VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A PROJECT REPORT ON

"MOVIE RECOMMENDATION SYSTEM"

Submitted in Partial fulfillment of the Requirements for the Degree of

Master of Computer Applications

Submitted by

VISHNUPRIYA P (1CR20MC116)

Under the Guidance of,
Dr.Gnaneswari G
Assistant Professor
Dept. of MCA

DEPARTMENT OF MCA

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF MCA

CELEBRATING 25 YEARS

CMRIT

CMR INSTITUTE OF TECHNOLOGY, BENGALURU.

ACCREDITED WITH A+ GRADE BY NAAC

CERTIFICATE

Certified that the project work entitled **"MOVIE RECOMMENDATION SYSTEM"** carried out by **VISHNUPRIYA P** USN **1CR20MC116,** Bonafide student of CMR Institute of Technology, in partial fulfillment for the award of **Master of Computer Applications** of the Visveswaraya Technological University, Belgaum during the year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Dr. Gnaneswari G
Assistant Professor
Dept. of MCA,
CMRIT

Ms. Gomathi T
Professor & Head Dept. of MCA,
CMRIT

DECLARATION

I, the student of MCA, CMR Institute of Technology, Bangalore declare that the work entitled "**MOVIE RECOMMENDATION SYSTEM**" has been successfully completed under the guidance of Dr. Gnaneswari G, Assistant Professor, Department of MCA, CMR Institute of Technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Master of Computer Applications during the academic year 2021 - 2022. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place:

Date:

VISHNUPRIYA P (1CR20MC116) ____________________

ACKNOWLEDGEMENT

At the various stages in making the mini project, a number of people have given me invaluable comment on the manuscript. I take this opportunity to express my deepest gratitude and appreciation to all those who helped me directly or indirectly towards the successful completion of this project.

I would like to thank **Principal Dr. Sanjay Jain, CMR Institute of Technology** for his support throughout this project.

I express mywhole hearted gratitude to **Prof. Gomati T,** who is our respectable **Head of Dept, MCA.** I wish to acknowledge for her valuable help and encouragement.

In this regard we owe a heartfelt gratitude to my guide **Dr. Gnaneswari G, Assistant Professor, Department of MCA,** for her timely advice on the mini project and regular assistance throughout the project work. I would also like to thank the staff members of Department of MCA for their corporation.

ABSTRACT

In this hustling world, entertainment is a necessity for each one of us to refresh our mood and energy. Entertainment regains our confidence for work and we can work more enthusiastically. For revitalizing ourselves, we can listen to our preferred music or can watch movies of our choice. For watching favourable movies online we can utilize movie recommendation systems, which are more reliable, since searching of preferred movies will require more and more time which one cannot afford to waste. In this paper, to improve the quality of a movie recommendation system, a Hybrid approach by combining content based filtering and collaborative filtering, using Support Vector Machine as a classifier and genetic algorithm is presented in the proposed methodology and comparative results have been shown which depicts that the proposed approach shows an improvement in the accuracy, quality and scalability of the movie recommendation system than the pure approaches in three different datasets. Hybrid approach helps to get the advantages from both the approaches as well as tries to eliminate the drawbacks of both methods.

MINI PROJECT

REPORT

MOVIE RECOMMENDATION SYSTEM

DONE BY:

TARUN S RAJU (1CR20MC106)

VISHNUPRIYA P (1CR20MC116)

INDEX

INTRODUCTION

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggests based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities. These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future.

Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films. Movies can be easily differentiated through their genres like comedy, thriller, animation, action etc. Other way to distinguish among movies can be either by releasing year, language, director etc. Watching movies online, there are a number of movies to search in our most liked movies . Movie Recommendation Systems helps us to search our preferred movies among all of these different types of movies and hence reduce the trouble of spending a lot of time searching our favourable movies. So, it requires that the movie recommendation system should be very reliable and should provide us with the recommendation of movies which are exactly same or most matched with our preferences.

A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience. Recommendation systems have several benefits, the most important being customer satisfaction and revenue. Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffers with poor recommendation quality and scalability issues.

REQUIREMENT ANALYSIS / SOFTWARE REQUIREMENT SPECIFICATION

JUPYTER NOTEBOOK

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components:

A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.

Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

You can start running a notebook server from the command line using the following command:

```
jupyter notebook
```

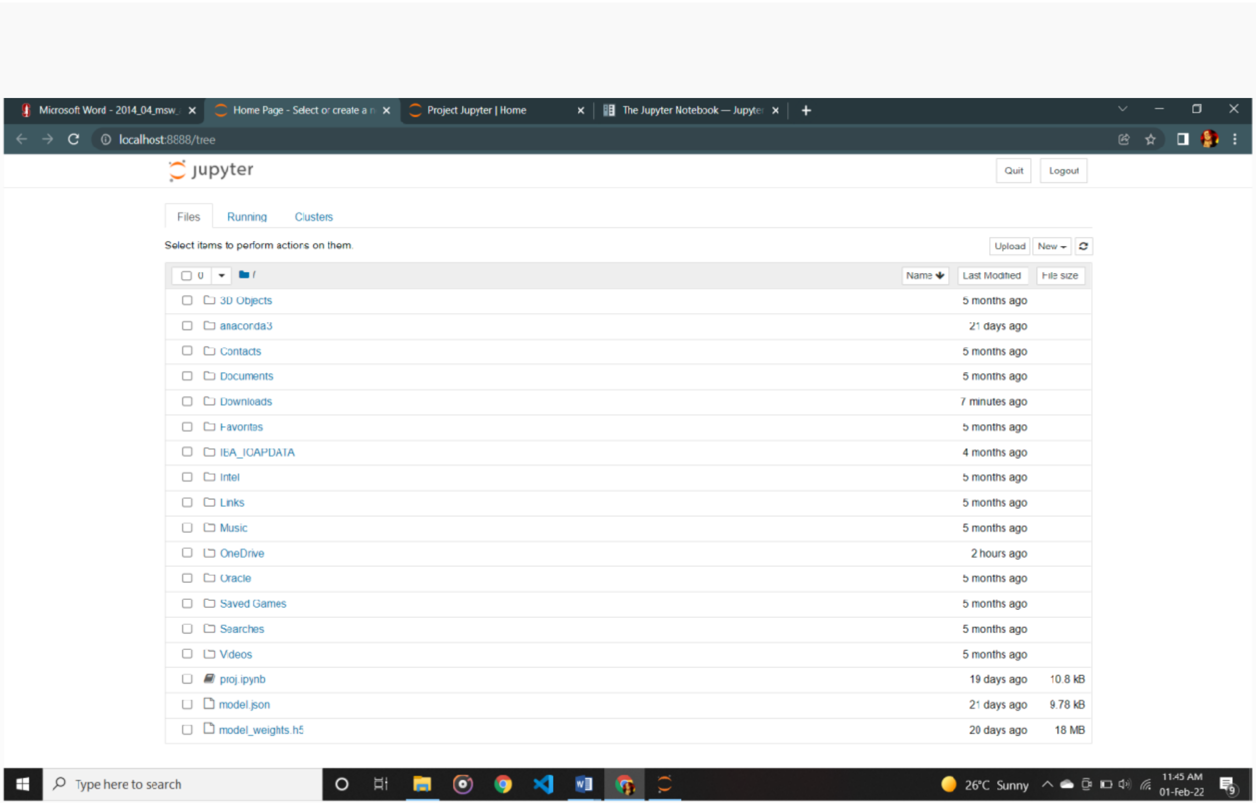
This will print some information about the notebook server in your console, and open a web browser to the URL of the web application (by default, http://127.0.0.1:8888).

The landing page of the Jupyter notebook web application, the **dashboard**, shows the notebooks currently available in the notebook directory (by default, the directory from which the notebook server was started).

You can create new notebooks from the dashboard with the `New Notebook` button, or open existing ones by clicking on their name. You can also drag and drop `.ipynb` notebooks and standard`.py`Python source code files into the notebook list area.

When starting a notebook server from the command line, you can also open a particular notebook directly, bypassing the dashboard, with `jupyter notebook my_notebook.ipynb`. The `.ipynb` extension is assumed if no extension is given.

When you are inside an open notebook, the *File | Open...* menu option will open the dashboard in a new browser tab, to allow you to open another notebook from the notebook directory or to create a new notebook.



Select items to perform actions on them.

