# CONTENT BASED MOVIE RECOMMENDATION SYSTEM

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

**Computer Science and Engineering/Information Technology**

By

Harshit Sharma 181483
Sachin Sharma 181478

Under the supervision of

(Dr. Amol Vasudeva)

to

Department of Computer Science & Engineering and Information Technology
**Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh**

## CERTIFICATE

### Candidate's Declaration

I hereby declare that the work presented in this report entitled "CONTENT BASED MOVIE RECOMMENDATION SYSTEM" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Wakanaghat is an authentic record of my own work carried out over a period from January 2022 to May 2022 under the supervision of **Dr Amol Vasudeva**(Assistant Professor(SG)).
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Harshit Sharma
181483
Sachin Sharma
181478

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr Amol Vasudeva
Assistant Professor
Computer Science & Engineering
Dated:

# ACKNOWLEDGEMENT

I would like to thank and express our gratitude to our Project supervisor Dr. Amol Vasudeva for the opportunity that he provided us with this wonderful project “Content Based Movie Recommendation System”. The outcome would not be possible without his guidance. This project taught me many new things and helped to strengthen concepts of Machine Learning. Next, I would like to express my special thanks to the Lab Assistant for cordially contacting us and helping us in finishing this project within the specified time. Lastly, I would like to thank my friends and parents for their help and support.

Harshit Sharma 181483

Sachin Sharma 181478

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

A typical Recommender system engine extracts different types of data by using various algorithms and then giving out the best fit of the relevant items as needed by Looking at the nature of the previous behaviour of the user on the basis of which it recommends the items or products by recommending the best items which are currently in great demand. The other major reasons for this may be maximizing the gain. There are basically 3 ways which are used on recommending the items to the users. Demographic filtering which gives a general recommendation which is based on any one of the more popular movies or its genre. Having same demographic matches it recommends the users movies. In general, the movies that are popular are possibly liked more by the greater number of users. Other technique is content -based filtering that takes into account the interest of the people and then suggesting the movies to the users. Third technique that is used here is that Collaborative method of filtering where same people are put or grouped together which have same characteristic interest or movies pattern.

# CHAPTER 1

## 1.1) INTRODUCTION

A recommendation system is essentially a filtering system that anticipates the users' choices and then suggests the most accurate results based on the users' past preferences. We have a variety of applications for our recommendation system that we have used throughout the years and are actively applying on various internet platforms. [2] All of these platforms' fundamental content is just different movie genres such as action thriller romance or even your eCommerce website any social networking platform with a professional website such as LinkedIn.

For illustration, when we are using Instagram, we can see previous stories on the feeds of the people we follow, indicating that Instagram can observe our interactions with various people and our earlier activities, and then it simply suggests other related stories of some other accounts that have done some similar activity previously or currently. Many times, the recommender system keeps enhancing the activities of a large number of people based on the activities they have scrolled through you attempted. For example, when we buy something new like a laptop or a mobile phone on Flipkart, it simply offers purchasing a mobile cover tempered glass or a USB type C or type A adaptor for the laptop.

[3] Users get good recommendations all the time, and it keeps improving as we move forward in the twenty-first century, and they produce virtually precise answers. In the event of a conflict between any e App Music, any music platform, or any educational app, just prohibit using the app. Additionally, firms must focus on their recommendation system, which is more complex than it appears. Every user has different interests and preferences based on their numerous pursuits and emotions, such as music when playing, travelling, exercising, or after a relationship disagreement, and so on.

## 1.2) PROBLEM STATEMENT

Recommender systems are tools that aims to get the user's rating and then recommend the movies from a big set of data on the basis of the users matching interest and then classify them into different categories. [4] The sole purpose of the whole system of this recommendation is the search for the content that it would fit into the person's interest for an individual's personal oasis.

However, it takes into account different factors that would create some different list of content that is specific to different categories of individual/ users. [5] AI based algorithms that recommender systems basically used creates a list of possible different scenarios of devices and then customizing that all the interesting and matching interest/ choices of the individual categories in the end.

All the results are basically based on the different activities that they have done previously such as how does the profile look what have gone through the Chrome Browser Opera browser and other browser which includes their previously browsed history for considering the demographic traits or the possibility how they would like the movie is based on the genre, a set of predictive modelling is constructed through the data(big) which is available and then the movies are protected through the list of 2000 movies set a bunch of few selected movies are recommended using different algorithms different methods different similarity measures.

## 1.3) OBJECTIVE

Movie recommendation system provides the mechanism and classifying the users with the same interest and searches for the content that would be so much interesting belonging to different set of users and then creating different kind of lists and providing interesting recommendations to the individual based on the content the love.

[6] The main objective of the recommender system is to used approaches suggest demographic filtering, content-based filtering, collaborative filtering to find the set of movies with every user likes for specific set of users. The movies that have high probability of being liked by the general set of users will be displayed to the user by the recommender in the end and then in another technique we will try to find the users with different interest using the information collected through different activities an Indian in collaborative filtering will test all those users which have same type of interests to get the final set of movies to be recommended to the users individually.

So, we will use different categories of recommender filtering techniques and then compare in contrast that results obtained in different methods and will try to improve the results as h dataset for set of movies goes larger and larger above the computational bound of the system which is generally a limitation on the large dataset.

## 1.4) METHODOLOGY

There are several types of recommender systems, which we may categorize as follows:

**Demographic Filtering**

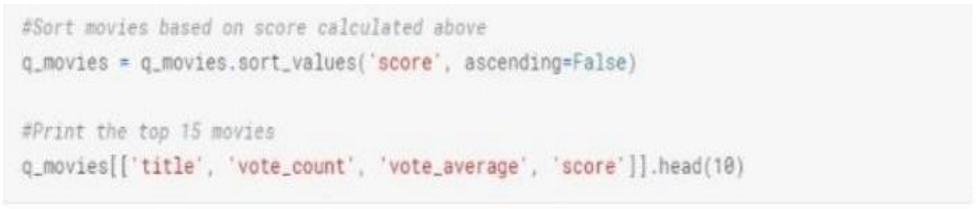
This suggestion filtering mechanism is based on the popularity of gender-specific users. [7] The technology merely recommends movies to people who share similar demographics. Because each user is unique in this scenario, using this strategy is quite straightforward. The concept is that films that are widely popular and accepted by a large number of people have the greatest chance of being liked by users.

To comprehend this demographic filtering, consider the following:

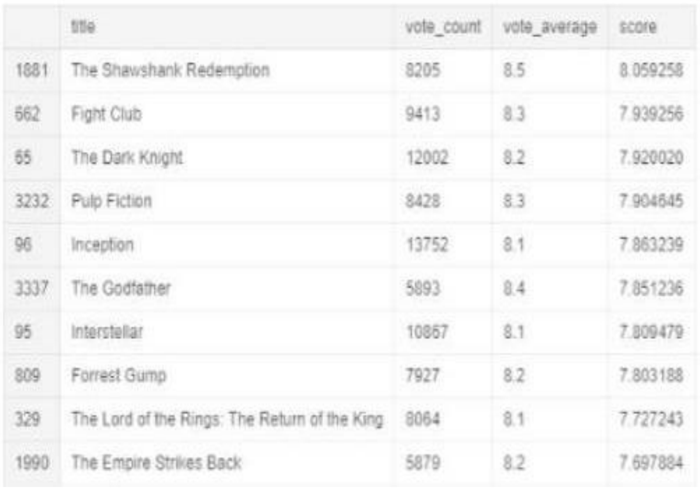
- Create a measure for rating the film
- Determining the various metric scores.
- Shortening the scores and then proposing the best-rated films for the viewers

