MOVIE RECOMMENDATION SYSTEM

ROSHNI S
PES1UG20CS360
PES UNIVERSITY(RR CAMPUS)
roshnishivu50@gmailcom

AKASH S
PES1UG20CS534
PES UNIVERSITY(RR CAMPUS)

SAI MEGHANA
PES1UG20CS191
PES UNIVERSITY(RR CAMPUS)
Kancharalmeghana30@gmail.com

Abstract*— A recommendation system is a system that, depending on some data, makes suggestions to users for specific resources like books, movies, songs, etc. In this world, entertainment is one of the most important necessities to refresh our mood and energy. Recommendation engines have become more popular as e-commerce has expanded. A movie recommendation system is an approach to filter or predict the users' film preferences based on their past choices and behavior or content. There are generally three ways of movie recommendation-content based, collaborative filtering and a hybrid of both.When creating a movie recommendation system, many variables can be taken into account, including the movie's genre, cast, and even director. The algorithms can suggest movies based on a single attribute or a combination of two or more.To improve the quality of a movie recommendation system, a Hybrid approach by combining content based filtering and collaborative filtering.Hybrid approach helps to get the advantages from both the approaches as well as tries to eliminate the drawbacks of both methods.

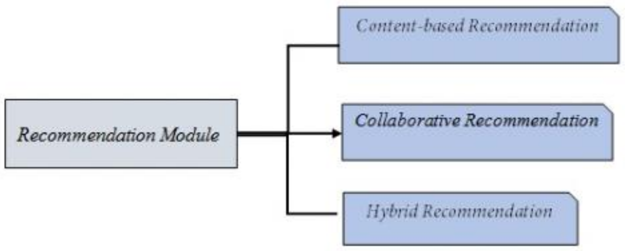
***Keywords*— movie recommendation system,Hybrid,Content based, collaborative filtering.**

INTRODUCTION:

The amount of data transfers that take place every minute in this Internet-era have drastically expanded. With the rise of Internet users, the enormous volume of data has dramatically expanded. However, not all of the information on the Internet is useful or gives users what they want. Large volumes of data frequently turn out to be inconsistent, and without proper processing, hence they are wasted. In such circumstances, consumers must rerun their search several times before finding what they first sought after. To solve this problem ,researchers have come up with recommendation systems .In today's age, a majority of organizations implement recommendation systems for fulfilling customer requirements. LinkedIn, Amazon, and Netflix are just a few to name. The approach adopted to do so is content-based filtering using genre correlation or even the user's history. We usually use pandas and numpy in python.

Recommendation systems are Artificial Intelligence based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their profile, search/browsing history.

Recommender systems are broadly classified into three types—collaborative filtering systems, content-based filtering systems, and hybrid systems



1. SIMPLE RECOMMENDERS :

The simple recommendation system process provides generalized recommendations to each user, supporting movie popularity and / or genre. The basic approach behind this technique is that more popular and critically acclaimed movies will be likely to be liked by the general audience. For example, IMDB Top 250 is an example of this technique.

2. CONTENT BASED FILTERING :

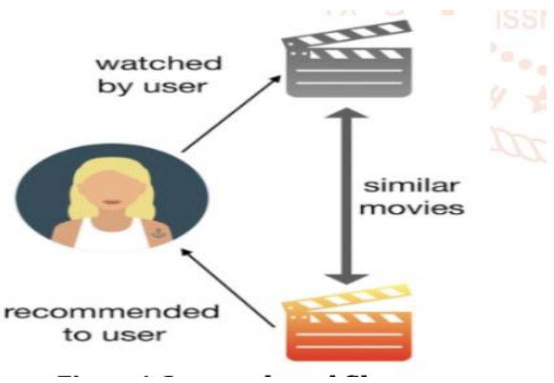
Content-based filtering systems examine documents or user preferences to create a model based on this information. They employ the user's specific interests and make an effort to match the properties that the various content items to be recommended have to the user's profile. They additionally suffer from the drawback of needing a sufficient amount of data to create a solid classifier.-The point of content-based filtering system is to know the content of both user and item. Usually it constructs and then compare user-profile and item-profile using the content of shared attribute space.

Advantages of content-based filtering:

- This has the capability of mentioning unrated objects
- It is easy to explain the work of recommender system by catalog the content features of an item.
- Content-based recommender systems require only the rating of the concerned client, no one else's.

Disadvantages of content-based filtering:

- A new user who has not rated any item will not work for them, as enough ratings are required content-based recommender assesses the client inclinations and gives exact proposals.
- No proposals of unexpected objects.
- Restricted Content Study - If the system fails to distinguish the items the recommender does not work.
- A client likes items that he/she does not like.



Examples of content-based filtering:

1 . LIBRA : A Content-Based Book Recommending System, using learning for text categorization.

2. CBMRS : A Content-Based Music Recommendation System.

3. PRES : A Content-Based Home Improvement Recommender System.

4. Cobra : A Content-Based Filtering and Aggregation of Blogs and RSS Feeds

3. COLLABORATIVE FILTERING :

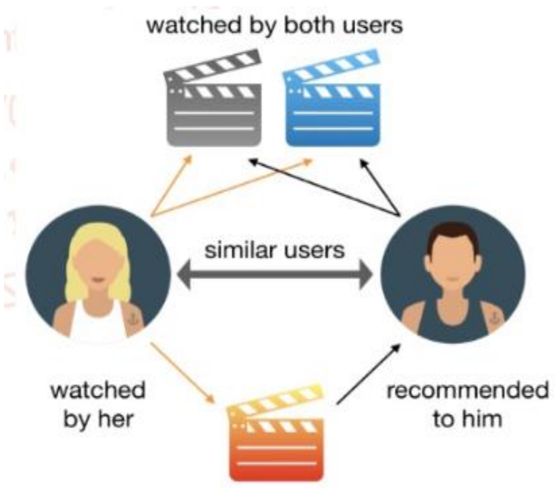
Collaborative systems take inputs from many users and compare these inputs in various ways. They construct models based on users prior behavior. Systems for recommending movies, for instance, make use of user ratings for various films in an effort to connect with other users who share their opinions and suggest films they have enjoyed. Collaborative filtering systems have two approaches—memory-based approaches and model-based approaches . Both memory-based and model-based techniques are used in collaborative filtering. Memory-based methods continually examine user data before making recommendations. They gradually get more accurate as they use the user ratings. They don't need content analysis and are independent of domains. By creating a model of a user's past behavior, model-based techniques can then predict that user's future behavior using a set of parameters. Additionally, the employment of partitioning-based techniques improves accuracy and scalability.

Advantages of collaborative filtering-based filtering:

- It relies upon the connection between clients which infers that it is content-autonomous.
- Collaborative Filtering recommender systems can propose unexpected items by detecting like-minded people's behavior.
- They can make genuine quality evaluations of objects by considering other people's experience.
- Cold start problem.

Disadvantages of collaborative filtering-based filtering:

- Early rater problem: Collaborative filtering systems cannot suggest for fresh items as there are no client ratings on which to base a forecast.
- Gray sheep: In order to Collaborative filtering-based system to work, group with related features are required. It will be very tough to suggest clients who do not steadily agree or disagree to these groups, if such groups exist.
- Sparsity problem: Mostly, the number of items exceeds the number of clients by a huge margin which makes it tough to find items that are rated by ample individuals.