Name	Last Modified	File size
3D Objects	5 months ago	
anacorda3	21 days ago	
Contacts	5 months ago	
Documents	5 months ago	
Downloads	7 minutes ago	
Favorites	5 months ago	
IBA_IOAPDATA	4 months ago	
Intel	5 months ago	
Links	5 months ago	
Music	5 months ago	
OneDrive	2 hours ago	
Oracle	5 months ago	
Saved Games	5 months ago	
Searches	5 months ago	
Videos	5 months ago	
proj.ipynb	19 days ago	10.8 kB
model.json	21 days ago	9.78 kB
model_weights.h5	20 days ago	18 MB

GOOGLE COLAB

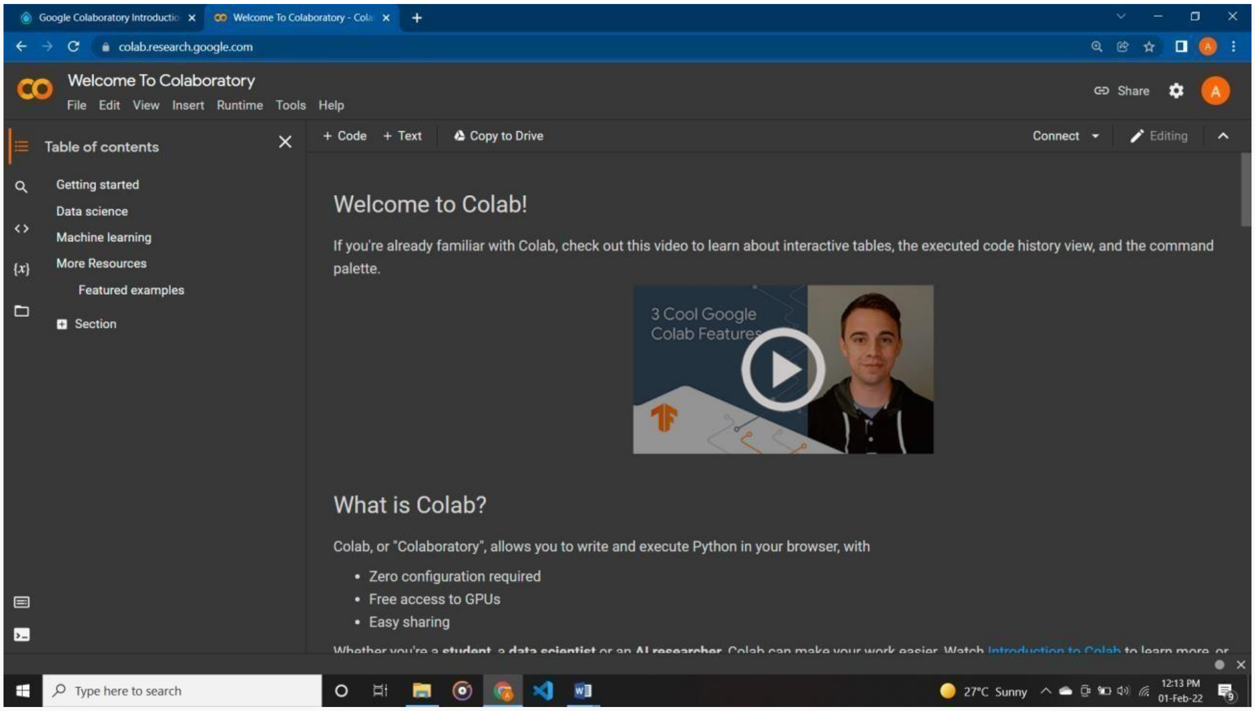
Colab is a free notebook environment that runs entirely in the cloud. It lets you and your team members edit documents, the way you work with Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook. Google is quite aggressive in AI research. Over many years, Google developed AI framework called **TensorFlow** and a development tool called **Colaboratory**. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply **Colab**.

Google Colaboratory, or Colab, is a cloud-based environment for writing documents with live code, visualizations, and narrative text. For those who are familiar with Jupyter notebooks, Colab notebooks are the same, including the .ipynb extension. Unlike Jupyter and Atom (our previous editor for code and reports), however, Colab requires no setup on your computer! It also provides a large amount of free computing power and easy document sharing.