Weighted Ratings (WR)=$\frac{v}{v+m}.R+\frac{m}{v+m}.C$

- 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
- c is the mean vote across the whole report



```
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)

#Print the top 15 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```



| | title | vote_count | vote_average | score |
|---|---|---|---|---|
| 1881 | The Shawshank Redemption | 8205 | 8.5 | 8.059258 |
| 662 | Fight Club | 9413 | 8.3 | 7.939256 |
| 65 | The Dark Knight | 12002 | 8.2 | 7.920020 |
| 3232 | Pulp Fiction | 8428 | 8.3 | 7.904645 |
| 96 | Inception | 13752 | 8.1 | 7.863239 |
| 3337 | The Godfather | 5893 | 8.4 | 7.851236 |
| 95 | Interstellar | 10867 | 8.1 | 7.809479 |
| 809 | Forrest Gump | 7927 | 8.2 | 7.803188 |
| 329 | The Lord of the Rings: The Return of the King | 8064 | 8.1 | 7.727243 |
| 1990 | The Empire Strikes Back | 5879 | 8.2 | 7.697884 |

**Fig 1: Demographic Filtering Method**

## Content Based Filtering system

We compare the different things with the user's interest profile in the content-based filtering technique. So, fundamentally, the user profile contains material that is much more relevant to using the form of the features. [8] Previous actions or feedback is often taken into consideration, as is the description of the information that has been modified by users of various selections.

[9] Consider the following scenario: A person buys a favorite item 'M,' but the item is sold out; as a consequence, he must buy the item 'N,' on the suggestion of someone else, because 'N' has the same sort of matching features as the first one. And that is basically the content-based filtration that is shown below.

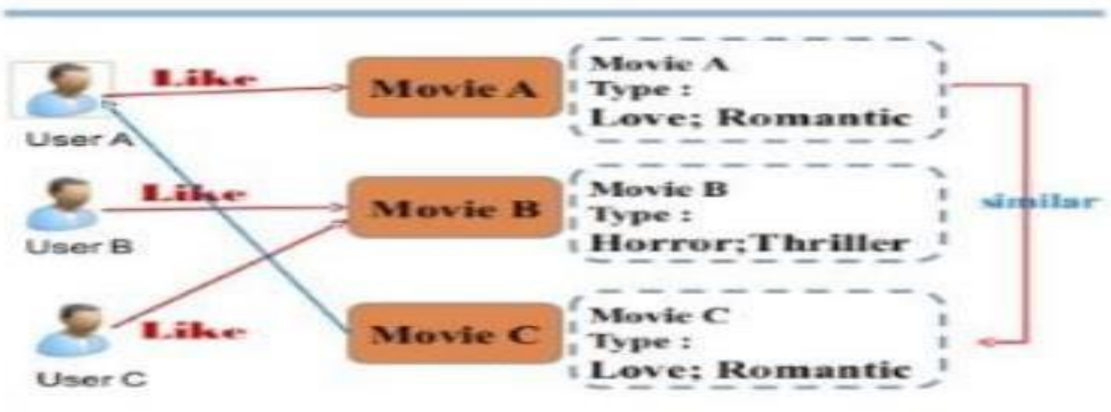


**Fig 2: Content Based Filtering Method**

So, in this case, the numeric number that will be utilized to determine the similarities between the two types of movies will be cosine similarity, and we will get the score extremely quickly.

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

The following are the stages required in obtaining a movie recommendation:

- With the title, you may locate the index of the movie.
- Computing the cosine similarity scores for each film.
- Arrange the scores in ascending order, starting with the greatest priority.
- The group is then pruned based on the similarity scores.
- Obtaining the first ten elements of the list, with the exception of the first, which is the movie title in and of itself.
- Obtaining the most important ingredients

Repeating the preceding steps, we would then find the top films based on the distances that it might get the highest suitable recommendation, the movies that have a high probability of being preferred by the general set of users will be displayed to the user by the recommender in the end, and then in another method, we would therefore try to find the users with various interests by using information gathered through multiple things, and an Indian in collaborative filtering will test all of the users.

The cosine similarity is the cause of the angle between the two vectors where the vectors are not zero, and it is defined in the inner product space as the dot product of the two vectors divided by the product of the Euclidean magnitude. [10] Cosine similarity is commonly used to generate preferred suggestions for users.

This method simply computes the cosine distance between the vectors before using similarity to compute the score and, lastly, the user's decision.

[11] For example, if a movie has a large number of actors that a group of users likes and only a few actors that a group of user's dislikes, we believe that plotting a good sign angle between the user and the movie vectors, which will generally be a large positive fraction, so the angle is nearly close to zero, and a comparatively small distance of cosine will be present between the two vectors is appropriate.

The cosine similarity fails when the cosine distance is large. To improve the recommender system in this case, we will employ a novel mechanism known as decision tree.

The following are the benefits of content-based filtering:

- We could suggest the unrated items.
- Can suggest movies depending on the user's ratings
- The following are the drawbacks of content-based filtering:
- Can't work upon the new user since the user hasn't seen the red kidney movie act.
- It is not possible to make the user like with un-likes.

**Collaborative based Filtering**

Content-based filtering has some limitations, including the ability to propose movies based on only one sort of user choice and the inability to make genre suggestions. However, a collaborative filtering-based method adds a layer of complexity to establishing a match between a user's similarities and the likes of other users with similar interests.

Cosine similarity or the Peerson's correlation can be used to assess the similarities of user viewpoints. [12] For example, in the Matrix below, each row includes a user with a column matching to the movies with the same resemblance; it also contains the ratings of different movies that the user has given to each movie, and each movie has a target user.

All of the collaborative filtering in the case of user-based is easy, but it has limitations, the most significant of which is that even the choices of the user where is with time. Pre-compiling the Matrix orphaned the issue of poor performance.

So, we may utilize item-based collaborative filtering, which simply examines the things based on their similarity with the items and that it finds comparable matches with the target users using the same similarity coefficients, such as Pearson's correlations or Cosines similarity. Item-based collaborative filtering is the most static.

[14] As an example, just one person has linked both Matrix and Titanic, therefore the resemblance between them is limited to one. There may be circumstances where we can have millions of users and the similarities between the two separate movies is really great since the person who rated them both has the same rank.

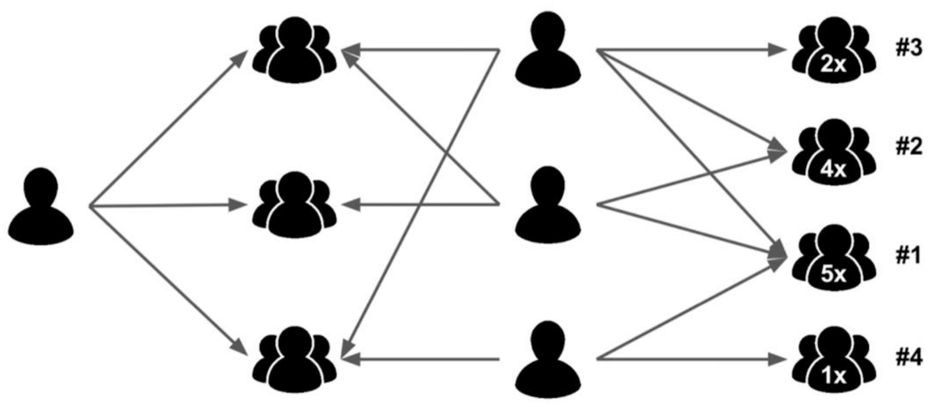
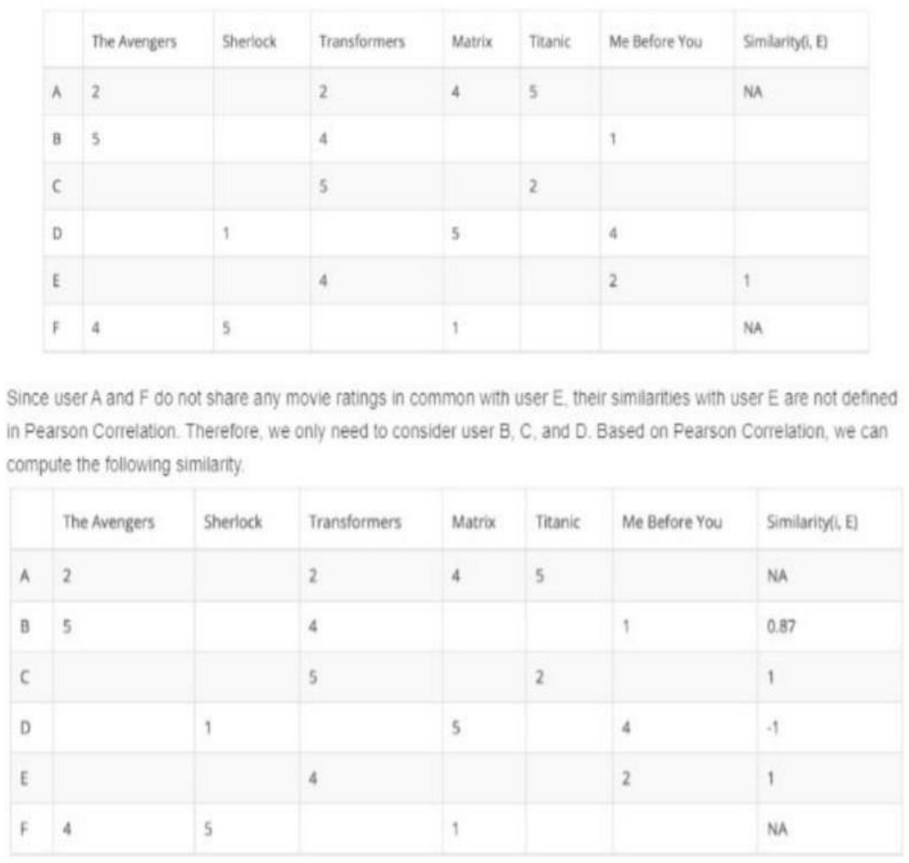


**Fig 3: User Based Collaborative Filtering Algorithm**

Using collaborative filtering, look for people who have a name and similar interests. Rather of promoting an item based on its features, we employ the categorization of user into groups of similar types and then split every cluster into the sequence of the user's choosing. In this case, we may also use the cosine distance, which takes into account persons with similar interests. The greater the cosine tiny angle between two users, the closer they are.

To simplify computations, we simply are using the utility matrix by assigning zero values to the discrete columns. According to an item in general, collaborative filtering is preferred since it evaluates the film instead of the quantity of people, making it easier to classify movies and users.

As a result of which, user-based collaborative filtering is not preferred since it only considers the user's and ignores the sparse values, causing challenges with the recommender system's performance.



| | The Avengers | Sherlock | Transformers | Matrix | Titanic | Me Before You | Similarity(i, E) |
|---|---|---|---|---|---|---|---|
| A | 2 | | 2 | 4 | 5 | | NA |
| B | 5 | | 4 | | | 1 | |
| C | | | 5 | | 2 | | |
| D | | 1 | | 5 | | 4 | |
| E | | | 4 | | | 2 | 1 |
| F | 4 | 5 | | 1 | | | NA |

Since user A and F do not share any movie ratings in common with user E, their similarities with user E are not defined in Pearson Correlation. Therefore, we only need to consider user B, C, and D. Based on Pearson Correlation, we can compute the following similarity.

| | The Avengers | Sherlock | Transformers | Matrix | Titanic | Me Before You | Similarity(i, E) |
|---|---|---|---|---|---|---|---|
| A | 2 | | 2 | 4 | 5 | | NA |
| B | 5 | | 4 | | | 1 | 0.87 |
| C | | | 5 | | 2 | | 1 |
| D | | 1 | | 5 | | 4 | -1 |
| E | | | 4 | | | 2 | 1 |
| F | 4 | 5 | | 1 | | | NA |

**Fig 4: User Based Collaborative Filtering**

So now we want to transform our recommendation problem into an optimization problem. The most commonly used measure is the root - mean - square -error (RMSE). The lower the RMSE number, the better the performance.

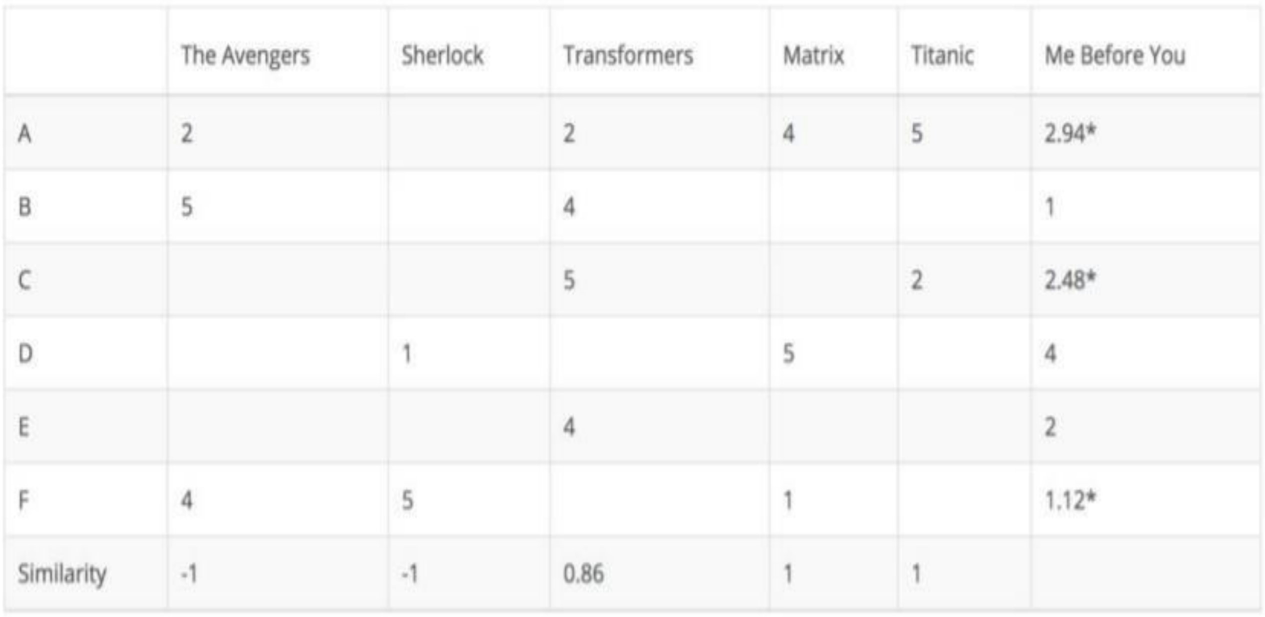
The following are the benefits of collaborative filtering-based systems:

- It is just depending on the content.
- It frequently reads the minds of people who share similar inclinations.
- Make a true quality appraisal of the things.

The following are the disadvantages of collaborative filtering-based systems:

- The most prevalent early rater problem occurs when the collaborative filtering approach fails to offer ratings for a video with no user waiting.

- Sparsity is more prevalent in this type of welding procedure since there are so many null values that it is difficult to discover objects that are assessed by the majority of people.



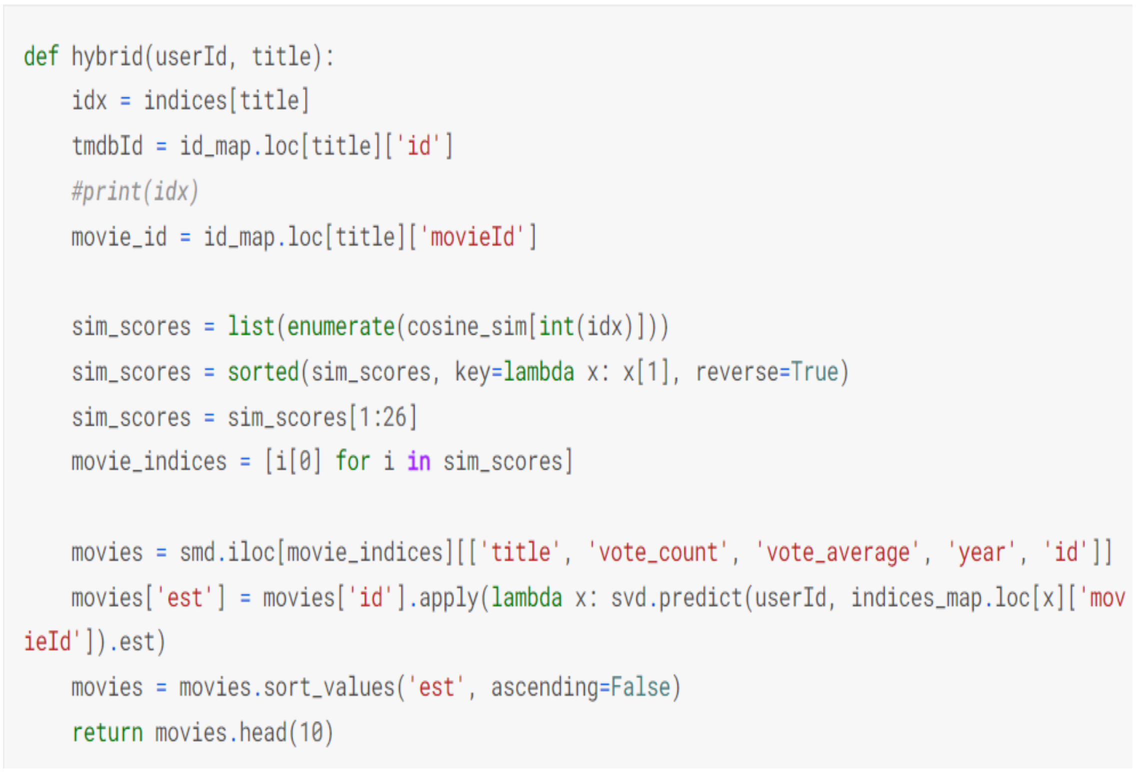
| | The Avengers | Sherlock | Transformers | Matrix | Titanic | Me Before You |
|---|---|---|---|---|---|---|
| A | 2 | | 2 | 4 | 5 | 2.94* |
| B | 5 | | 4 | | | 1 |
| C | | | 5 | | 2 | 2.48* |
| D | | 1 | | 5 | | 4 |
| E | | | 4 | | | 2 |
| F | 4 | 5 | | 1 | | 1.12* |
| Similarity | -1 | -1 | 0.86 | 1 | 1 | |

**Fig 5: Item based Collaborative Filtering**

**Hybrid Based Filtering**

It is basically a hybrid of content-based and collaborative-based filtering approaches, with the input being the user ID and the title of the movie, and the output being a list of comparable movies shorted by the specific users based on the predicted ratings. Estimated ratings are computed internally, where ideas from content and collaborative based filtering are utilized to develop an engine that suggests movies to the specific user, and then ratings are estimated.

In the comparative section below, we will observe how movies are selected using a hybrid filtering strategy that employs both a content-based and a collaborative-based filtering method. [15] It is obvious that the hybrid filtering approach is beneficial in the majority of circumstances and conditions in which it is hard to identify or obtain the precision with which consumers may obtain suggested movies.



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

    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]

    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year', 'id']]
    movies['est'] = movies['id'].apply(lambda x: svd.predict(userId, indices_map.loc[x]['mov
ieId']).est)
    movies = movies.sort_values('est', ascending=False)
    return movies.head(10)
```