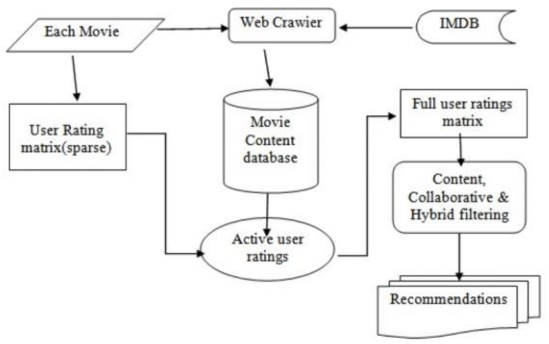


4. HYBRID SYSTEMS :

Hybrid systems combine collaborative and content- based filtering systems, in order to optimize the recommender systems, and reduce the drawbacks present in each of the two methods . Thus, it aims to maximize the advantages of one method while minimizing the drawbacks of the other. Weighted hybrid, mixed hybrid, and cross-source hybrid are the three different forms of hybrid systems. For each object in weighted hybrid systems, a score is kept, and the weighted aggregate is determined in relation to the numerous context sources. Depending on the user's settings, these are given varying weights. In mixed hybrid techniques, the top few items are chosen from each rank list after each source has been used to rank the various items. Hybrid cross-source approaches suggest things that show up in various context sources. These techniques operate under the premise that an item's importance increases with the number of sources it appears in.

Advantages of Hybrid approach :

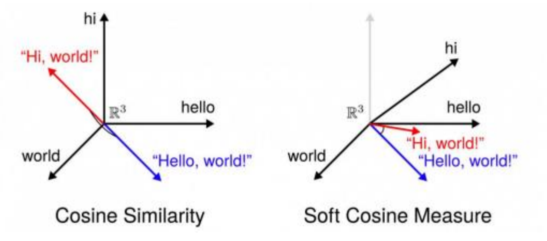
- Compared to pure collaborative and content-based methods, hybrid methods can provide more accurate recommendations.
- They can also overcome the common issues in recommendation systems such as cold start and the data paucity troubles.

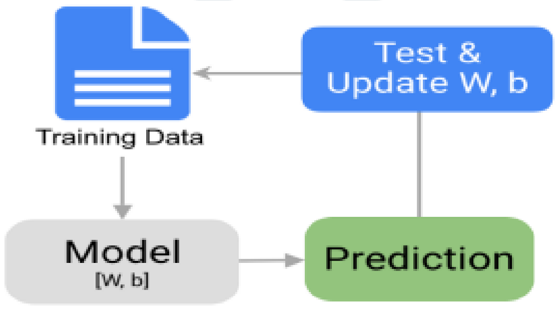


LITERATURE SURVEY:

A recommendation system developed by a previous researcher utilizing a content-based ,collaborative and hybrid approach is viewed as a distinct approach to the creation of recommendation engines in the context of literature review. Using the hybrid filtering method, a web-based and knowledge-based intelligence movie recommendation system was presented in 2007. In 2017, the authors developed a movie recommendation system that was supported by style and rating coefficient of correlation. In 2013, it was suggested to utilize a Bayesian network and trust model-based movie recommendation engine to predict ratings for users and objects, mostly from datasets, in order to recommend choices to users and the other way around. In 2018, the authors built a recommendation engine by analyzing the ratings dataset collected from Kaggle to recommend movies for a user selected from Python.In 2018, users can categorize people using movie recommendation engines that use reinforcement learning-based recommender systems and similar k-mean cuckoo values. Initial study mostly focused on the recommendation system's content, which looked at the characteristics of the object to finish the recommended task. Experiments showed how flexible and precise their methods were. For model-based preferences dependent on their context, Bayesian networks are used. The probabilistic matrix factor, a collaborative filtering technique that can handle big datasets, was proposed by Salakhuddinov and Minh in 2007. In Hurlkartal's movie suggestion, the collaborative filtering algorithm was divided into sections for further investigation. Collaborative filtering finds it challenging to respond quickly when clients adopt new behavior . Therefore, in order to address the problem, both scholars and practitioners thought of matching collaborative filtering approach and the content-based methodology.

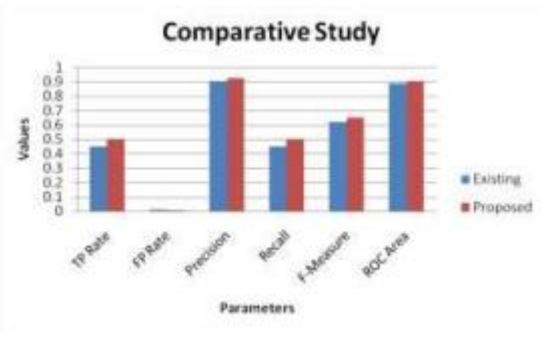
Based on MOVIE RECOMMENDATION SYSTEM USING CONTENT BASED FILTERING paper written by A Kiran Kumar,Gourab Pal Chowdhury, Sruti Bihani ,Rimpi Datta which was published in 2022 in which items were grouped based on their features and movies were recommended using the cosine similarity algorithm after conducting univariate and multivariate analysis on the dataset.





This method helps in improvising the accuracy of the model more and more.

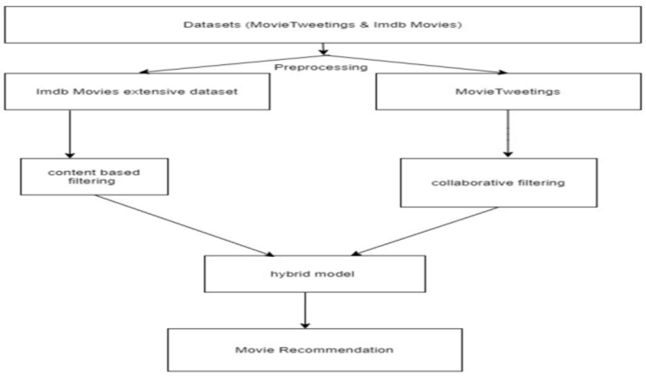
Based on Movie Recommendation System paper written by Yogesh Kumar, Ms. Naveen Kumari was published on 2020 in which classification was done based on collaborative filtering. It can be done based on many approaches such as matrix factorization, user-based recommendation, item-based recommendation. In this paper, a movie recommendation system based on the combination of opinion mining and user similarity analysis was created. This system helped to recommend top-k movies for target user. Their method showed better performance than ALS and SEHRS.



Based on An Improved Approach for Movie Recommendation System written by Shreya Agarwal and Pooja Jain on hybrid approach by combining content based filtering and collaborative filtering, using Support Vector Machine as a classifier and genetic algorithm. This helps to get the advantages from both the approaches as well as tries to eliminate the drawbacks of both methods. This paper also proves that Hybrid approach and genetic algorithm is better and provides better performance than the existing pure content based or collaborative filtering movie recommendation system in terms of accuracy, quality, scalability and computing time. Comparative results depicts that the proposed approach shows an improvement in the accuracy, quality and scalability of the movie recommendation system than the pure approaches. Also, computing time of the proposed approach is lesser than the other two pure approaches.

PROBLEM STATEMENT AND APPROACH :

Entertainment is a necessity for each one of us to refresh our mood and energy .It makes us gain our confidence for work which makes us to work more enthusiastically and efficiently. Recommendation systems are becoming increasingly important in today's extremely busy world. People are always short on time with all the tasks they need to accomplish in the limited 24 hours. Therefore, the recommendation systems are important as they help them make the right choices, without having to expend their cognitive resources. We have decided to use hybrid approach as it gives better and accurate results than content based and collaborative filtering individually .It required less computational time and it meets all the advantages of content based and collaborative but the disadvantages are easily overcome. We are planning to increase the efficiency of this hybrid model so that the results can be more accurate and according to the desires of the users. we form a batch set from the whole data set that get recommended using CBF technique i.e., we find top 25 from the dataset that are similar to the movie given as input using the cosine similarity model in the CBF technique and apply collaborative approach wherein we find the users that have similar interest as the input user, and then pick out the top 10 movies out of the 25 using the estimate factor .Sentiment analysis is a core of the hybrid implementation as we are optimizing the results gotten by the model using sentiment analysis hence locating the tweets, gathering them and using the VADER algorithm for analysis and to find the sentiment rating is a task and updating the results each day would make it more difficult, that is why the dataset MovieTweetings is used.



