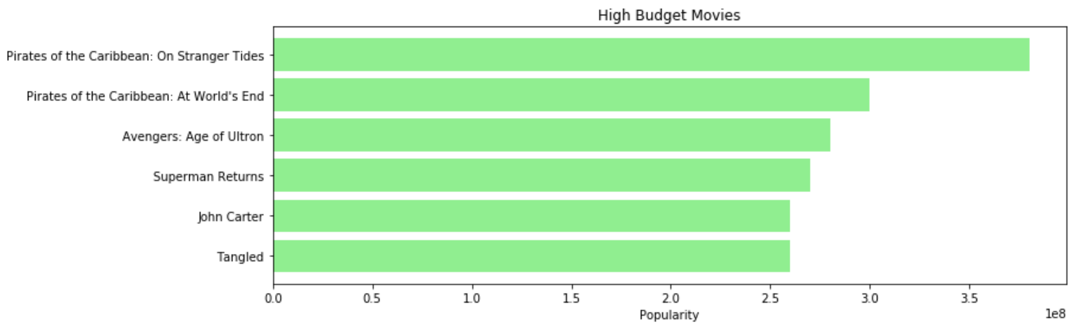
$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m} \cdot R\right) + \left(\frac{m}{v+m} \cdot C\right)$$

where,

- v is the number of votes for the movie;
- m is the minimum votes required to be listed in the chart;
- R is the average rating of the movie; and
- C is the mean vote across the whole report

With the above mentioned formula we can able to get hold of scores for each movie . By sorting the dataframe on the basis of score , we would get the list of top movies with respect to popularity, high budget etc.
By the help of matplotlib module we got the analysis of the movies with respect to popularity and budget.





3.2.3 MODEL :

After cleaning the data , I have imported Count Vectorizer and created the count matrix and then computed the cosine similarity matrix based on count matrix . With this model had been prepared . Now , by making use of this model user can get recommendations.

3.2.4 WEB APP CONVERSION:

3.2.4.1 Flask Web Framework :

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
By using flask frame work I have converted my python file to a web app .I have used HTML , CSS , BOOTSTRAP for creating the basic markup page with few stylings.

3.3 TECHNOLOGIES USED :

3.3.1 KAGGLE DATASET:

Kaggle supports a variety of dataset publication formats, but we strongly encourage dataset publishers to share their data in an accessible, non-proprietary format if possible. Not only are open, accessible data formats better supported on the platform, they are also easier to work with for more people regardless of their tools.

3.3.2 MACHINE LEARNING :

Machine learning is a subfield of soft computing within computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.

Recommender systems are used in a variety of areas, with commonly recognised examples taking the form of playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms and open web content recommenders.

3.3.3 FRAMEWORK

Flask is one of the most popular python web frameworks because of its lightweight. Although it is micro it is an extensible python web framework. By providing the required functionality, flask accelerates the development of simple web application. So Flask, is more suitable for smaller, less complicated applications.

3.3.4 TOOLS USED

- VSCODE
- JUPYTER NOTEBOOK
- KAGGLE DATASET

4. TESTING EXPERIMENT / RESULTS:

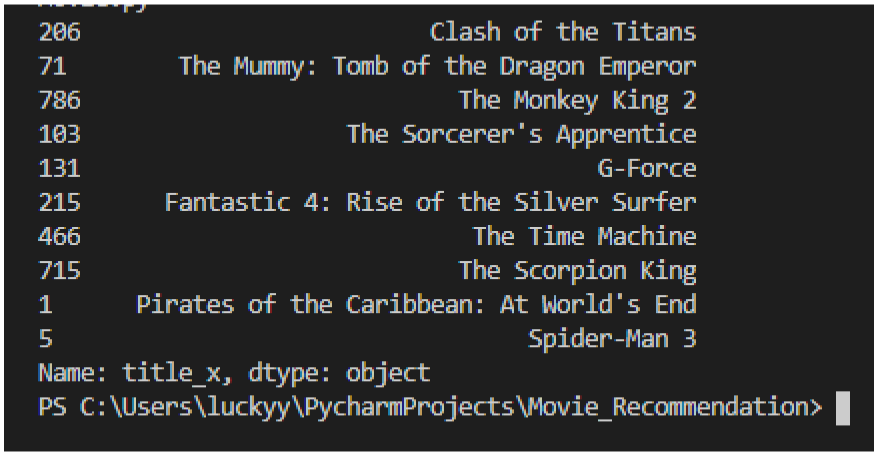
4.1 Testing :