**Fig 5: Hybrid Filtering Method**

## 1.5) Organization

The following is a general outline of the report's structure:

**Part 1:** Describes the project's overall presentation, issue proclamation venture points, and scope.

**Part 2:** It provides an overview of current research in the topic. It explains in full all of the research, investigations, theories, and social gatherings that took place throughout the project.

**Part 3:** Discusses the project's framework and plan in order to forecast the correct outcome.

**Part 4:** Discusses the results and provides screenshots.

**Part 5:** Complete the project and submit a proposal for future work.

# CHAPTER 2

# LITERATURE SURVEY

**2.1) Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey**
**Authors:** Dheeraj  Bokde , Sheetal Girase, Debajyoti  Mukopadhyay

**Publisher:** Procedia Computer Science,

**Year:** 2015

This paper attempt presents a comprehensive survey of Factorization model like Singular Value Decomposition address the challenges of Collaborative Filtering algorithms, which can be served as a roadmap for research and practice in this area.

**2.2) Latent Factor Models for Web Recommender Systems**

**Authors:** Bee-Chung Chen, Deepak Agarwal, Pradheep Elango, Raghu Ramakrishnan

**Publisher:** Yahoo! Research & Yahoo! Labs

This paper discusses about the model that

- uses feature-based regression to predict the initial point for online learning, and
- reduces the dimensionality of online learning

Rapidly update online models once new data is received:

- Fast learning: Low dimensional and easily parallelizable
- Online selection for the best dimensionality

## 2.3) Matrix Factorization Techniques for Recommender Systems

**Authors:** Yehuda Koren, Yahoo Research, Robert Bell and Chris Volinsky,
AT&T Labs---Research

**Publisher:** IEEE Computer Society Press Los Alamitos, CA, USA

**Year:** 2009
Used in Netflix Competitions

This paper discusses about matrix factorization techniques have become a dominant methodology within collaborative filtering recommenders.

Experience with datasets such as the Netflix Prize data has shown that they deliver accuracy superior to classical nearest-neighbour techniques. At the same time, they offer a compact memory-efficient model that systems can learn relatively easily.
What makes these techniques even more convenient is that models can integrate naturally many crucial aspects of the data, such as multiple forms of feedback, temporal dynamics, and confidence levels.

## 2.4) Related Articles and Research Papers

- In December 2017, Springer US published a research article titled Recommendation System USING K-MEANS CLUSTERING on "Active Learning in Recommender Systems." Lior Rokach and Bracha Shapira were the authors of the paper.

- A research article titled "A content-based recommender system for computer science papers" was published in IJCRT in 2018.] are the writers of the articles. R.C. Guan, D.H. Wang, Y.C. Liang, D.Xu, X.Y. Feng

- A study titled MOVREC utilising Machine Learning Techniques was published in SAIT. Ashrita Kashyap, Sunita. B, Sneha Srivastava, Aishwarya. PH, and Anup Jung Shah are the authors of the article.

- Lopez-Nores, M., J. Garca-Duque, A. Fernandez- Vilas, R. P. Daz-Redondo, J. Bermejo-Mu noz, “A Flexible Semantic Inference Metho-dology to Reason about User Preferences in Knowledge-based Recommender Systems”, Knowledge-Based Systems, Vol.21, No.4(2016), 305～32

- Gurpreet Singh, International Journal of Advanced Trends in Computer Applications (IJATCA)Volume 3, Number 7, July - 2016, pp. 11-14 ISSN: 2395-3519

# CHAPTER 3

# SYSTEM DESIGN

## 3.1) Dataset

**For collaborative based and content-based filtration:**

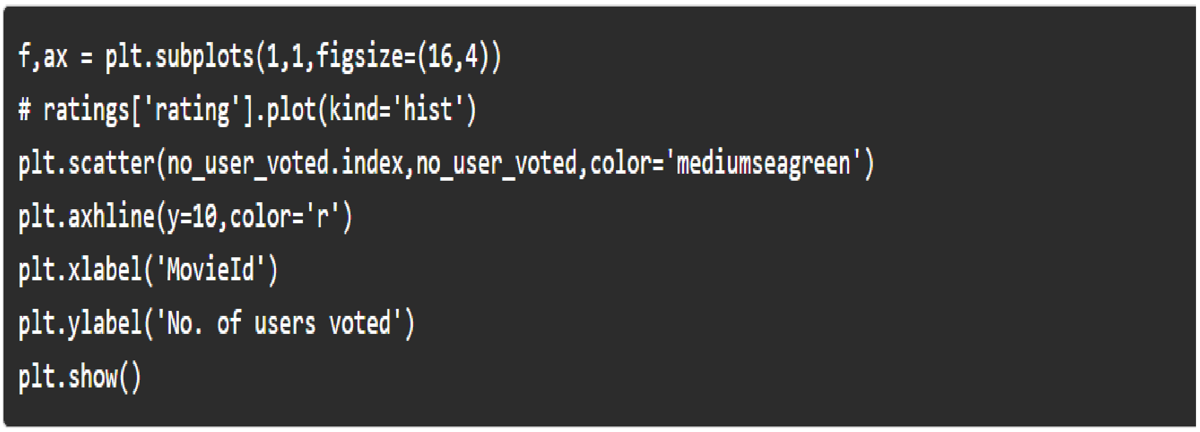
The dataset has been taken from Kaggle. It is used as the standard dataset by the movie recommendation system.

1) Movie dataset is used in “content-based movie recommendation system” in this project.
2) The ratings and movies are taken in account.
3) Total number of movies.
4) Total number of ratings in dataset.
5) The Movie Wens users were chosen randomly
6) An uniquid is assigned to every movie and the user.

**The Hybrid Filtering Method**

It is made of 25,000,000 ratings and 700,000 tag applications is applied to 50,000 movies by 250,000 users.