DATASET :

The datasets used are :

- MovieTweetings dataset
- IMIDB dataset

Sentiment analysis can be done by optimizing the results gotten by the model using sentiment analysis hence locating the tweets, gathering them and using the VADER algorithm for analysis and to find the sentiment rating is a task and updating the results each day would make it more difficult, that is why the dataset MovieTweetings is used.

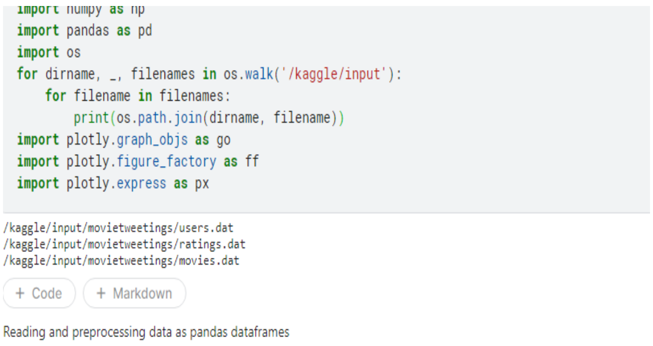
The MovieTweeting dataset is an open-source research project that has a system set up that locates, gathers and analyses the tweets for sentiment scores and also updates the dataset each day so that whatever data is contained is new and cannot be ruled out as old and unusable.

The system that's set up uses keywords to search for the movie and/or uses the "#IMIDB" for searching some of the ratings that the users rate on IMIDB platform, all these (the direct rating form IMIDB and sentiment analysis) are considered and cumulatively used to find the actual average sentiment rating that is recorded in the Ratings.dat file of the dataset. The dataset contains 3 files:

movies.dat: This file contains the movieId, movie title and the genre of the movie

users.dat: This file contains the userId, and the twitterId of the user

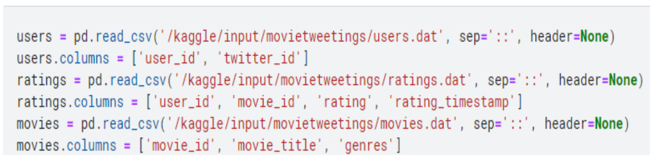
ratings.dat: this file contains userId, movieId, rating(calculated using sentiment analysis) and the rating timestamp.



```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
import plotly.graph_objs as go
import plotly.figure_factory as ff
import plotly.express as px
```

```
/kaggle/input/movietweetings/users.dat
/kaggle/input/movietweetings/ratings.dat
/kaggle/input/movietweetings/movies.dat
```

Reading and preprocessing data as pandas dataframes

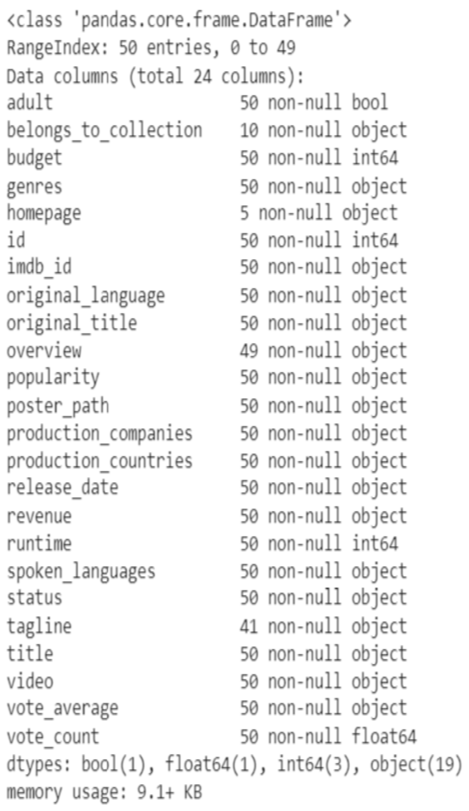


```
users = pd.read_csv('/kaggle/input/movietweetings/users.dat', sep='::', header=None)
users.columns = ['user_id', 'twitter_id']
ratings = pd.read_csv('/kaggle/input/movietweetings/ratings.dat', sep='::', header=None)
ratings.columns = ['user_id', 'movie_id', 'rating', 'rating_timestamp']
movies = pd.read_csv('/kaggle/input/movietweetings/movies.dat', sep='::', header=None)
movies.columns = ['movie_id', 'movie_title', 'genres']
```

One of few popular and reliable techniques used to find the sentiment scores using the text is called Valence Awareness Dictionary for sentiment reasoning (VADER). VADER is sensitive to both polarity (whether the sentiment is positive or negative) and Intensity (how positive or negative is sentiment) of emotions, hence the pre-processed data/tweets from the gingerit function is taken as the input to this algorithm.

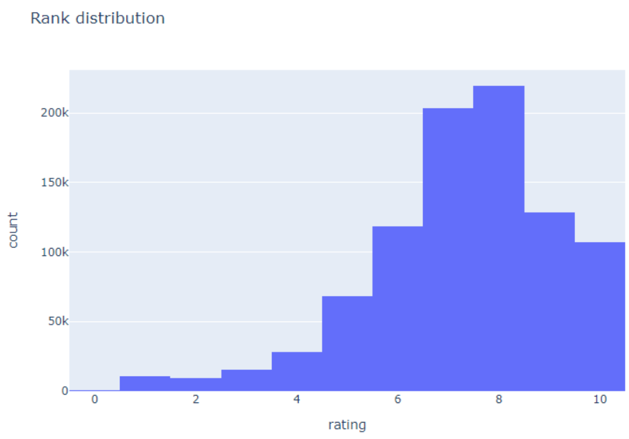
INITIAL ANALYSIS:(EXPLORATORY DATA ANALYSIS)

The dataset movies_metadata.csv has the following details(information of the dataset)