Code for testing:

```python
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
from ast import literal_eval
import pickle
import numpy as np
import pandas as pd
df1 = pd.read_csv('credits.csv')
df2 = pd.read_csv('movies.csv')
df1.columns = ['id', 'title', 'cast', 'crew']
df2 = df2.merge(df1, on='id')
def get_recommendations(title, cosine_sim):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:11]
    movie_indices = [i[0] for i in sim_scores]
    return df2['title_x'].iloc[movie_indices]


features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(literal_eval)


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



def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        if len(names) > 3:
            names = names[:3]
        return names

    return []

df2['director'] = df2['crew'].apply(get_director)

features = ['cast', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(get_list)

df2[['title_x', 'cast', 'director', 'keywords', 'genres']].head(5)

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

features = ['cast', 'keywords', 'director', 'genres']

for feature in features:
    df2[feature] = df2[feature].apply(clean_data)

def create_soup(x):
    return ' '.join(x['keywords']) + ' ' + ' '.join(x['cast']) + ' ' + x['
director'] + ' ' + ' '.join(x['genres'])

df2['soup'] = df2.apply(create_soup, axis=1)

# Import CountVectorizer and create the count matrix
```

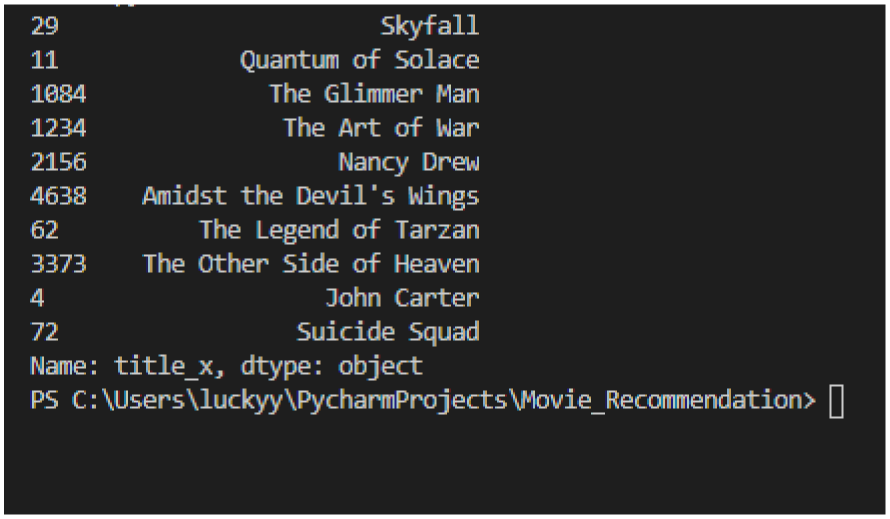
```python
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df2['soup'])

# Compute the Cosine Similarity matrix based on the count_matrix

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)

pickle.dump(cosine_sim2, open('model1.pkl', 'wb'))
model1 = pickle.load(open('model1.pkl', 'rb'))

# Reset index of our main DataFrame and construct reverse mapping as befor
e
df2 = df2.reset_index()
indices = pd.Series(df2.index, index=df2['title_x'])
print(get_recommendations("Avatar", cosine_sim2))
```

We have got 10 recommendations for user givien movie ‘Avatar’



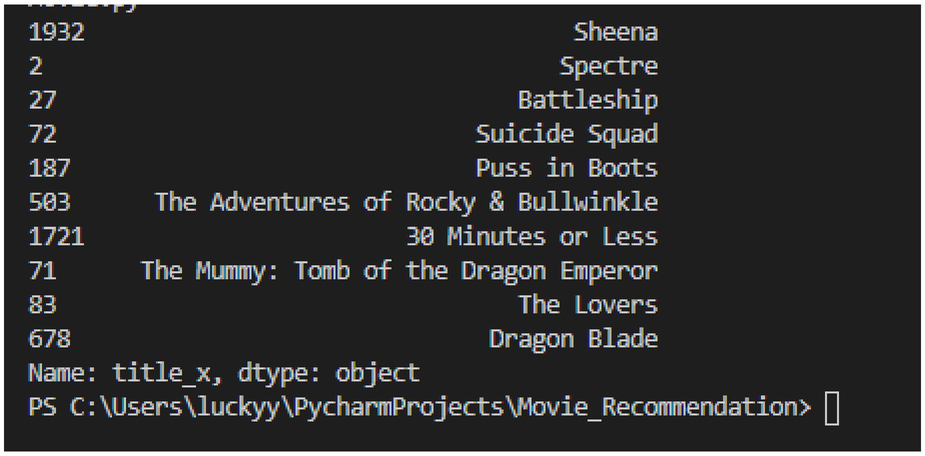
```
206                          Clash of the Titans
71          The Mummy: Tomb of the Dragon Emperor
786                            The Monkey King 2
103                      The Sorcerer's Apprentice
131                                      G-Force
215         Fantastic 4: Rise of the Silver Surfer
466                             The Time Machine
715                            The Scorpion King
1        Pirates of the Caribbean: At World's End
5                                   Spider-Man 3
Name: title_x, dtype: object
PS C:\Users\luckyy\PycharmProjects\Movie_Recommendation>
```