## 3.2) Visualizing the no. of users Voted



```
f,ax = plt.subplots(1,1,figsize=(16,4))
# ratings['rating'].plot(kind='hist')
plt.scatter(no_user_voted.index,no_user_voted,color='mediumseagreen')
plt.axhline(y=10,color='r')
plt.xlabel('MovieId')
plt.ylabel('No. of users voted')
plt.show()
```

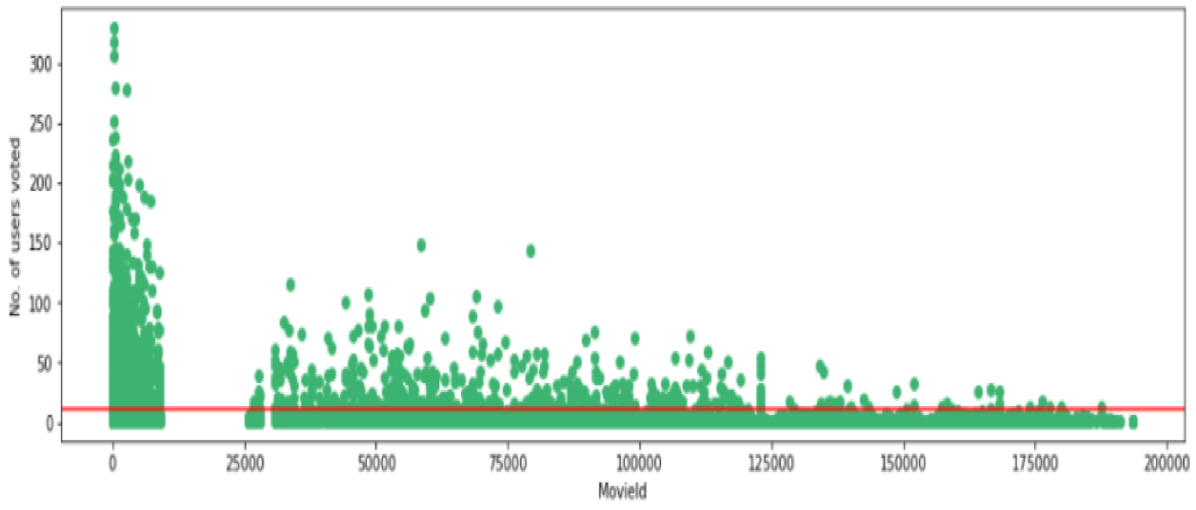
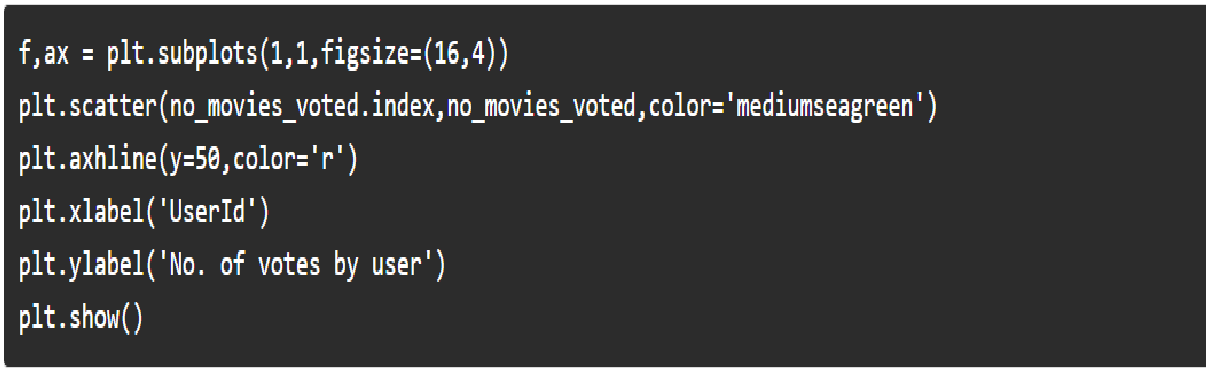


**Fig 6: Visualization of the no. of Users Voted**

## 3.3) Visualizing the no. of Votes by User



```
f,ax = plt.subplots(1,1,figsize=(16,4))
plt.scatter(no_movies_voted.index,no_movies_voted,color='mediumseagreen')
plt.axhline(y=50,color='r')
plt.xlabel('UserId')
plt.ylabel('No. of votes by user')
plt.show()
```

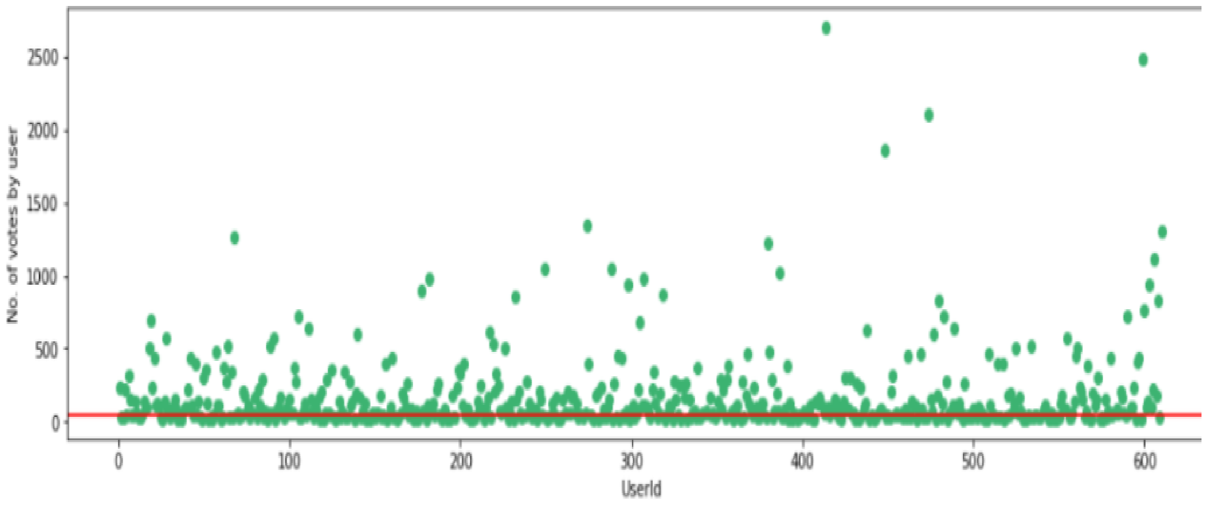


**Fig 7: Visualizing the no. of Votes by Users**

## 3.4) Algorithms Used

### Cosine Similarity

Cosine Similarity is defined as the cosine of angle between two vectors where they are non-zero and inner product space id defined as the dot product of the two vectors divide by the product of Euclidian magnitude. Because the angle is lower in the case of cosine similarity, the similarity is significantly more than the equilibrium distance.

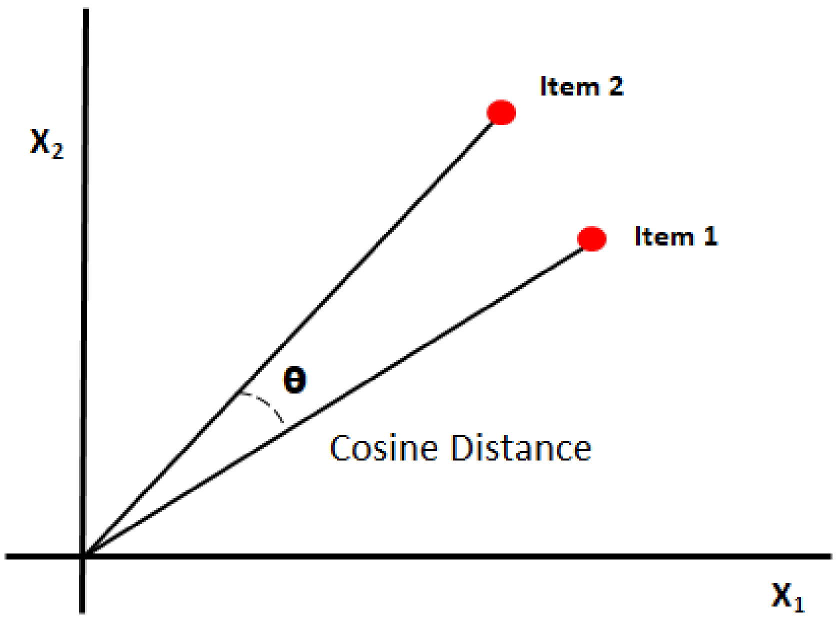


**Fig-8: Cosine Similarity**

## K means Algorithm

K means clustering algorithm denotes a clustering approach that simply creates a cluster within a cluster that has identical matching attributes. The degree of closeness describes the similarity foundation as the degree to which two points are connected to each other. In this approach, the centroid is re-simplified and the procedure is repeated until the optimal centroid is computed or found. It just iteratively gets the optimal value for the K Centre points and then assigns each data point to the nearest centre of K value. The number of clusters discovered from the data is simply represented by the letter 'K.' Even with very little knowledge about the data, a simple unsupervised ML system can classify data points into subgroups.

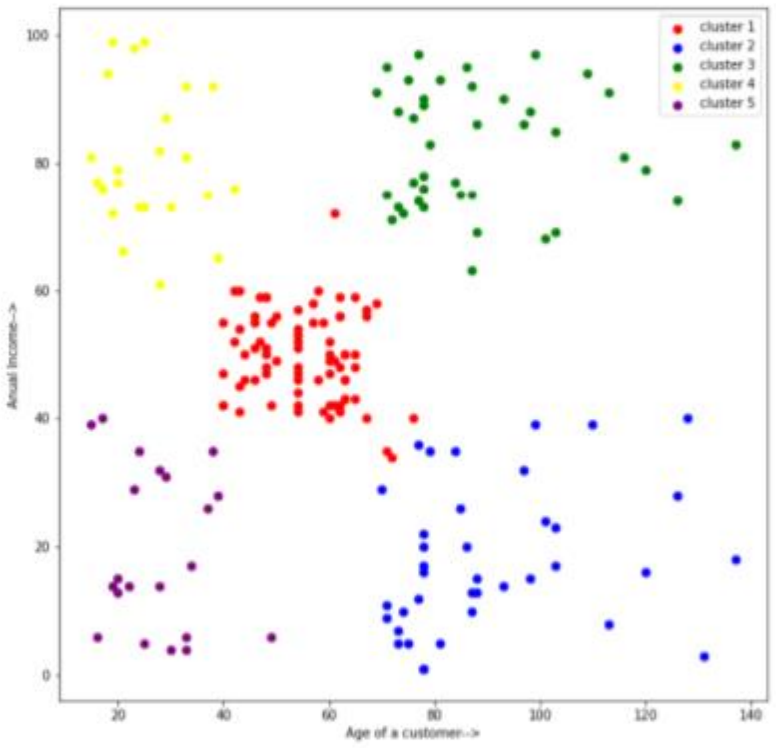


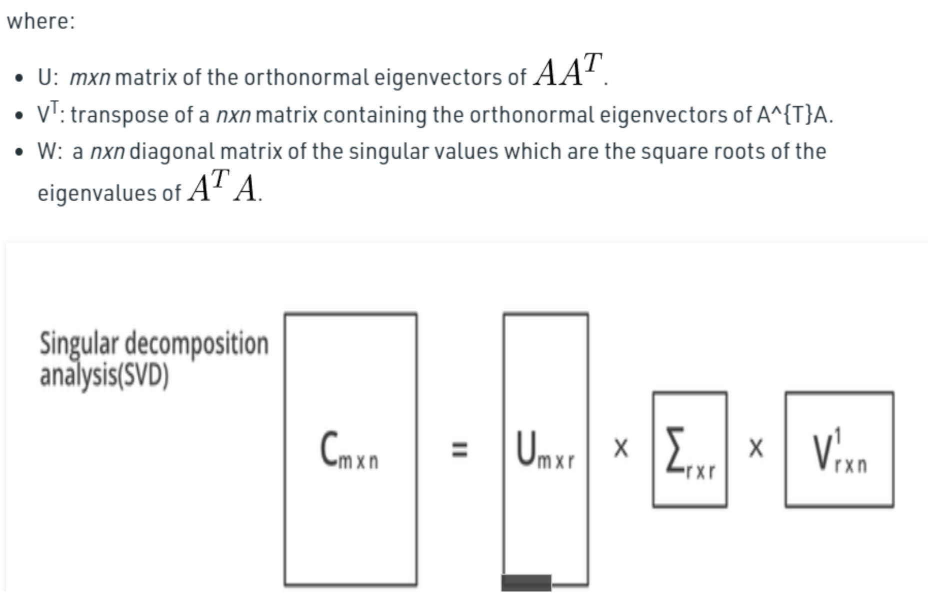
**Fig 9: K-Mean Algorithm**

**Singular Value Decomposition (SVD)**

SVD is the matrix factorization of a matrix into three matrices. It holes qualities and conveys some geometrical as well as theoretical outputs in a mathematically represented linear transformation.