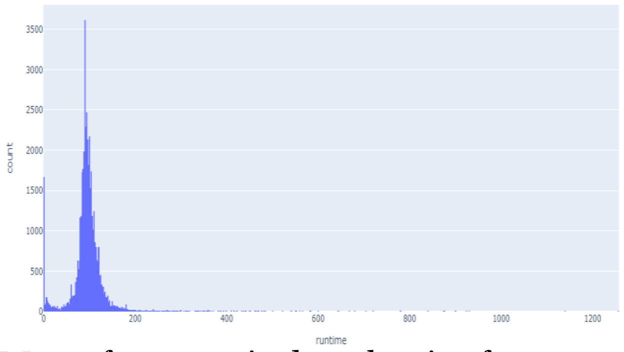


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 24 columns):
adult                    50 non-null bool
belongs_to_collection    10 non-null object
budget                   50 non-null int64
genres                   50 non-null object
homepage                 5 non-null object
id                       50 non-null int64
imdb_id                  50 non-null object
original_language        50 non-null object
original_title           50 non-null object
overview                 49 non-null object
popularity               50 non-null object
poster_path              50 non-null object
production_companies     50 non-null object
production_countries     50 non-null object
release_date             50 non-null object
revenue                  50 non-null object
runtime                  50 non-null int64
spoken_languages         50 non-null object
status                   50 non-null object
tagline                  41 non-null object
title                    50 non-null object
video                    50 non-null object
vote_average             50 non-null object
vote_count               50 non-null float64
dtypes: bool(1), float64(1), int64(3), object(19)
memory usage: 9.1+ KB
```

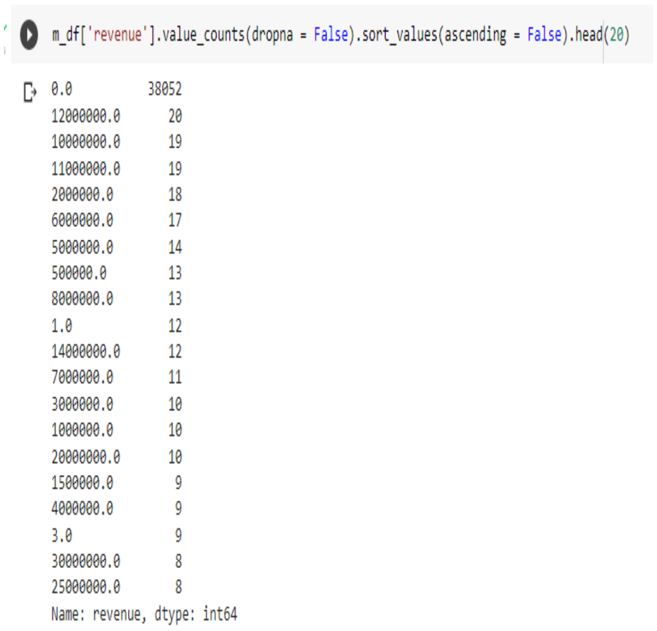
This is the histogram which represents rank distribution with count on y axis and rating on x axis.



First we read the dataset movies_metadata.csv from our drive and view all its columns.Then we view the shape of the data and its total size.We also plot a histogram of runtime to see various movies' duration.

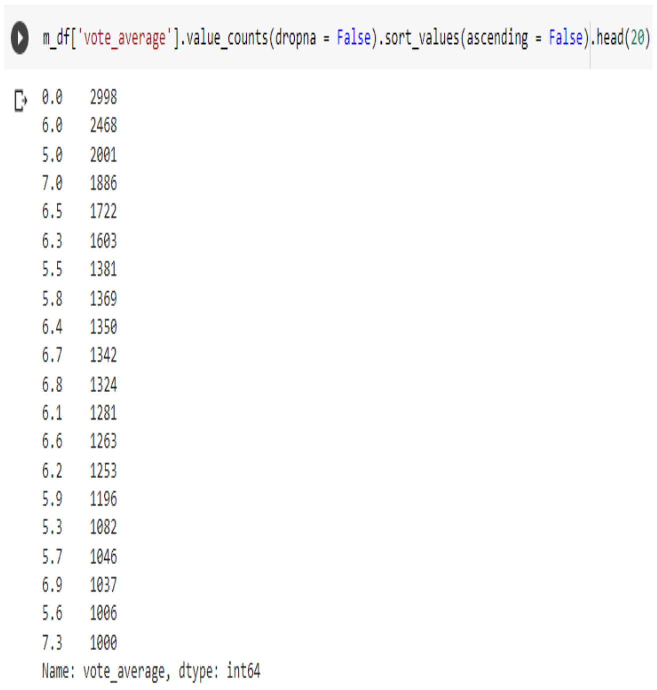


We perform certain data cleaning factors on revenue,vote_average and vote_count.



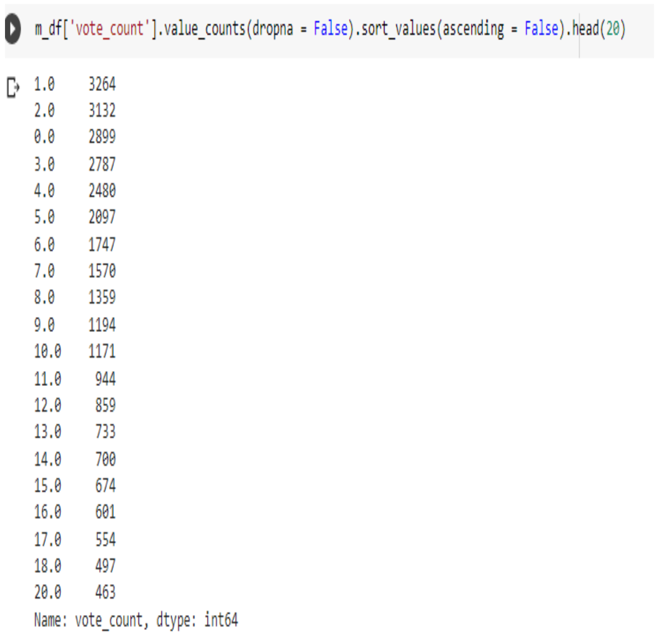
```
m_df['revenue'].value_counts(dropna = False).sort_values(ascending = False).head(20)
```

```
0.0           38052
12000000.0       20
10000000.0       19
11000000.0       19
2000000.0        18
6000000.0        17
5000000.0        14
500000.0         13
8000000.0        13
1.0              12
14000000.0       12
7000000.0        11
3000000.0        10
1000000.0        10
20000000.0       10
1500000.0         9
4000000.0         9
3.0               9
30000000.0        8
25000000.0        8
Name: revenue, dtype: int64
```



```
m_df['vote_average'].value_counts(dropna = False).sort_values(ascending = False).head(20)
```

```
0.0    2998
6.0    2468
5.0    2001
7.0    1886
6.5    1722
6.3    1603
5.5    1381
5.8    1369
6.4    1350
6.7    1342
6.8    1324
6.1    1281
6.6    1263
6.2    1253
5.9    1196
5.3    1082
5.7    1046
6.9    1037
5.6    1006
7.3    1000
Name: vote_average, dtype: int64
```

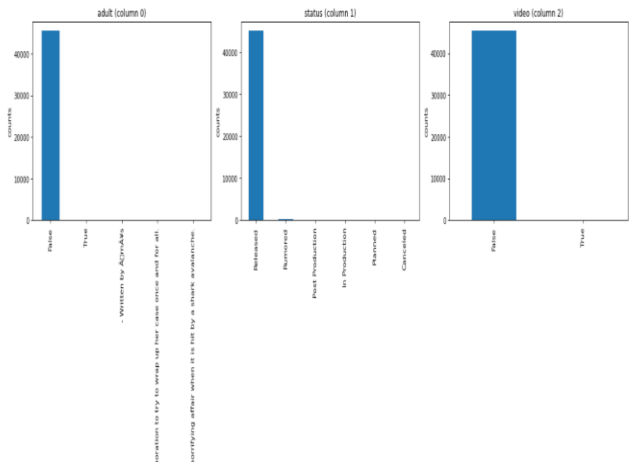