10 recommendations for user givien movie ‘Spectre’



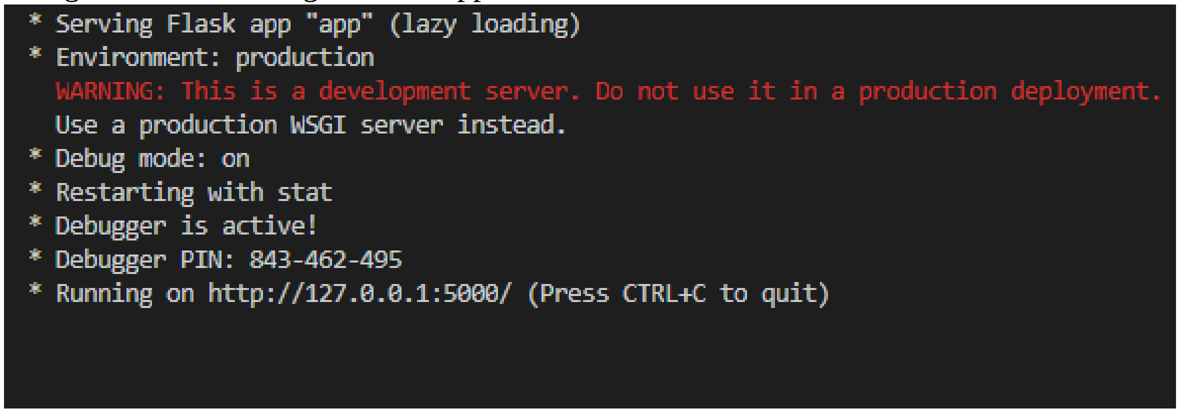
```
29                    Skyfall
11           Quantum of Solace
1084           The Glimmer Man
1234            The Art of War
2156                Nancy Drew
4638    Amidst the Devil's Wings
62          The Legend of Tarzan
3373    The Other Side of Heaven
4                  John Carter
72               Suicide Squad
Name: title_x, dtype: object
PS C:\Users\luckyy\PycharmProjects\Movie_Recommendation>
```

10 recommendation for ‘The Legend Of Tarzan’
’



```
1932                                      Sheena
2                                        Spectre
27                                    Battleship
72                                 Suicide Squad
187                                Puss in Boots
503       The Adventures of Rocky & Bullwinkle
1721                          30 Minutes or Less
71       The Mummy: Tomb of the Dragon Emperor
83                                    The Lovers
678                                 Dragon Blade
Name: title_x, dtype: object
PS C:\Users\luckyy\PycharmProjects\Movie_Recommendation>
```