The formula for the SVD far a given matrix is given by:

A=UWV^T



where:

- U: *mxn* matrix of the orthonormal eigenvectors of $AA^T$.
- V$^T$: transpose of a *nxn* matrix containing the orthonormal eigenvectors of A^{T}A.
- W: a *nxn* diagonal matrix of the singular values which are the square roots of the eigenvalues of $A^TA$.

Singular decomposition analysis(SVD)
$C_{m\,x\,n} = U_{m\,x\,r} \times \sum_{r\,x\,r} \times V^1_{r\,x\,n}$

**Fig 10: SVD Algorithm**

**The RMSE (Root Mean Square Error)**

The root mean square error (RMSE) is just the standard deviation of the expected errors. The residuals are the measure of the regression where the data points are, but it also displays the dispersion of the residuals in the data points and determines the best fit in the data. It is also utilized in forecasting and regression analysis to get confirmed experimental results. The lower the RMSE number, the better the performance.

$$\mathbf{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$$

| | |
|---|---|
| $\mathbf{RMSD}$ | = root-mean-square deviation |
| $i$ | = variable i |
| $N$ | = number of non-missing data points |
| $x_i$ | = actual observations time series |
| $\hat{x}_i$ | = estimated time series |

## 3.5) Hardware and Software Requirements

The Following hardware and softwares were required for this project:

- 4.2 GB RAM
- MS Window 7 and above Software Requirements
- Jupyter  Notebook
- Wamdp Server
- Visual Studio Code
- Sublime Text
- MYSQL

## 3.6) Concepts Requirements

- Machine Learning Algorithms
- Data Pre-processing Functions and tools
- scikit-learn
- seaborn
- knowledge of K-Means clustering
- NumPy is a Python programming language.
- Panda bears
- matplotlib (matplotlib)
- Cleaning of data
- 64bit processors are required

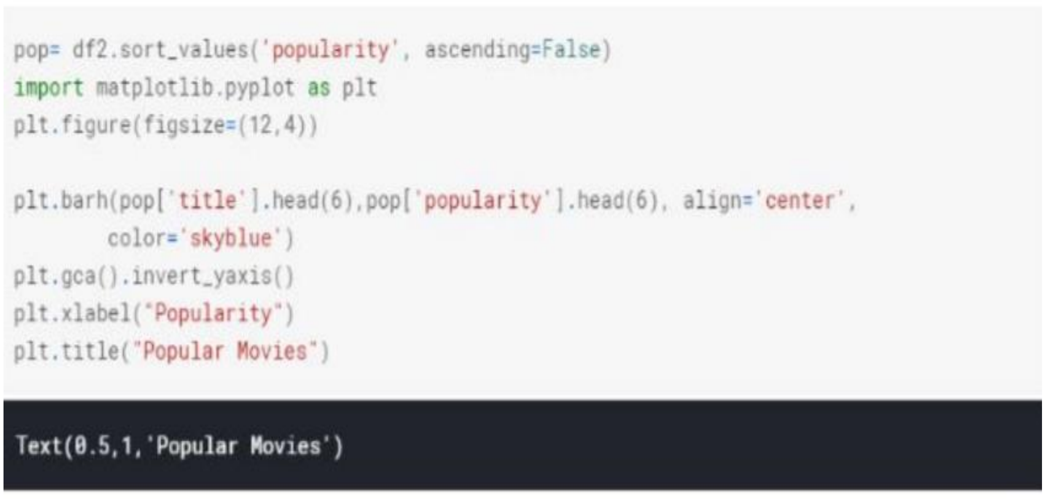
# CHAPTER 4

# PERFORMANCE ANALYSIS

## 4.1) Comparisons and Results

**Demographic Filtering**

Filtering the cause of short in the movies recommended which are best to the users based on the metric scores and personalized and generalized recommendations are recommended to every user on the basis of the popularity which are generally like by the average audience.



```
pop= df2.sort_values('popularity', ascending=False)
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))

plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center',
        color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("Popular Movies")
```

```
Text(0.5,1,'Popular Movies')
```

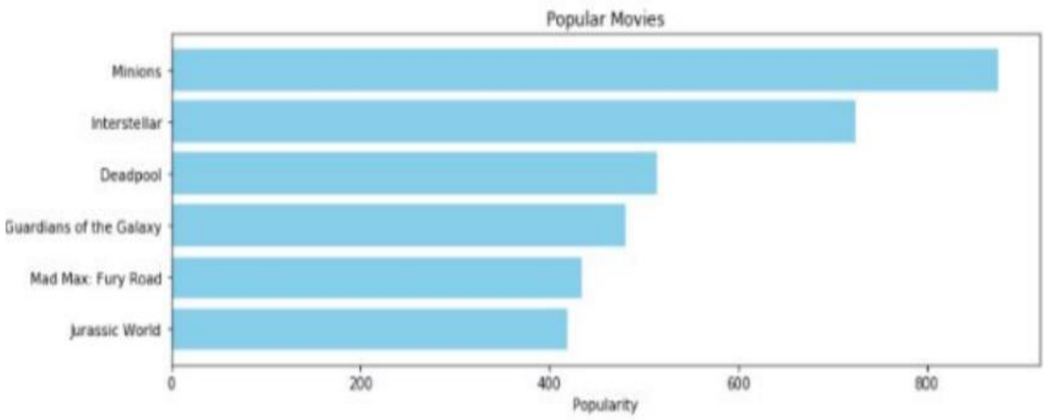
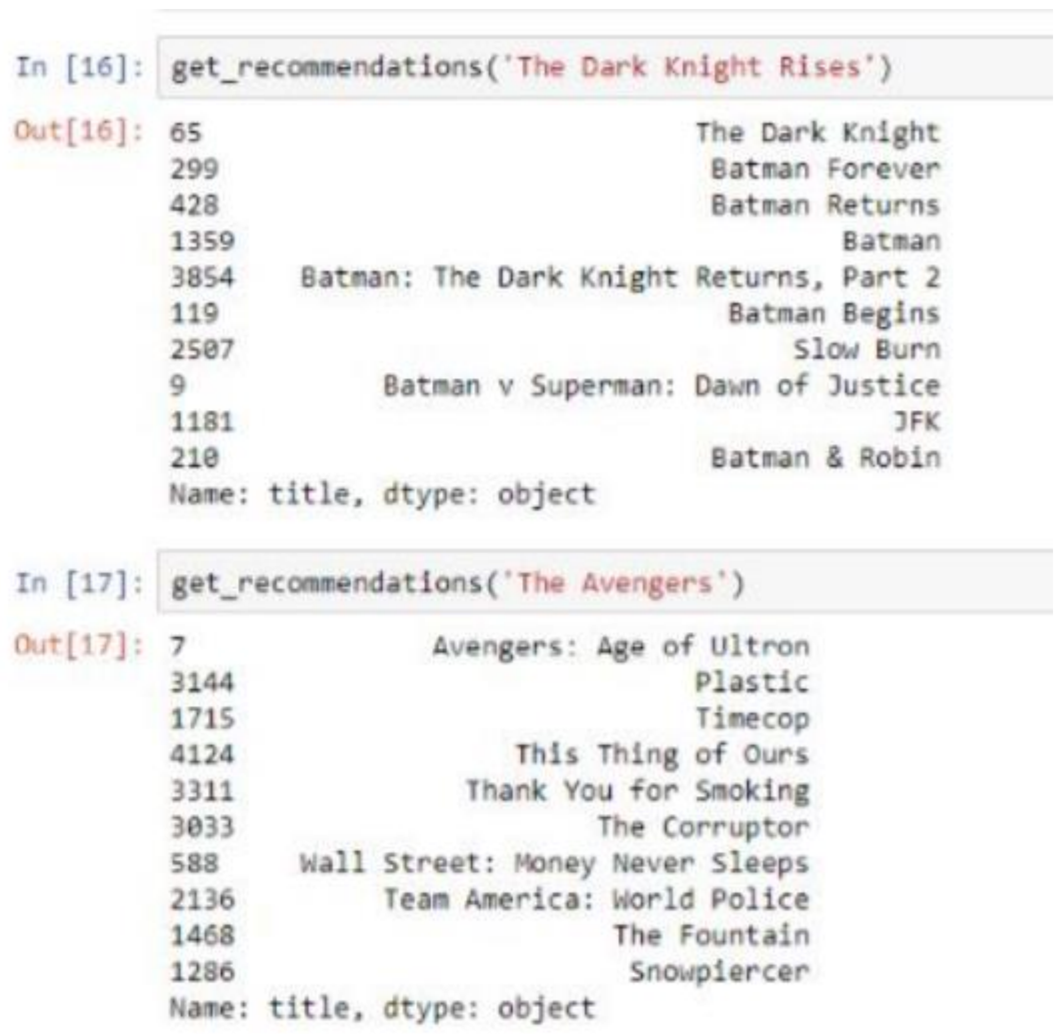


**Fig11: Demographic Filtering Output**

**Content Based Filtering**

User profile holds the content that is much more matching to use the form of the features. The previous actions or for the feedback is taken into account a generally takes into account the description of the content that has been edited by the users of different choices.



```
In [16]: get_recommendations('The Dark Knight Rises')

Out[16]: 65                                   The Dark Knight
         299                                   Batman Forever
         428                                   Batman Returns
         1359                                          Batman
         3854    Batman: The Dark Knight Returns, Part 2
         119                                    Batman Begins
         2507                                       Slow Burn
         9             Batman v Superman: Dawn of Justice
         1181                                             JFK
         210                                   Batman & Robin
         Name: title, dtype: object

In [17]: get_recommendations('The Avengers')

Out[17]: 7                 Avengers: Age of Ultron
         3144                              Plastic
         1715                              Timecop
         4124                   This Thing of Ours
         3311                Thank You for Smoking
         3033                        The Corruptor
         588        Wall Street: Money Never Sleeps
         2136           Team America: World Police
         1468                         The Fountain
         1286                          Snowpiercer
         Name: title, dtype: object
```