```
m_df['vote_count'].value_counts(dropna = False).sort_values(ascending = False).head(20)
```

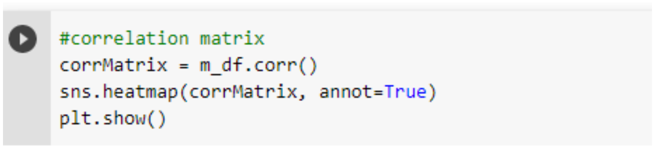
```
1.0     3264
2.0     3132
0.0     2899
3.0     2787
4.0     2480
5.0     2097
6.0     1747
7.0     1570
8.0     1359
9.0     1194
10.0    1171
11.0     944
12.0     859
13.0     733
14.0     700
15.0     674
16.0     601
17.0     554
18.0     497
20.0     463
Name: vote_count, dtype: int64
```

Then we create a plotPerColumnDistribution function to plot the below histograms.

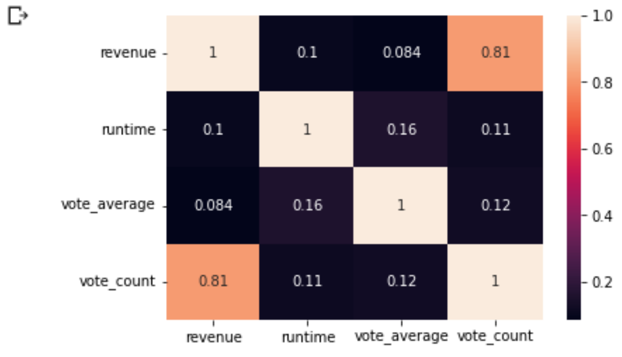
```
plotPerColumnDistribution(df1,20, 4)
```

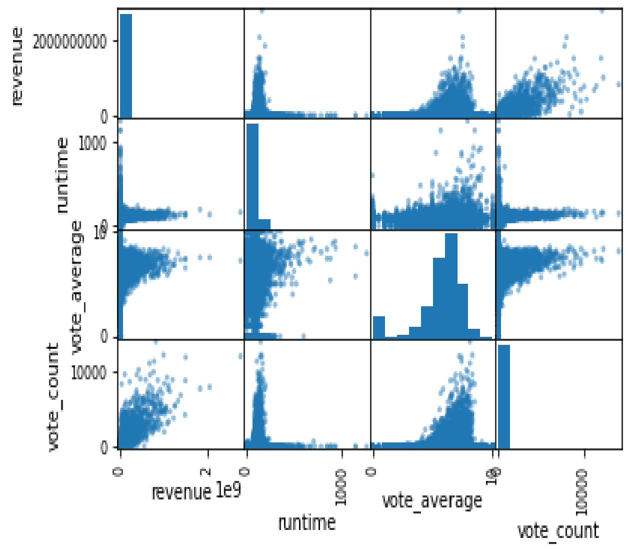


We plot a heatmap representing the correlation matrix which represents the relation between each columns with one another,i.e, interdependencies among each other.



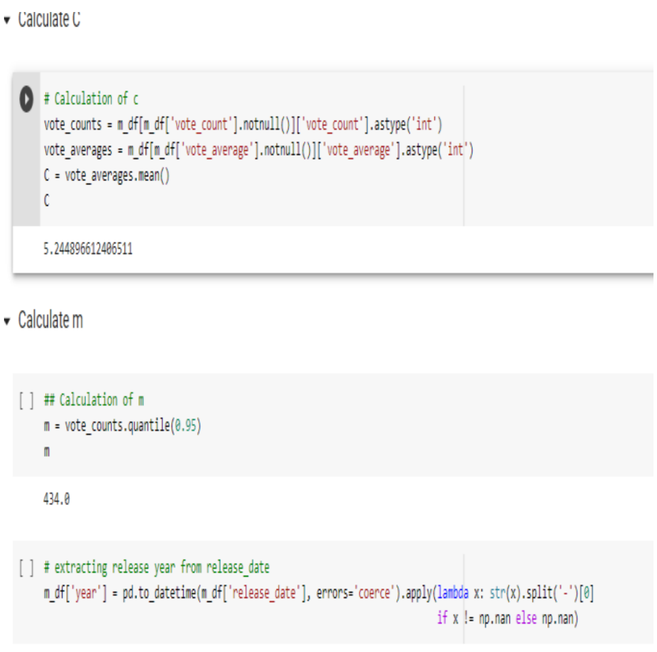
```
#correlation matrix
corrMatrix = m_df.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```





Finally we plot a scatter plot which represents bi-variate or pairwise relationship between different combinations of variables while laying them in grid form.

 Then we calculate c and m which represent the average of the votes by the people on those movies and their mean respectively. Then we extract the release year from release_date.



▾ Calculate C

```
# Calculation of c
vote_counts = m_df[m_df['vote_count'].notnull()]['vote_count'].astype('int')
vote_averages = m_df[m_df['vote_average'].notnull()]['vote_average'].astype('int')
C = vote_averages.mean()
C
```

```
5.244896612406511
```

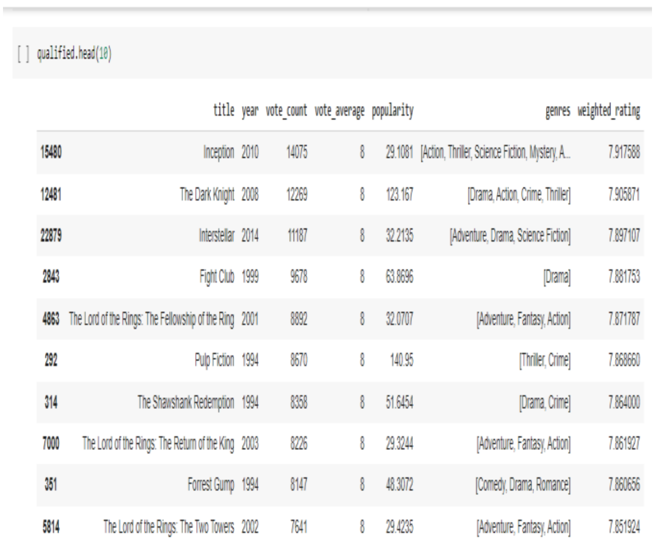
▾ Calculate m

```
[ ] ## Calculation of m
    m = vote_counts.quantile(0.95)
    m
```

```
434.0
```

```
[ ] # extracting release year from release_date
    m_df['year'] = pd.to_datetime(m_df['release_date'], errors='coerce').apply(lambda x: str(x).split('-')[0]
                                                                              if x != np.nan else np.nan)
```

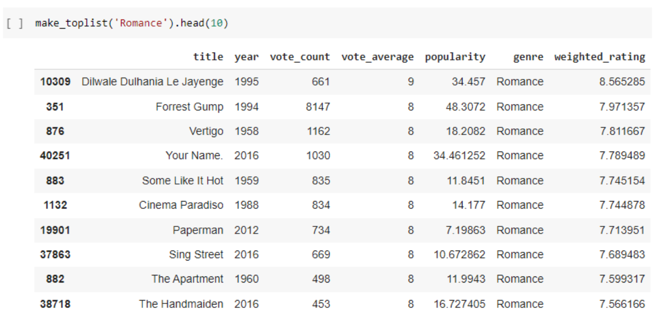
Then the qualifies movies are segregated by using 95% as cutoff.



```
[ ] qualified.head(10)
```

	title	year	vote_count	vote_average	popularity	genres	weighted_rating
15480	Inception	2010	14075	8	29.1081	[Action, Thriller, Science Fiction, Mystery, A...	7.917588
12481	The Dark Knight	2008	12269	8	123.167	[Drama, Action, Crime, Thriller]	7.905871
22879	Interstellar	2014	11187	8	32.2135	[Adventure, Drama, Science Fiction]	7.897107
2843	Fight Club	1999	9678	8	63.8696	[Drama]	7.881753
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.0707	[Adventure, Fantasy, Action]	7.871787
292	Pulp Fiction	1994	8670	8	140.95	[Thriller, Crime]	7.868660
314	The Shawshank Redemption	1994	8358	8	51.6454	[Drama, Crime]	7.864000
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.3244	[Adventure, Fantasy, Action]	7.861927
351	Forrest Gump	1994	8147	8	48.3072	[Comedy, Drama, Romance]	7.860656
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.4235	[Adventure, Fantasy, Action]	7.851924