A Colab notebook consists of **text cells**, **code cells**, and **outputs of code cells**.

1. Text cells are written in **Markdown**, a markup language we'll learn about in the next section. This means they can contain formatted text, images, HTML, LaTeX, and more.
2. Code cells are written in Python. We can also insert a system/terminal command by prefixing a line with !, like so:

   ```
   print("This is Python code.")
   !echo This is a system command.
   ```
3. Outputs of code cells appear below their corresponding cell. They can include text, graphics, and information about errors that occurred while executing the code.



Welcome to Colab!

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.

What is Colab?

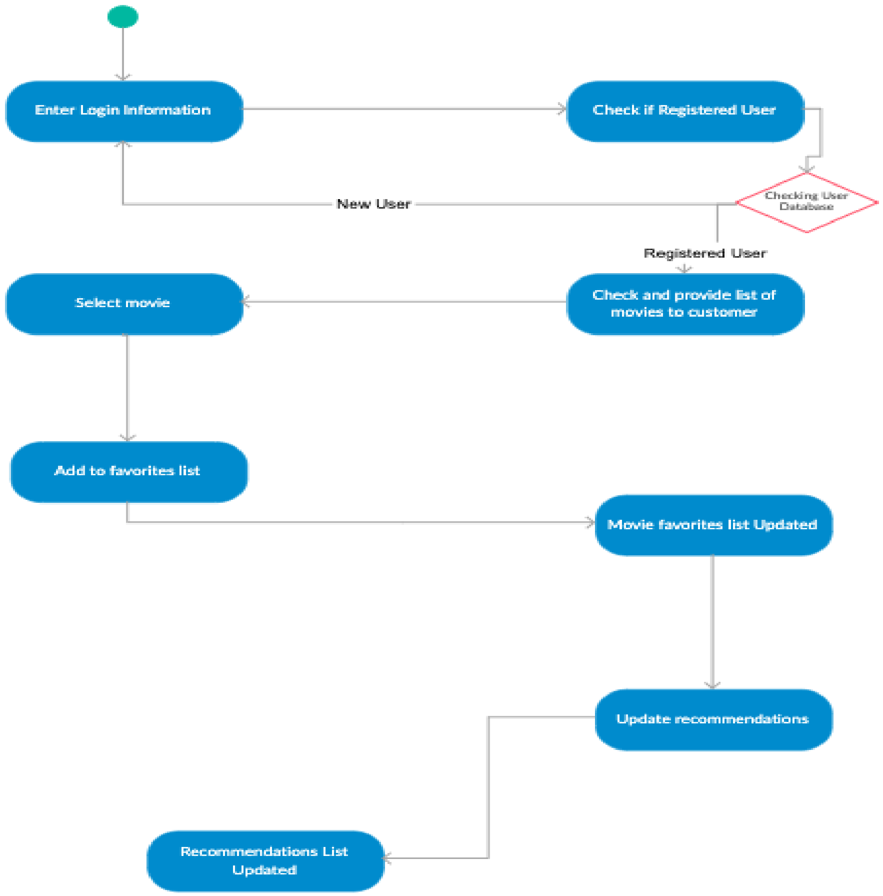
Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