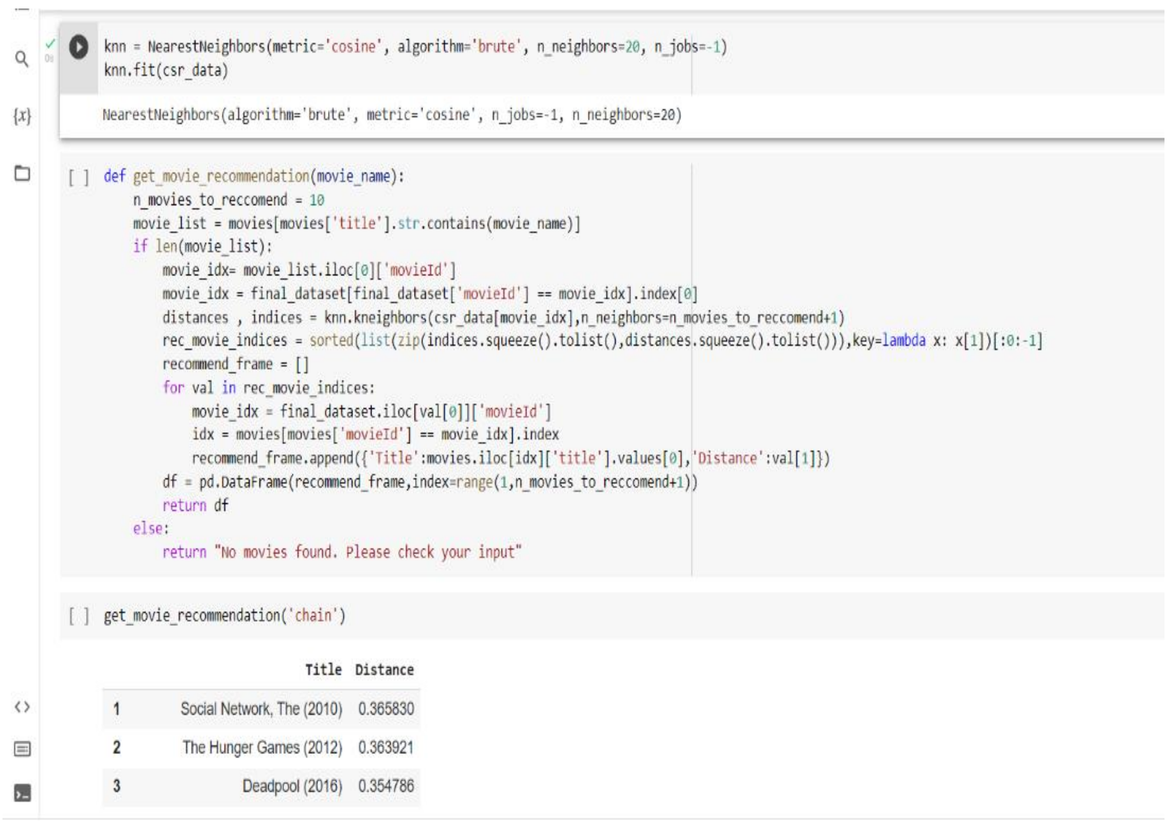
**Fig 12: Content Based Filtering Output**

## Collaborative based Filtering

In the collaborative filtering behaviour used here item based collaborative filtering where we have taken 3 different types of metrics and varied the results accordingly. [2] Brief comparison of three of the metrics used in the collaborative filtering are shown with the movies recommended from them based on the bounds set to the number of users and a number of ratings by a user to a movie.

### Metric =”Cosine”

Cosine similarity, or the cosine kernel, computes similarity as the normalized dot product of X and Y: K(X, Y) = <X, Y> / (||X||*||Y||) On L2-normalized data, this function is equivalent to linear_kernel.



```
knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20, n_jobs=-1)
knn.fit(csr_data)
```

```
NearestNeighbors(algorithm='brute', metric='cosine', n_jobs=-1, n_neighbors=20)
```

```
def get_movie_recommendation(movie_name):
    n_movies_to_reccomend = 10
    movie_list = movies[movies['title'].str.contains(movie_name)]
    if len(movie_list):
        movie_idx= movie_list.iloc[0]['movieId']
        movie_idx = final_dataset[final_dataset['movieId'] == movie_idx].index[0]
        distances , indices = knn.kneighbors(csr_data[movie_idx],n_neighbors=n_movies_to_reccomend+1)
        rec_movie_indices = sorted(list(zip(indices.squeeze().tolist(),distances.squeeze().tolist())),key=lambda x: x[1])[:0:-1]
        recommend_frame = []
        for val in rec_movie_indices:
            movie_idx = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_idx].index
            recommend_frame.append({'Title':movies.iloc[idx]['title'].values[0],'Distance':val[1]})
        df = pd.DataFrame(recommend_frame,index=range(1,n_movies_to_reccomend+1))
        return df
    else:
        return "No movies found. Please check your input"
```

```
get_movie_recommendation('chain')
```

| | Title | Distance |
|---|---|---|
| 1 | Social Network, The (2010) | 0.365830 |
| 2 | The Hunger Games (2012) | 0.363921 |
| 3 | Deadpool (2016) | 0.354786 |

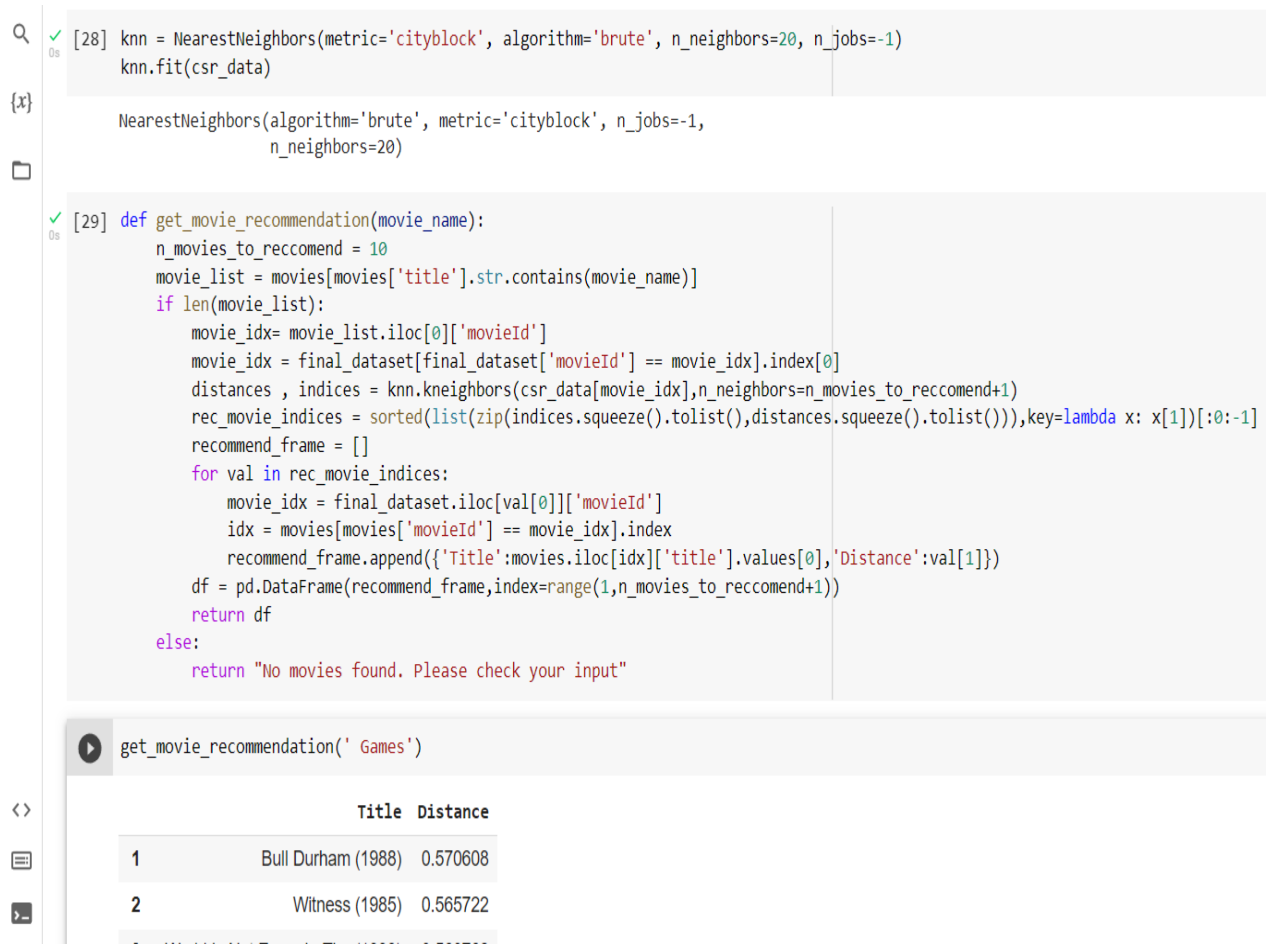
**Fig 13: Collaborative Based Filtering Method (Metric =Cosine) Output**

**Metric=”Cityblock”->**

This function simply returns the valid pairwise distance metrics. It exists to allow for a description of the mapping for each of the valid strings.

The function for the cityblock is as below:

‘cityblock’ =metrics.pairwise.manhattan_distances



```
[28] knn = NearestNeighbors(metric='cityblock', algorithm='brute', n_neighbors=20, n_jobs=-1)
     knn.fit(csr_data)
```

```
NearestNeighbors(algorithm='brute', metric='cityblock', n_jobs=-1,
                 n_neighbors=20)
```

```
[29] def get_movie_recommendation(movie_name):
         n_movies_to_reccomend = 10
         movie_list = movies[movies['title'].str.contains(movie_name)]
         if len(movie_list):
             movie_idx= movie_list.iloc[0]['movieId']
             movie_idx = final_dataset[final_dataset['movieId'] == movie_idx].index[0]
             distances , indices = knn.kneighbors(csr_data[movie_idx],n_neighbors=n_movies_to_reccomend+1)
             rec_movie_indices = sorted(list(zip(indices.squeeze().tolist(),distances.squeeze().tolist())),key=lambda x: x[1])[:0:-1]
             recommend_frame = []
             for val in rec_movie_indices:
                 movie_idx = final_dataset.iloc[val[0]]['movieId']
                 idx = movies[movies['movieId'] == movie_idx].index
                 recommend_frame.append({'Title':movies.iloc[idx]['title'].values[0],'Distance':val[1]})
             df = pd.DataFrame(recommend_frame,index=range(1,n_movies_to_reccomend+1))
             return df
         else:
             return "No movies found. Please check your input"
```

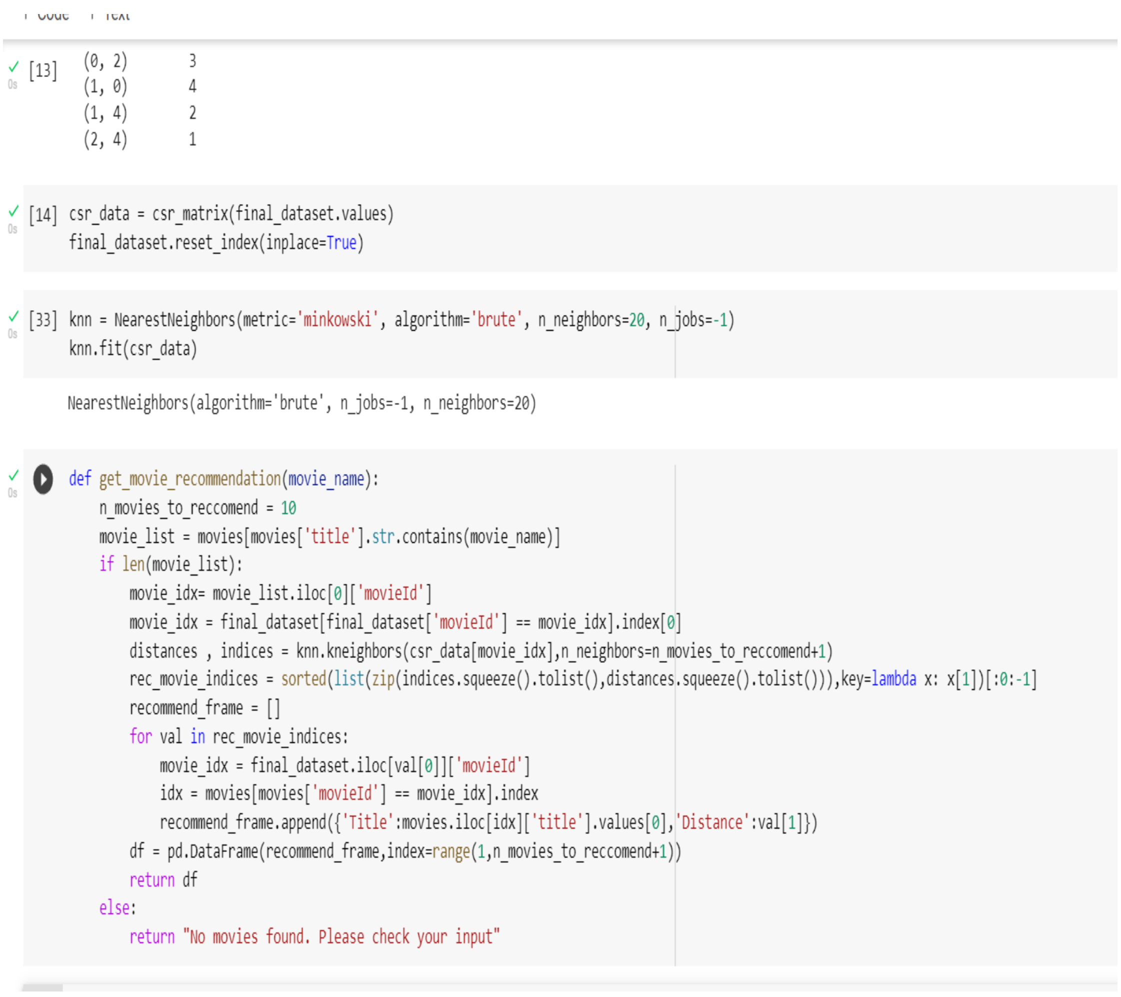
```
get_movie_recommendation(' Games')
```

```
               Title  Distance
1  Bull Durham (1988)  0.570608
2      Witness (1985)  0.565722
```