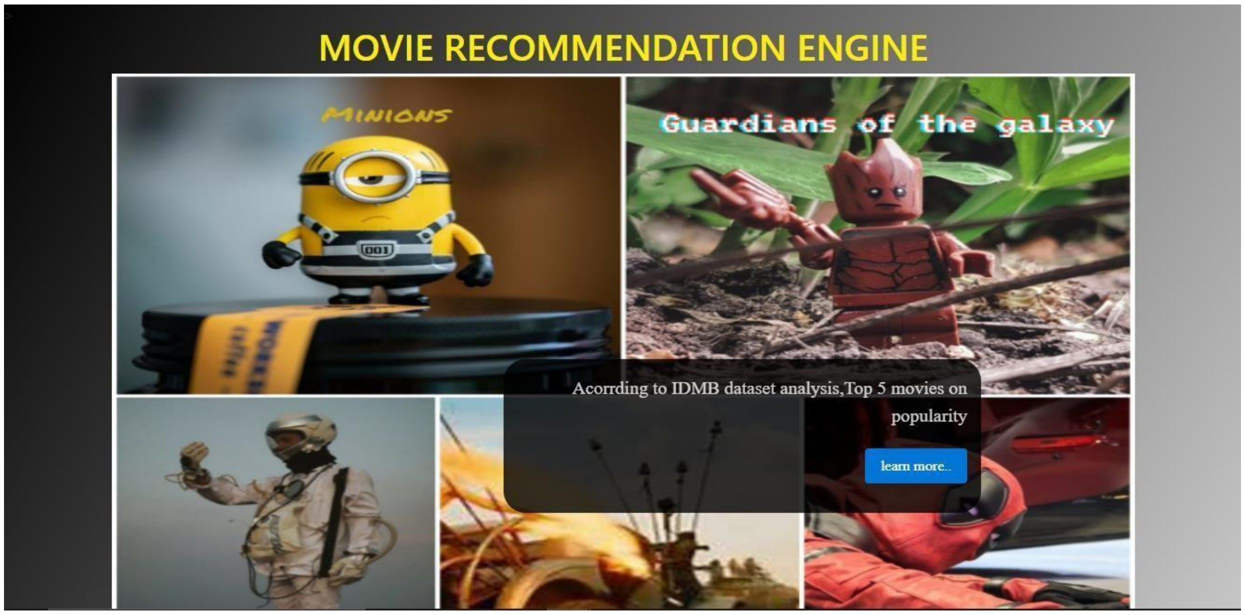
Testing after transforming as a web app



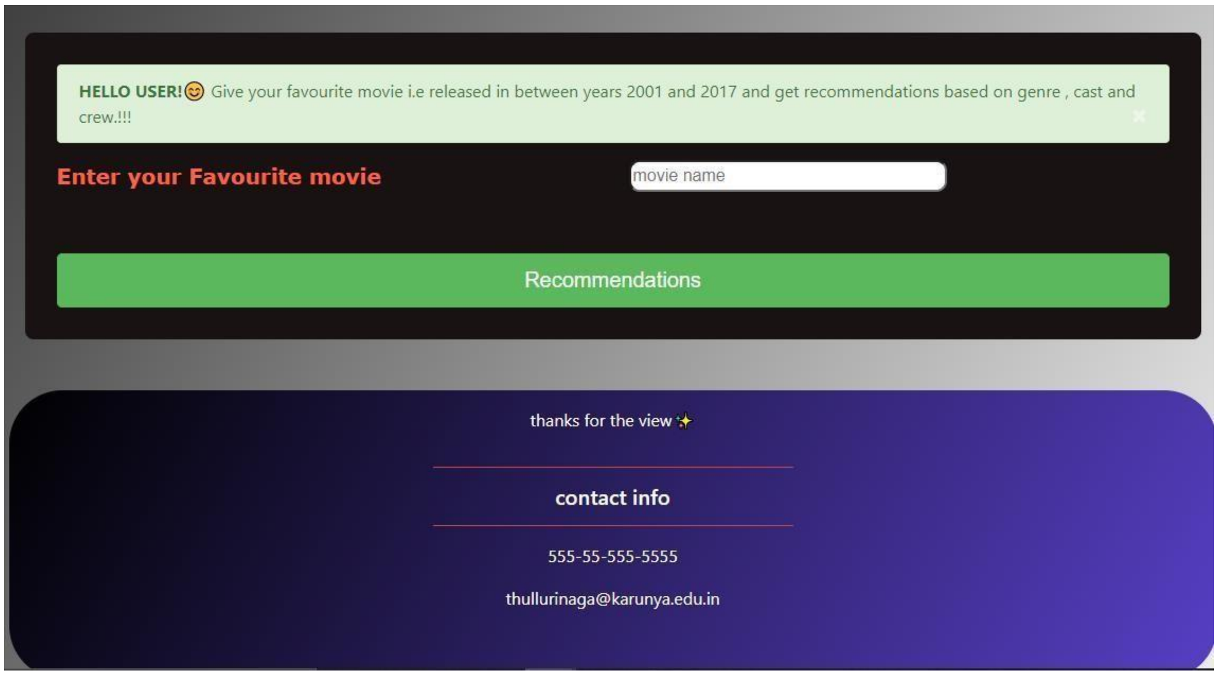
```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 843-462-495
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

4.2 Results:

Here is the final web app snippets which gives you more understanding about the final product and its working.



Acorrding to IDMB dataset analysis,Top 5 movies on popularity



HELLO USER!😊 Give your favourite movie i.e released in between years 2001 and 2017 and get recommendations based on genre , cast and crew.!!!

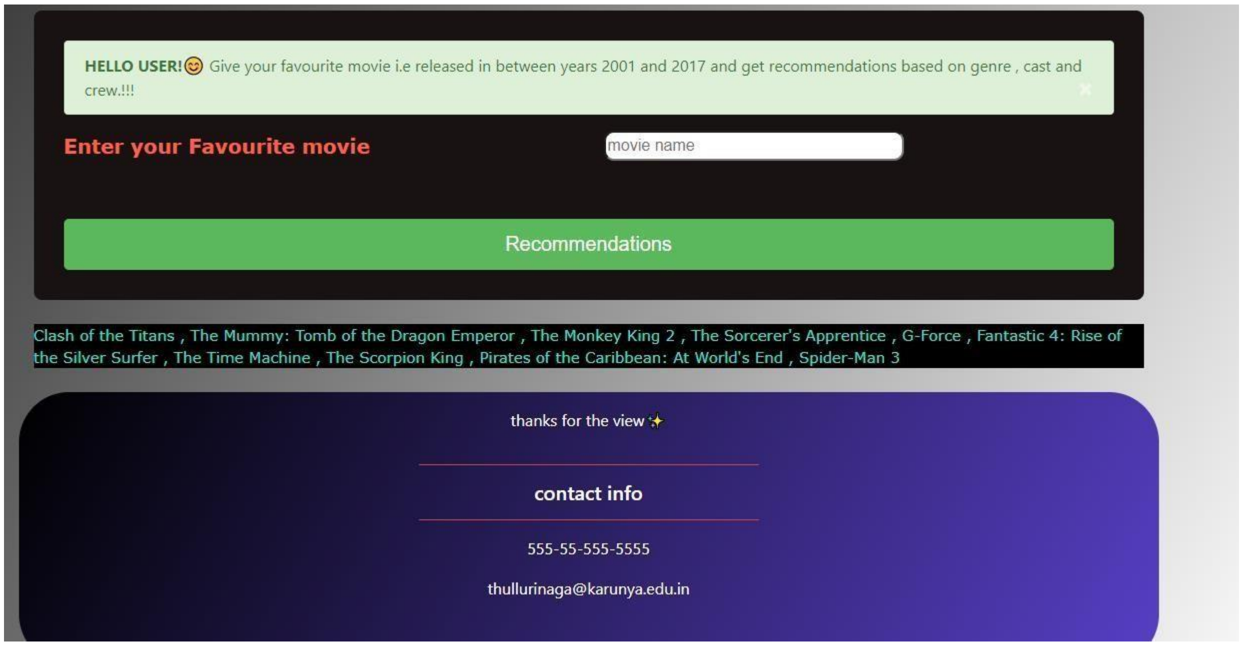
thanks for the view ✨

contact info

555-55-555-5555

thullurinaga@karunya.edu.in

4.3 Verification:



HELLO USER!😊 Give your favourite movie i.e released in between years 2001 and 2017 and get recommendations based on genre , cast and crew.!!!

Clash of the Titans , The Mummy: Tomb of the Dragon Emperor , The Monkey King 2 , The Sorcerer's Apprentice , G-Force , Fantastic 4: Rise of the Silver Surfer , The Time Machine , The Scorpion King , Pirates of the Caribbean: At World's End , Spider-Man 3

thanks for the view ✨

contact info

555-55-555-5555

thullurinaga@karunya.edu.in

We have verified the web app , it is suggesting the movies based on genre , cast and crew of the user given movie

5. Conclusion and Future Scope :

I would like to conclude this report by stating that I hope you have got an understanding of how this recommendation system works. We have mainly discussed about content-based recommendation system and further developing it as an web app. Recommendation systems are very effective systems that are tremendous. Recommendation systems provide insights that help companies build better products for their users. It helps apps anticipate the user's preferences and also recommend them new or similar products/ movies based on what the system knows about the user. It is not a perfect system, but relevant in the sense of how it interacts with users in providing them similar items of interest.

And I want to further develop this as a more user friendly app

6. References :

- **www.google.com**
- **https://www.kaggle.com/tmdb/tmdb-movie-metadata**
- **https://stackoverflow.com/**
- **https://machinemantra.in**
- **www.youtube.com**
- **https://en.wikipedia.org/**