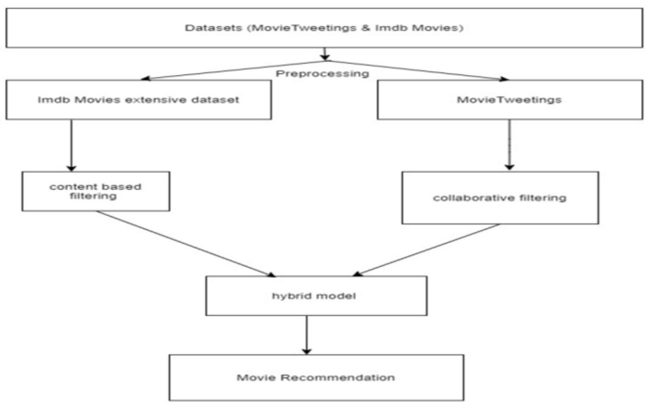
Finally we are segregating top movies in each genre according to their rating resulting in the new modified data.For example lets see top movies in the genre romance:



```
[ ] make_toplist('Romance').head(10)
```

	title	year	vote_count	vote_average	popularity	genre	weighted_rating
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457	Romance	8.565285
351	Forrest Gump	1994	8147	8	48.3072	Romance	7.971357
876	Vertigo	1958	1162	8	18.2082	Romance	7.811667
40251	Your Name.	2016	1030	8	34.461252	Romance	7.789489
883	Some Like It Hot	1959	835	8	11.8451	Romance	7.745154
1132	Cinema Paradiso	1988	834	8	14.177	Romance	7.744878
19901	Paperman	2012	734	8	7.19863	Romance	7.713951
37863	Sing Street	2016	669	8	10.672862	Romance	7.689483
882	The Apartment	1960	498	8	11.9943	Romance	7.599317
38718	The Handmaiden	2016	453	8	16.727405	Romance	7.566166

SYSTEM ARCHITECTURE :



As we can see in the above architecture diagram, after preprocessing of the dataset Imdb movies and MovieTweetings we are going to implement content based and collaborative filtering and then combine then forming the hybrid model for the best results .

1. **For Content Based filtering**
- Data Preprocessing
- Text vectorization
- Cosine Similarity matrix
- Get recommendation
2. **For Collaborative filtering**
- SVD (Singular Value Decomposition)
3. **For Hybrid Recommendation system**

Hybrid approach to the recommender system involves combining two - Content based and Collaborative filtering into one single model that has the features of both the techniques. Basically we form a batch set from the whole data set that get recommended using CBF technique i.e., we find top 25 from the dataset that are similar to the movie given as input using the cosine similarity model in the CBF technique and apply collaborative approach wherein we find the users that have similar interest as the input user, and then pick out the top 10 movies out of the 25 using the estimate factor.

TECHNIQUES USED IN PROPOSED METHODOLOGY :

Cosine Similarity matrix :

Cosine similarity is the cosine of the angle between two vectors and it is used as a distance evaluation metric between two points in the plane. The cosine similarity measure operates entirely on the cosine principles where with the increase in distance the similarity of data points reduces. Cosine similarity in the recommendation system is used with the same principle of cosine angles, where even if the similarity of the content is less similar it would be considered as the least recommended content, and for higher similarity of contents, the recommendations generated would be at the top. Cosine similarity is also used in textual data to find the similarity between the vectorized texts from the original text document. Hamming distance considers only the character type of data of the same length but cosine similarity has the ability to handle variable length data. So the cosine similarity would yield a similarity matrix for the selected textual data for recommendation and the content with higher similarity scores can be sorted using lists. Cosine similarity entirely operates on the cosine angle properties and it is vastly used in recommendation systems as it will help us recommend content to the user according to his most viewed content and characteristics and is also majorly used in finding the similarity between text documents as it considers the frequently occurring terms. This made cosine similarity a popular metric for evaluation in various applications.

SVD (Singular Value Decomposition) :

SVD is used as a collaborative filtering technique. It uses a matrix structure where each row represents a user, and each column represents an item. The elements of this matrix are the ratings that are given to items by users.

$$A = USV^T$$

Where *A* is a *m x n* utility matrix, *U* is a *m x r* orthogonal left singular matrix, which represents the relationship between users and latent factors, *S* is a *r x r* diagonal matrix, which describes the strength of each latent factor and *V* is a *r x n* diagonal right singular matrix, which indicates the similarity between items and latent factors. The SVD decreases the dimension of the utility matrix *A* by extracting its latent factors. It maps each user and each item into a *r*-dimensional latent space. This mapping facilitates a clear representation of relationships between users and items. Let each item be represented by a vector *xi* and each user is represented by a vector *yu*. The expected rating by a user on an item $\hat{r}_{ui}$ can be given as: Here, $\hat{r}_{ui}$ is a form of factorisation in singular value decomposition.
The *xi* and *yu* can be obtained in a manner that the square error difference between their dot product and the expected rating in the user-item matrix is minimum. It can be expressed as :In order to let the model generalise well and not overfit the training data, a regularisation term is added as a penalty to the above formula. In order to reduce the error between the value predicted by the model and the actual value, the algorithm uses a bias term. Let for a user-item pair (u, i), *μ* is the average rating of all items, *bi* is the average rating of item i minus *μ* and *bu* is the average rating given by user *u* minus *μ*, the final equation after adding the regularisation term and bias can be given as:

$$\hat{r}_{ui} = x_i^T . y_u$$

$$Min(x, y) \sum_{(u,\, i) \in K} (r_{ui} - x_i^T . y_u)^2$$

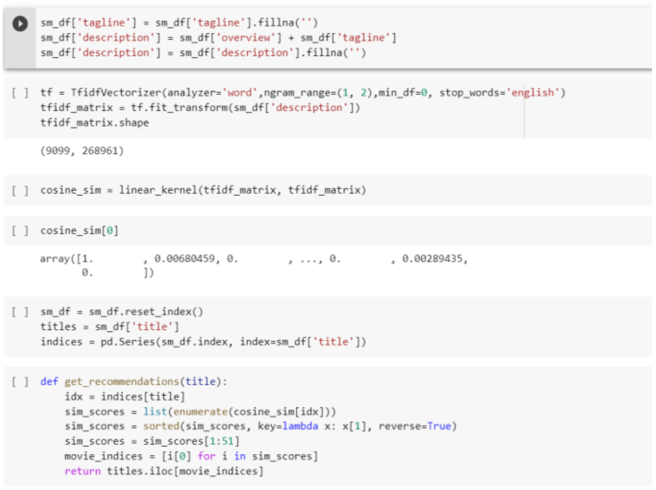
$$Min(x, y) \sum_{(u,\, i) \in K} (r_{ui} - x_i^T . y_u)^2 + \lambda(||x_i||^2 + ||y_u||^2)$$

$$Min(x, y, b_i, b_u) \sum_{(u,\, i) \in K} (r_{ui} - x_i^T . y_u - \mu - b_i - b_u)^2 + \lambda(||x_i||^2 + ||y_u||^2 + b_i^2 + b_u^2)$$

PROPOSED METHODOLOGY :

CONTENT BASED RECOMMENDATION MODEL :

We have used text vectorization and then implemented cosine similarity to perform content based recommendation. We have used overview and tagline of the movies for getter better recommendation results. We have used details of the movie from links-small.csv file. So the movies are suggested based on the content and description of movies.



```
sm_df['tagline'] = sm_df['tagline'].fillna('')
sm_df['description'] = sm_df['overview'] + sm_df['tagline']
sm_df['description'] = sm_df['description'].fillna('')
```

```
tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(sm_df['description'])
tfidf_matrix.shape
```

```
(9099, 268961)
```

```
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

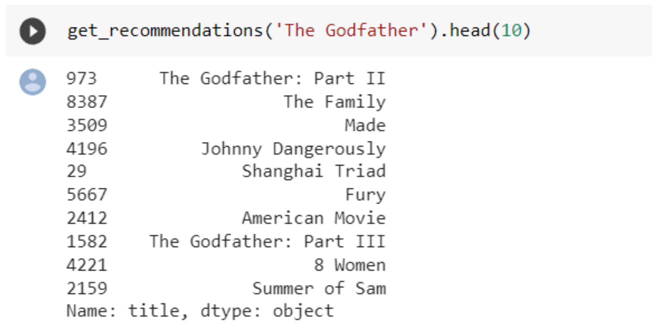
```
cosine_sim[0]
```

```
array([1.        , 0.00680459, 0.        , ..., 0.        , 0.00289435,
       0.        ])
```