**Fig 14: Collaborative Based Filtering Method (Metric =Cityblock) Output**

**Metric=”Minkowski”**

It is a metric intended for real-valued vector spaces. We can calculate Minkowski distance only in a normed vector space, which means in a space where distances can be represented as a vector that has a length and the lengths cannot be negative.



```
[13]  (0, 2)        3
      (1, 0)        4
      (1, 4)        2
      (2, 4)        1
```

```
[14] csr_data = csr_matrix(final_dataset.values)
     final_dataset.reset_index(inplace=True)
```

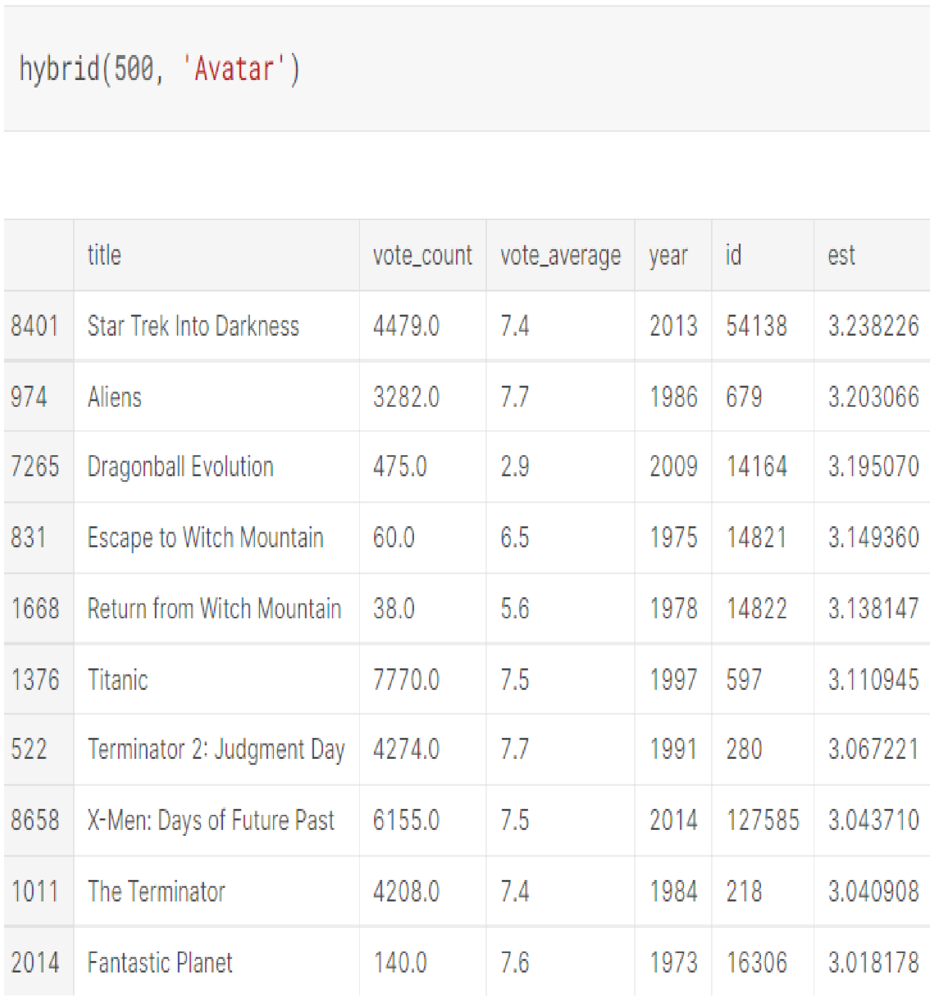
```
[33] knn = NearestNeighbors(metric='minkowski', algorithm='brute', n_neighbors=20, n_jobs=-1)
     knn.fit(csr_data)

     NearestNeighbors(algorithm='brute', n_jobs=-1, n_neighbors=20)
```

```
def get_movie_recommendation(movie_name):
    n_movies_to_reccomend = 10
    movie_list = movies[movies['title'].str.contains(movie_name)]
    if len(movie_list):
        movie_idx= movie_list.iloc[0]['movieId']
        movie_idx = final_dataset[final_dataset['movieId'] == movie_idx].index[0]
        distances , indices = knn.kneighbors(csr_data[movie_idx],n_neighbors=n_movies_to_reccomend+1)
        rec_movie_indices = sorted(list(zip(indices.squeeze().tolist(),distances.squeeze().tolist())),key=lambda x: x[1])[:0:-1]
        recommend_frame = []
        for val in rec_movie_indices:
            movie_idx = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_idx].index
            recommend_frame.append({'Title':movies.iloc[idx]['title'].values[0],'Distance':val[1]})
        df = pd.DataFrame(recommend_frame,index=range(1,n_movies_to_reccomend+1))
        return df
    else:
        return "No movies found. Please check your input"
```

**Fig 15: Collaborative Based Filtering Method (Metric =Minkowski) Output**

## Hybrid Based filtering

It is simply a mixture of content based filtering and collaborative based filtering methods where we will take the input as the userid and the title of the movie and the output will be e the similar movies shorted by the particular users based on the expected ratings. [10] Expected ratings are calculated internally where the ideas from content and collaborative filtering are used to build an engine where movies are suggested to the particular user and then estimation of the ratings takes place.



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

| | title | vote_count | vote_average | year | id | est |
|---|---|---|---|---|---|---|
| 8401 | Star Trek Into Darkness | 4479.0 | 7.4 | 2013 | 54138 | 3.238226 |
| 974 | Aliens | 3282.0 | 7.7 | 1986 | 679 | 3.203066 |
| 7265 | Dragonball Evolution | 475.0 | 2.9 | 2009 | 14164 | 3.195070 |
| 831 | Escape to Witch Mountain | 60.0 | 6.5 | 1975 | 14821 | 3.149360 |
| 1668 | Return from Witch Mountain | 38.0 | 5.6 | 1978 | 14822 | 3.138147 |
| 1376 | Titanic | 7770.0 | 7.5 | 1997 | 597 | 3.110945 |
| 522 | Terminator 2: Judgment Day | 4274.0 | 7.7 | 1991 | 280 | 3.067221 |
| 8658 | X-Men: Days of Future Past | 6155.0 | 7.5 | 2014 | 127585 | 3.043710 |
| 1011 | The Terminator | 4208.0 | 7.4 | 1984 | 218 | 3.040908 |
| 2014 | Fantastic Planet | 140.0 | 7.6 | 1973 | 16306 | 3.018178 |

**Fig 16: Hybrid Based Filtering Output**

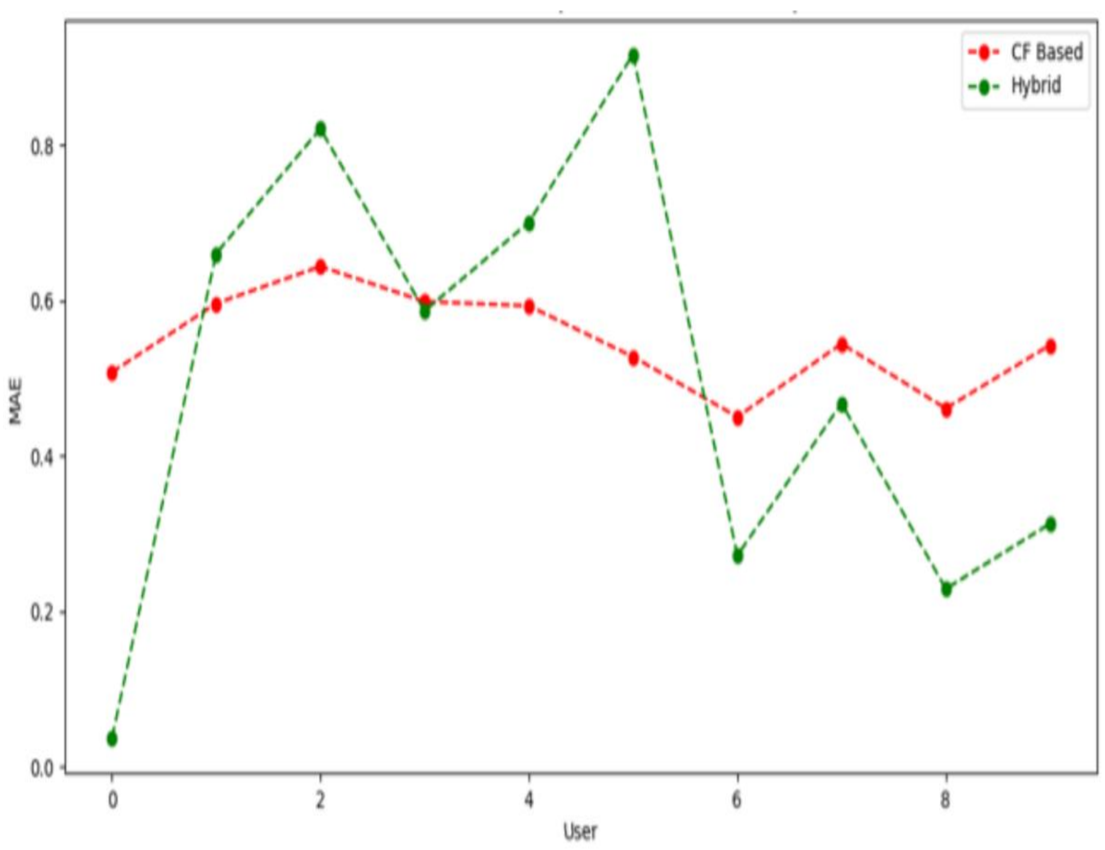


**Fig 17: Comparative output**

Here we can see that the hybrid filtering technique stands good in in overcoming the issues faced in the content-based filtering technique and the collaborative-based filtering method we can generalize from the method of root mean square error that the value for hybrid filtering method is less so performance is higher for hybrid case.