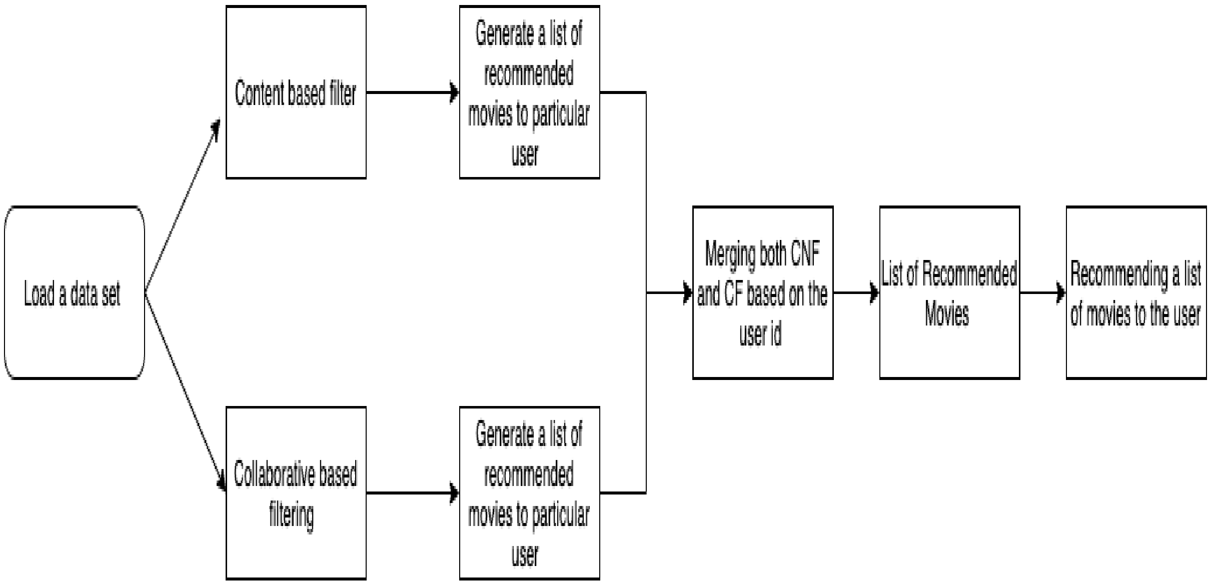
Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or

ANALYSIS AND DESIGN

ACTIVITY DIAGRAM



DATAFLOW DIAGRAM


Load a data set
Content based filter
Collaborative based filtering
Generate a list of recommended movies to particular user
Generate a list of recommended movies to particular user
Merging both CNF and CF based on the user id
List of Recommended Movies
Recommending a list of movies to the user

Initially load the data sets that are required to build a model the data set that are required in this project are movies.csv, ratinfg.csv, users.csv all the data sets are available in the Kaggle.com. Basically, two models are built in this project content based and collaborative filtering each produce a list of movies to a particular user by combining both based on the useid a single final list of movies are recommended to the particular user

IMPLEMENTATION

CODE

```
import numpy as np
import pandas as pd
import ast
import seaborn as sns
```

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

```
movies.head(1)
```

	budget	genres	homepage	id	keywords	original_languag
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	e

```
movies.shape
```

```
(4803, 20)
```

```
credits.head()
```

	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	[{"credit_id": "54805967c3a36829b5002c41"

```
credits.shape
```

```
(4803, 4)
```

```
movies = movies.merge(credits,on='title')
```

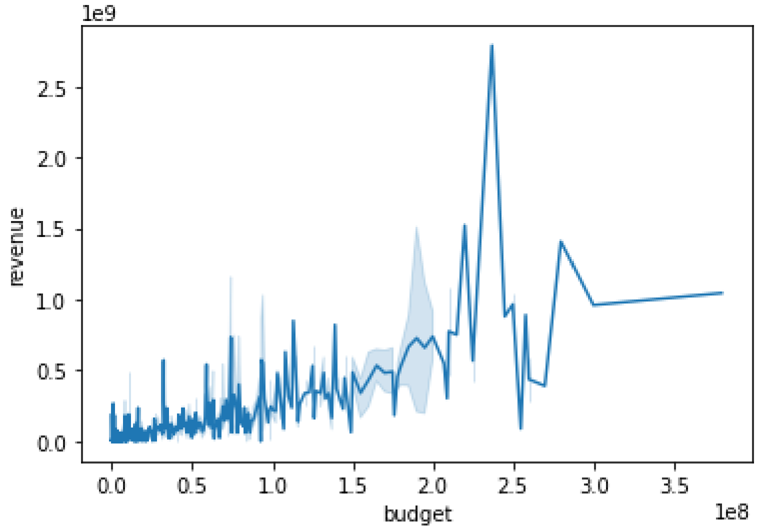
```
movies.head(1)
```

duction_companies	production_countries	release_date	revenue	runtime	spoken_

[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "US", "name": "United States o...	2009-12-10	2787965087	162.0	[{"iso_63 "name":