```
sm_df = sm_df.reset_index()
titles = sm_df['title']
indices = pd.Series(sm_df.index, index=sm_df['title'])
```

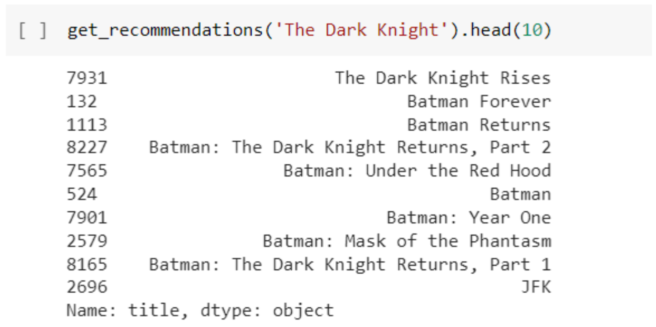
```
def get_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:51]
    movie_indices = [i[0] for i in sim_scores]
    return titles.iloc[movie_indices]
```

Lets see a few examples :



```
get_recommendations('The Godfather').head(10)
```

```
973       The Godfather: Part II
8387                  The Family
3509                        Made
4196          Johnny Dangerously
29                Shanghai Triad
5667                        Fury
2412              American Movie
1582     The Godfather: Part III
4221                     8 Women
2159               Summer of Sam
Name: title, dtype: object
```



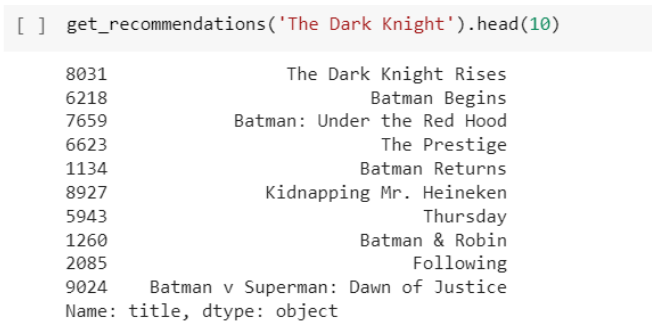
```
get_recommendations('The Dark Knight').head(10)
```

```
7931                      The Dark Knight Rises
132                              Batman Forever
1113                             Batman Returns
8227     Batman: The Dark Knight Returns, Part 2
7565                 Batman: Under the Red Hood
524                                      Batman
7901                           Batman: Year One
2579                Batman: Mask of the Phantasm
8165     Batman: The Dark Knight Returns, Part 1
2696                                        JFK
Name: title, dtype: object
```

METADATA BASED RECOMMENDATION MODEL:

First we create a dataframe with required information to the movie dataframe from credits.csv and keywords.csv and then we create a improved recommendation model using details such as vote_count and vote_average to get better results.

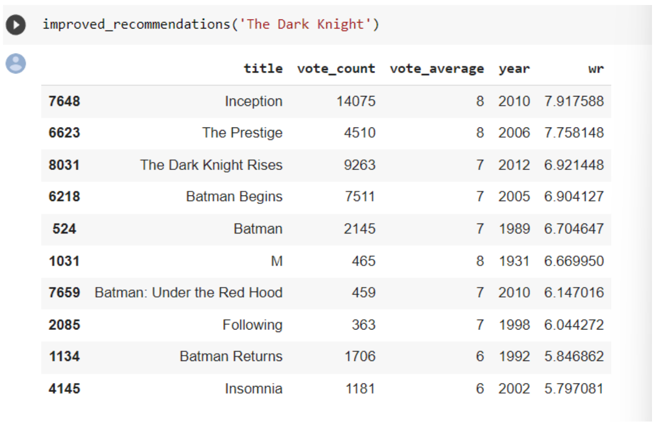
Normal content based recommendation :



```
get_recommendations('The Dark Knight').head(10)
```

```
8031                  The Dark Knight Rises
6218                          Batman Begins
7659             Batman: Under the Red Hood
6623                           The Prestige
1134                         Batman Returns
8927               Kidnapping Mr. Heineken
5943                               Thursday
1260                          Batman & Robin
2085                              Following
9024     Batman v Superman: Dawn of Justice
Name: title, dtype: object
```

Improvised recommendation :

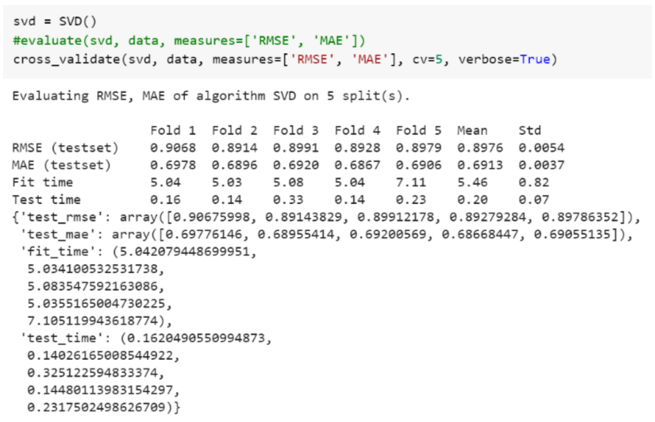


```
improved_recommendations('The Dark Knight')
```

	title	vote_count	vote_average	year	wr
7648	Inception	14075	8	2010	7.917588
6623	The Prestige	4510	8	2006	7.758148
8031	The Dark Knight Rises	9263	7	2012	6.921448
6218	Batman Begins	7511	7	2005	6.904127
524	Batman	2145	7	1989	6.704647
1031	M	465	8	1931	6.669950
7659	Batman: Under the Red Hood	459	7	2010	6.147016
2085	Following	363	7	1998	6.044272
1134	Batman Returns	1706	6	1992	5.846862
4145	Insomnia	1181	6	2002	5.797081

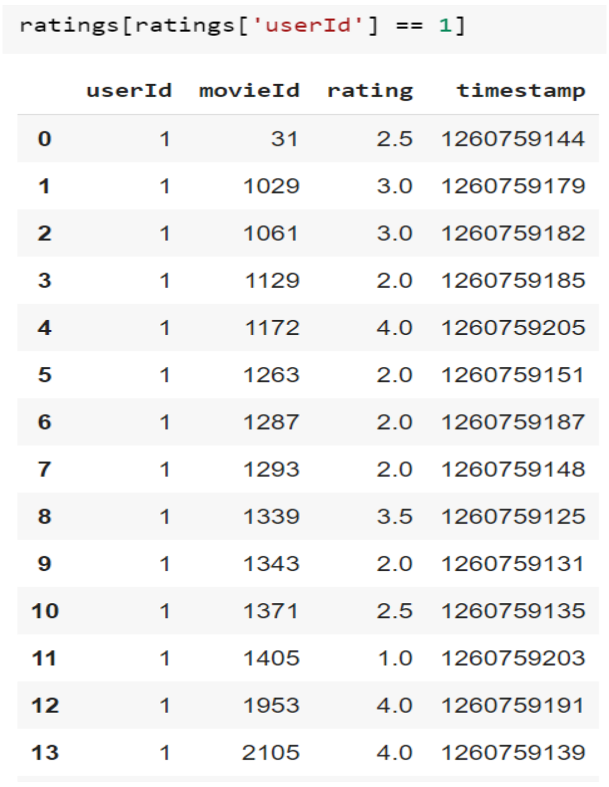
COLLABORATIVE FILTERING MODEL:

In this we consider that the user rates movies according to his liking and when another user watches the movie watched by the previous user then the movies are suggested to this user according to the prior user's liking represented as rating. We have used SVD technique to implement this. We have used data from raitings_small.csv and links_small.csv.