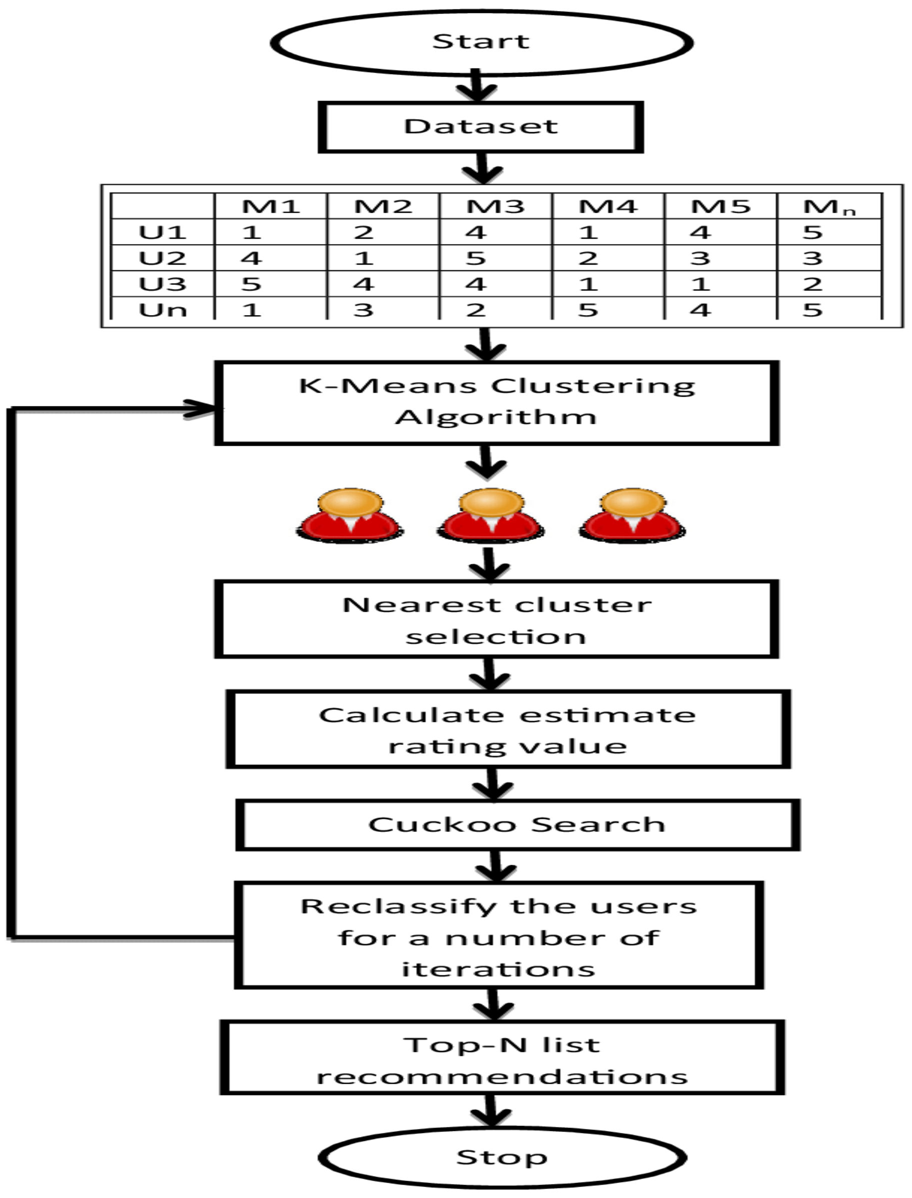
While we can say that collaborative filtering technique stands good only in terms of the quality perspective but when it comes to both qualitative and quantitative achievement of the result will prefer hybrid filtering technique where the all flaws.

While content-based filtering technique only outperform the collaborative in terms of similarity e the collaborative filtering technique can you recommend one item to the other item of the similar interest, the overall flaws can be removed by the hybrid based collaborative filtering with two or more examination techniques are combined to gain the better performance with the less possibilities of drawback of this system.

In general, in case of hybrid filtering techniques the collaborative filtering technique is combined with some other type of filtering technique to avoid the ramp up problem and thus it outperforms the major drawbacks of the system in case if we prefer to use single content based or collaborative filtering technique.

So, hybrid filtering recommender simply allows the user to select his own choices from a given data which contain some attributes or some set of values which contain user specific values and then recommend then the best movie which is based on the similarities based calculating the accumulator weight and then applies the algorithm which is in our case K mean algorithm. [13] Expected ratings are calculated internally where the ideas from content and collaborative filtering are used to build an engine where movies are suggested to the particular user and then estimation of the ratings takes place.

So, in the process of getting different results from different algorithms and techniques hybrid approach is preferred to be better one between the content and collaborative filtering techniques which simply overcomes the drawbacks of the single algorithm and then tries to improve the performance of the overall recommender system. Moreover, some other techniques like classification clustering can be used to get the best of the recommendations which would simply increase our accuracy for the recommender system. So, the better performance can be achieved in the end by a hybrid-based filtering technique which is why it is most preferable over the other two techniques.


Start
Dataset
K-Means Clustering Algorithm
Nearest cluster selection
Calculate estimate rating value
Cuckoo Search
Reclassify the users for a number of iterations
Top-N list recommendations
Stop

| | M1 | M2 | M3 | M4 | M5 | $M_n$ |
|---|---|---|---|---|---|---|
| U1 | 1 | 2 | 4 | 1 | 4 | 5 |
| U2 | 4 | 1 | 5 | 2 | 3 | 3 |
| U3 | 5 | 4 | 4 | 1 | 1 | 2 |
| Un | 1 | 3 | 2 | 5 | 4 | 5 |

**Fig 18: System Framework**

# CHAPTER 5

# CONCLUSION

So, for implementing a hybrid technique for content and collaborative based filtering we take into account the hybrid approach which improves the overall performance of the system and then recommended movies to the users as per the choice in a much better way than the other two system of recommendation lower the mean average error, it further increases the accuracy of the recommender system and then we can use h system of recommendation for future uses as well in a better way. We also have some system computational bounds or limitations to perform the recommender system on the large dataset here but we have done enough to distinguish between the various recommender system which finally put hybrid system of recommendation on the top of the all.

Hence, we can conclude that hybrid-based filtering helps in getting the system fragmentation much efficient enhance the Precision of the overall system and no doubt it is the mixture of both content in collaborative-based filtering methods where even if one method fails The Other takes over and maintains the overall accuracy of the system and simply increase the performance overall all around.

Overall flaws can be removed by the hybrid based collaborative filtering with two or more examination techniques are combined to gain the better performance with the less possibilities of drawback of this system. In general, in case of hybrid filtering techniques the collaborative filtering technique is combined with some other type of filtering technique to avoid the ramp up problem and thus it outperforms the major drawbacks of the system in case if we prefer to use single content based or collaborative filtering technique.While we can say that collaborative filtering technique stands good only in terms of the quality perspective but when it comes to both qualitative and quantitative achievement of the result will prefer hybrid filtering technique where the all flaws. In the end hybrid system stands alone the better performer for Recommending movies to the users of different taste, choices or similarities.

# FUTURE WORK

In case of content-based filtering method we can look up on the cast and crew also where we have only considered the genre and also, we can see at the movies are compatible or not. Comparison of collaborative filtering-based approaches and different kind of similarity measurements would be a good one for the recommender system . We can use matrix factorization for calculating the number of factors involved. We can also apply deep learning techniques to for the enhance the recommender system and optimising the efficiency of the system. We can work on different areas such as video some books aur even recommending some songs to the users of the mobile phones based on the platforms of the different apps available on the Play Store. Various techniques such as clustering classification can be used to get the better version of our recommender system which for the enhance the accuracy of the overall model**.**

# REFERENCES

(1) Hirdesh Shivhare, Anshul Gupta and Shalki Sharma, "Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure" Communication and Control. IEEE International Conference on Computer, 2015.

(2) Manoj Kumar, D.K. Yadav, Ankur Singh and Vijay Kr. Gupta. "A Movie Recommender System: MOVREC, International Journal of Computer Applications (0975-8887) Volume 124-No.3, 2015.

(3) RyuRi Kim, Ye Jeong Kwak, HyeonJeong Mo, Mucheol Kim, Seungmin Rho,Ka Lok Man, Woon Kian Chong "Trustworthy Movie Recommender System with Correct Assessment and Emotion Evaluation". Proceedings of the International MultiConference of Engineers and Computer Scientists Vol II, 2015

(4) Zan Wang, Xue Yu*, Nan Feng, Zhenhua Wang, "An Improved Collaborative Movie Recommendation System Using Computational Intelligence" Journal of Visual Languages & Computing. Volume 25, Issue 6, 2014

(5) Debadrita Roy, Arnab Kundu, "Design of Movie Recommendation System by Means of Collaborative Filtering". International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 4, 2013.

(6) E. Amolochitis, Algorithms and applications for academic search, recommendation and quantitative association rule mining. Gistrup, Denmark: River Publishers, 2018.

(7) M. Jalali, H. Gholizadeh and S. Hashemi Golpayegani, "An improved hybrid recommender system based on collaborative filtering, content based, and demographic filtering", International Journal of Academic Research, vol. 6, no. 6, pp. 22-28, 2014.

(8) C. Li and K. He, "CBMR: An optimized MapReduce for item-based collaborative filtering recommendation algorithm with empirical analysis", Concurrency and Computation: Practice and Experience, vol. 29, no. 10, p. e4092, 2017.

(9) Y. Ng, "MovRec: a personalized movie recommendation system for children based on online movie features", International Journal of Web Information Systems, vol. 13, no. 4, pp. 445-470, 2017.

(10) A. Roy and S. Ludwig, "Genre based hybrid filtering for movie recommendation engine", Journal of Intelligent Information Systems, vol. 56, no. 3, pp. 485-507, 2021.

(11) N. Shahabi and F. Najian, "A New Strategy in Trust-Based Recommender System using K-Means Clustering", International Journal of Advanced Computer Science and Applications, vol. 8, no. 9, 2017.

(12) S. Agrawal and P. Jain, "About Performance Evaluation of the Movie Recommendation Systems", International Journal of Computer Applications, vol. 158, no. 2, pp. 7-10, 2017.

(13) Y. Patil and D. Karandam, "COLLABORATIVE FILTERING APPROACHES FOR MOVIE RECOMMENDATION SYSTEM USING PROBABILISTIC RELATIONAL MODEL", International Journal of Advance Engineering and Research Development, vol. 2, no. 03, 2015.

(14) P. Sharma and L. Yadav, "MOVIE RECOMMENDATION SYSTEM USING ITEM BASED COLLABORATIVE FILTERING", International Journal of Innovative Research in Computer Science &amp; Technology, vol. 8, no. 4, 2020.

(15) T. Brodt, Collaborative Filtering. Saarbrücken: VDM, Müller, 2010.

(16) M. Gogri, D. Chheda and V. Solani, "Movie Recommendation Using Deep Learning with Hybrid Approach", Aksh - The Advance Journal, vol. 1, no. 2, pp. 1-4, 2020. Available: 10.51916/aksh.2020.v01i02.001.