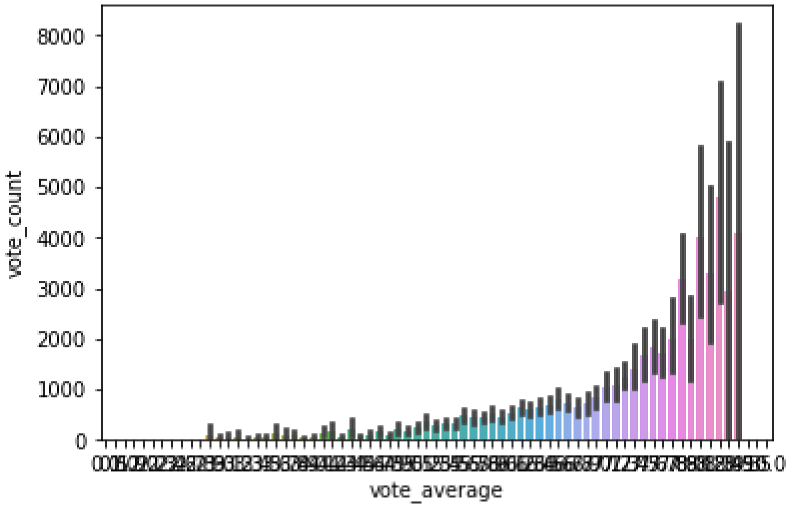
```
sns.lineplot(movies['budget'],movies['revenue'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: P
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f3880218850>
```



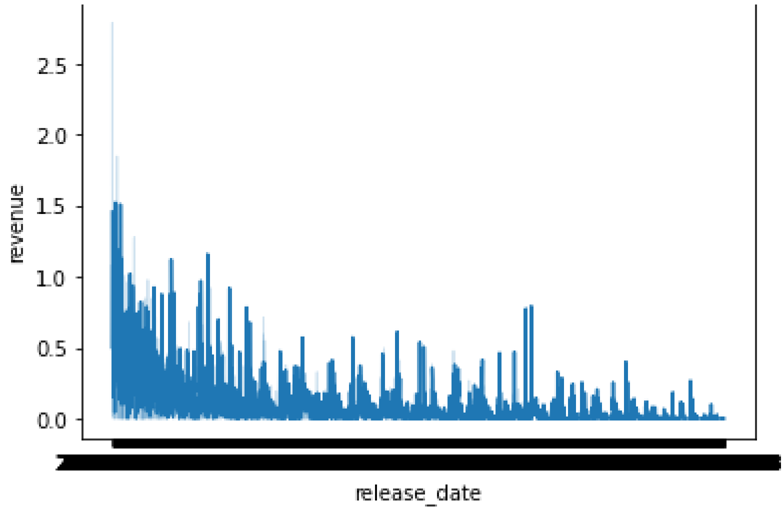
```
sns.barplot(x="vote_average",y="vote_count",data=movies)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f387fbf42d0>
```



```
sns.lineplot(x="release_date",y="revenue",data=movies)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f387779e110>
```



```
movies.head(1)
```

	budget	genres	homepage	id	keywords	original_languag
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	e

```
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
```

```
movies.head()
```

movie_id	title	overview	genres	keywords	cast
		In the	[{"id": 28	[{"id":	[{"cast_id":

22nd "name": 1463, 242,

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

the long "name": 4,

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

End dead, Jack Spa...

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

A cryptic [{"id":

```
movies.head()
```

	movie_id	title	overview	genres	keywords	cast	
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	"52fe4800925141
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	"52fe4232c3a36
2	206647	Spectre	A cryptic message from Bond's past	[Action, Adventure, Crime]	[{"id": 470, "name": "spy"}, {"id": 818	[{"cast_id": 1, "character": "James	"54805967c3a368

```
movies['cast'] = movies['cast'].apply(lambda x:x[0:3])
```

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

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

```
movies.sample(5)
```

	movie_id	title	overview	genres	keywords	cast	crew
		Dude	Jesse and		[{"id": 419,		

2741	8859	Where's My Car?	Chester, two bumbling stoners, wake ...	[Comedy]	"name": "dude"}, {"id": 1453, "na...	[{"	[Danny Leiner]
892	524	Casino	The life of the gambling paradise – Las Vegas ...	[Drama, Crime]	[{"id": 383, "name": "poker"}, {"id": 726, "na...	[{"	[Martin Scorsese]