```
svd = SVD()
#evaluate(svd, data, measures=['RMSE', 'MAE'])
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std     
RMSE (testset)    0.9068  0.8914  0.8991  0.8928  0.8979  0.8976  0.0054  
MAE (testset)     0.6978  0.6896  0.6920  0.6867  0.6906  0.6913  0.0037  
Fit time          5.04    5.03    5.08    5.04    7.11    5.46    0.82    
Test time         0.16    0.14    0.33    0.14    0.23    0.20    0.07    
{'test_rmse': array([0.90675998, 0.89143829, 0.89912178, 0.89279284, 0.89786352]),
 'test_mae': array([0.69776146, 0.68955414, 0.69200569, 0.68668447, 0.69055135]),
 'fit_time': (5.042079448699951,
  5.034100532531738,
  5.083547592163086,
  5.0355165004730225,
  7.105119943618774),
 'test_time': (0.1620490550994873,
  0.14026165008544922,
  0.325122594833374,
  0.14480113983154297,
  0.2317502498626709)}
```

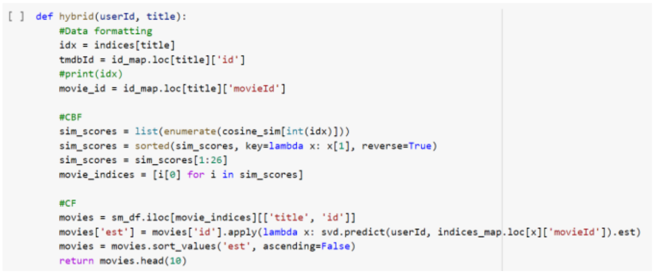


```
ratings[ratings['userId'] == 1]
```

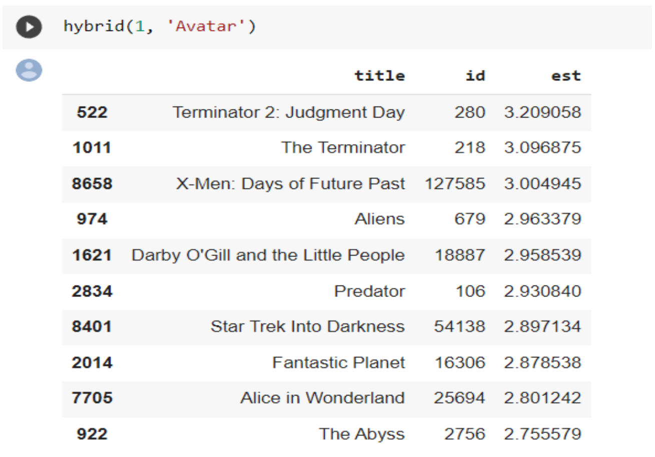
	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205
5	1	1263	2.0	1260759151
6	1	1287	2.0	1260759187
7	1	1293	2.0	1260759148
8	1	1339	3.5	1260759125
9	1	1343	2.0	1260759131
10	1	1371	2.5	1260759135
11	1	1405	1.0	1260759203
12	1	1953	4.0	1260759191
13	1	2105	4.0	1260759139

HYBRID RECOMMENDATION SYSTEM :

In this we combined both content and collaborative techniques which results in an effective model which has advantages of both content based and collaborative filtering but cancels out the drawbacks of the respective methods. Therefore this method can be considered one of the effective method for recommendation systems. We consider userid and movie name for further recommendations.

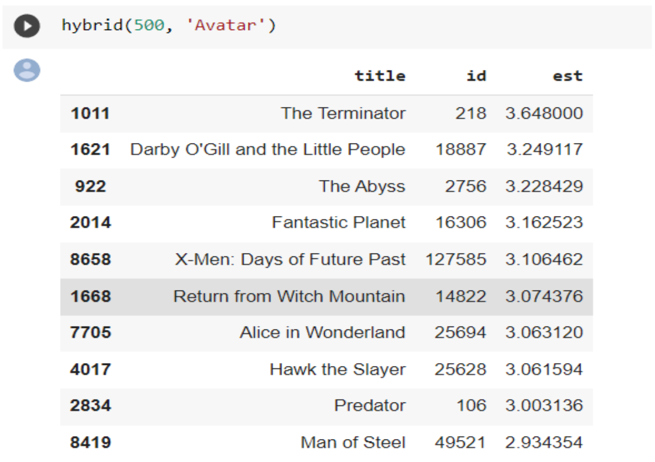


```
[ ] def hybrid(userId, title):
        #Data formatting
        idx = indices[title]
        tmdbId = id_map.loc[title]['id']
        #print(idx)
        movie_id = id_map.loc[title]['movieId']

        #CBF
        sim_scores = list(enumerate(cosine_sim[int(idx)]))
        sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
        sim_scores = sim_scores[1:26]
        movie_indices = [i[0] for i in sim_scores]

        #CF
        movies = sm_df.iloc[movie_indices][['title', 'id']]
        movies['est'] = movies['id'].apply(lambda x: svd.predict(userId, indices_map.loc[x]['movieId']).est)
        movies = movies.sort_values('est', ascending=False)
        return movies.head(10)
```



```
hybrid(1, 'Avatar')
```

	title	id	est
522	Terminator 2: Judgment Day	280	3.209058
1011	The Terminator	218	3.096875
8658	X-Men: Days of Future Past	127585	3.004945
974	Aliens	679	2.963379
1621	Darby O'Gill and the Little People	18887	2.958539
2834	Predator	106	2.930840
8401	Star Trek Into Darkness	54138	2.897134
2014	Fantastic Planet	16306	2.878538
7705	Alice in Wonderland	25694	2.801242
922	The Abyss	2756	2.755579



```
hybrid(500, 'Avatar')
```

	title	id	est
1011	The Terminator	218	3.648000
1621	Darby O'Gill and the Little People	18887	3.249117
922	The Abyss	2756	3.228429
2014	Fantastic Planet	16306	3.162523
8658	X-Men: Days of Future Past	127585	3.106462
1668	Return from Witch Mountain	14822	3.074376
7705	Alice in Wonderland	25694	3.063120
4017	Hawk the Slayer	25628	3.061594
2834	Predator	106	3.003136
8419	Man of Steel	49521	2.934354

CONCLUSION :

The Hybrid model's result is optimized as we use MovieTweetings dataset for the collaborative filtering part of and this dataset is an open-source dataset that is mode specifically for sentiment-based movie recommender.

FUTURE WORK :

We can expand the recommendation system by user a larger dataset and see how it works . By a larger dataset we mean to use a larger user ratings and opinions for a better recommendation system. We can further improve by creating a GUI or an app or even a website which can be user friendly. In future, we plan to make this system more accurate and platform for listing out movies with their metadata, where users can rate movie and get more accurate recommendation. This will help us to maintain user profiles more precisely. Currently our models support only English movies, in future we will add dataset of regional movies.

REFERENCES :

- sowmyashreem/Hybrid-Movies-Recommendation-System: Hybrid Movie Recommender system that uses concepts of 2 of the most popular recommender techniques: Collaborative and Content-based Filtering. (github.com)

- DA.ipynb - Colaboratory (google.com)

- (PDF) Movie Recommendation System (researchgate.net)

- Movie Recommendation System using content based filtering (jetir.org)

- An improved approach for movie recommendation system (researchgate.net)

- https://www.researchgate.net/publication/331966843_Content-Based_Movie_Recommendation_System_Using_Genre_Correlation

- https://www.ijert.org/recommendation-of-movies-based-on-collaborative-filtering-using-apache-spark

- https://www.iteratorshq.com/blog/an-introduction-recommender-systems

- C. S. M. Wu, D. Garg, and U. Bhandary, “Movie Recommendation System Using Collaborative Filtering,” In 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), pp. 11-15, IEEE, 2018 Nov.

- 2017, IEEE. 4. P. Phorasim and L. Yu, “Movies recommendation system using collaborative filtering and k-means,” International Journal of Advanced Computer Research, vol. 7, no. 29, p.52, 2017