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

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

```
movies.head()
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, ScienceFiction]	[[, {, ", i, d, ", :, , 1, 4, 6, 3, ,, , ", n,...	[[, {, "]	[JamesCameron]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[[, {, ", i, d, ", :, , 2, 7, 0, ,, , ", n, a,...	[[, {, "]	[GoreVerbinski]
2	206647	Spectre	A cryptic message from	[Action, Adventure,	[[, {, ", i, d, ", :, , 4, 7,	[[, {,	[SamMendes]

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

```
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast
```

```
new = movies.drop(columns=['overview','genres','keywords','cast','crew'])
#new.head()
```

```
new['tags'] = new['tags'].apply(lambda x: " ".join(x))
new.head()
```

	movie_id	title	tags
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...
2	206647	Spectre	A cryptic message from Bond’s past sends him o...
			Following the death of District Attorney

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

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

```
vector.shape
```

```
(4806, 5000)
```

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

```
similarity = cosine_similarity(vector)
```

```
similarity
```

```
array([[1.        , 0.1767767 , 0.0942809 , ..., 0.        , 0.        ,
        0.        ],
       [0.1767767 , 1.        , 0.1       , ..., 0.04003204, 0.        ,
        0.        ],
       [0.0942809 , 0.1       , 1.        , ..., 0.03202563, 0.        ,
        0.        ],
       ...,
       [0.        , 0.04003204, 0.03202563, ..., 1.        , 0.08006408,
        0.02787473],
       [0.        , 0.        , 0.        , ..., 0.08006408, 1.        ,
        0.05802589],
       [0.        , 0.        , 0.        , ..., 0.02787473, 0.05802589,
        1.        ]])
```

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

```
744
```

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

```
recommend('Batman')
```

```
Batman
The Dark Knight Rises
Batman & Robin
Batman Begins
Amidst the Devil's Wings
```

```
recommend('Batman')
```

```
Batman
The Dark Knight Rises
Batman & Robin
Batman Begins
Amidst the Devil's Wings
```

```
recommend('The Avengers')
```

```
Allegiant
Avengers: Age of Ultron
The Blood of Heroes
Team America: World Police
The Helix... Loaded
```

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

```
<module 'pickle' from '/usr/lib/python3.7/pickle.py'>
```

TESTING

Since our project is movie recommendation system .one can develop a movie recommendation system by using either content based or collaborative filtering or combining both.

In our project we have developed a hybrid approach i.e combination of both content and collaborative filtering .Both the approaches have advantages and dis-advantages
.in content based filtering the it based on the user ratings or user likes only such kind of movie will recommended to the user.

Advantages: it is easy to design and it takes less time to compute

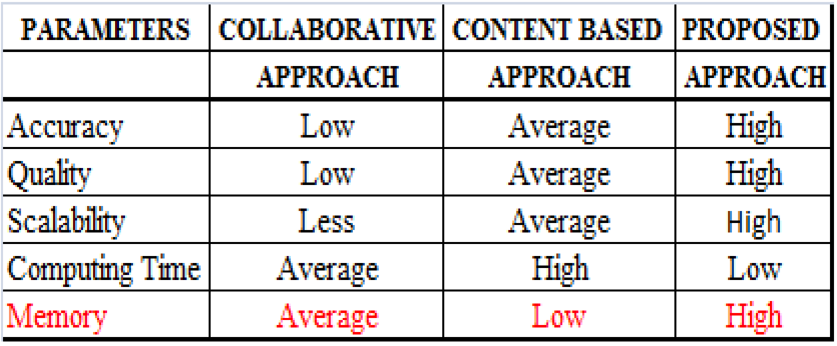
Dis-advantages: the model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

In Collaborative filtering the recommendation is comparison of similar users.

Advantages: No need domain knowledge because the embeddings are automatically learned. The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.

Dis-advantages: The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the **cold-start problem.**

The hybrid approach will resolves all these limitations by combining both content and collaborative filtering



PARAMETERS	COLLABORATIVE	CONTENT BASED	PROPOSED
	APPROACH	APPROACH	APPROACH
Accuracy	Low	Average	High
Quality	Low	Average	High
Scalability	Less	Average	High
Computing Time	Average	High	Low
Memory	Average	Low	High

Fig:-6.1 Comparison between the three approaches

REFERENCES

- https://colab.research.google.com/?utm_source=scs-index
- https://jupyter-notebook.readthedocs.io/en/latest/notebook.html#introduction