INSTITUTE FOR ADVANCED COMPUTING

AND SOFTWARE DEVELOPMENT AKURDI,

PUNE

Documentation On

"Movie Recommendation System Using Machine Learning"

PG-DBDA SEPT 2023

Submitted By:

Group No: 20

Roll No.	Name
239528	Nayana Patil
239545	Vaishnavi Chakkarwar

Dr. Shantanu Pathak
Project Guide

Mr. Rohit Puranik
Centre Coordinator

ABSTRACT

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

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who has contributed to the completion of our project.

First and foremost, we would like to thank our project guide **Dr.Shantanu Pathak.** Mam for their constant guidance and support throughout the project. We extend our sincere thanks to our respected centre coordinator, **Mr. Rohit Puranik** for allowing us to use the facilities available.

We would also like to express our appreciation to the faculty members of our department for their constructive feedback and encouragement. Their insights and suggestions have helped us to refine our ideas and announce the quality of our work.

Furthermore, we would like to thank our families and friends for their unwavering support and encouragement throughout our academic journey, their love and support have been a constant source of motivation and inspiration for us.

Thank you for all your valuable contributions to our project.

Nayana Patil (239528)

Vaishnavi Chakkarwar (239545)

Table of Contents

TABLE OF FIGURE:

1. INTRODUCTION

1.1 Problem Statement

Movie Recommendation System Using Machine Learning.

1.2 Introduction

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggests based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities. These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future. Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films.

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

A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience. scalability issues.

1.3 Product Scope

The objective of this project is to provide accurate movie recommendations to users. The goal of the project is to improve the quality of movie recommendation system, such as accuracy, quality and scalability of system than the pure approaches. This is done using Hybrid approach by combining content based filtering and collaborative filtering, To eradicate the overload of the data, recommendation system is used as information filtering tool in social networking sites .Hence, there is a huge scope of exploration in this field for improving scalability, accuracy and quality of movie recommendation systems Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffers with poor recommendation quality and scalability issues.

1.4 Aim & Objectives

- Improving the Accuracy of the recommendation system
- Improve the Quality of the movie Recommendation system
- Improving the Scalability.
- Enhancing the user experience.

1.5 Methodology for Movie Recommendation

The hybrid approach proposed an integrative method by merging fuzzy k-means clustering method and genetic algorithm based weighted similarity measure to construct a movie recommendation system. The proposed movie recommendation system gives finer similarity metrics and quality than the existing Movie recommendation system but the computation time which in taken by the proposed recommendation system is more than the existing

recommendation system. This problem can be fixed by taking the clustered data points as an input dataset.

1.6 Agile Methodology

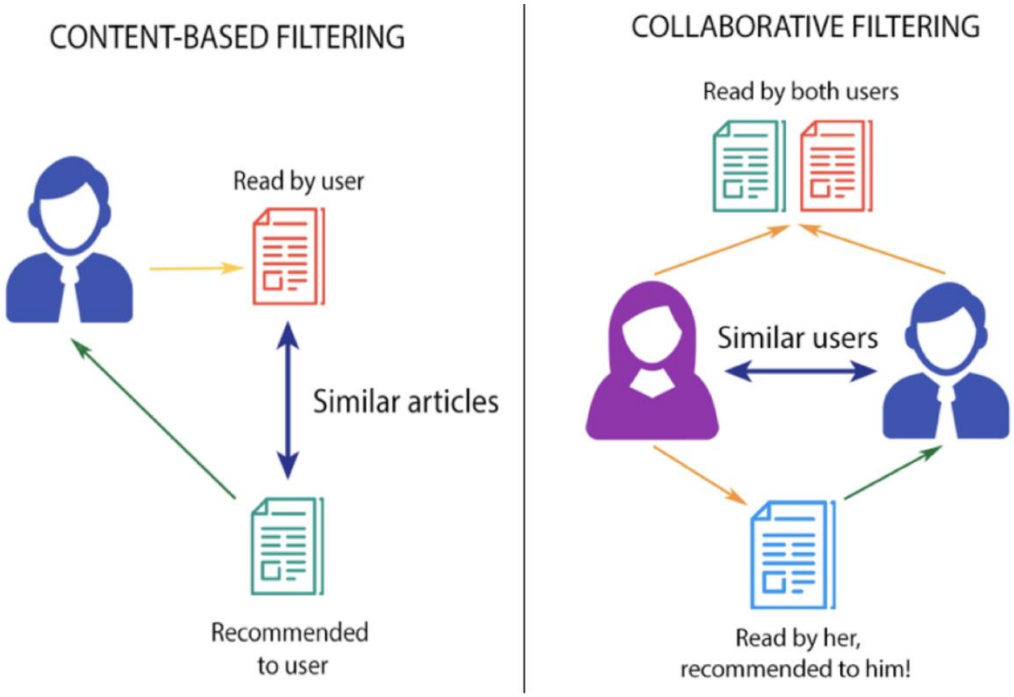
1. **Collecting the data sets**: Collecting all the required data set from Kaggle web site.in this project we require movie.csv, ratings.csv, users.csv.
2. **Data Analysis**: make sure that that the collected data sets are correct and analysing the data in the csv files. i.e. checking whether all the column Felds are present in the data sets.
3. **Algorithms**: in our project we have only two algorithms one is cosine similarity and other is single valued decomposition are used to build the machine learning recommendation model.
4. **Improvements in the project**: In the later stage we can implement different algorithms and methods for better recommendation.

2. LITERATURE SURVEY

2.1 Movie Recommendation System Using Collaborative Filtering

Collaborative filtering systems analyse the user's behaviour and preferences and predict what they would like based on similarity with other users. There are two kinds of collaborative filtering systems; user-based recommender and item-based recommender.

3. **Use-based filtering**: User-based preferences are very common in the field of designing personalized systems. This approach is based on the user's likings. The process starts with users giving ratings (1-5) to some movies. These ratings can be implicit or explicit. Explicit ratings are when the user explicitly rates the item on some scale or indicates a thumbs-up/thumbs-down to the item. Often explicit ratings are hard to gather as not every user is much interested in providing feedbacks. In these scenarios, we gather implicit ratings based on their behaviour. For instance, if a user buys a product more than once, it indicates a positive preference. In context to movie systems, we can imply that if a user watches the entire movie, he/she has some likeability to it. Note that there are no clear rules in determining implicit ratings.
4. **Item-based filtering**: Unlike the user-based filtering method, itembased focuses on the similarity between the item’s users like instead of the users themselves. The most similar items are computed ahead of time. Then for recommendation, the items that are most similar to the target item are recommended to the user.



2.2 Movie Recommendation System Using Content-based Filtering

Content-based recommendation is an important approach in recommender systems. The basic idea is to recommend items that are similar with what user liked before. The core mission of content-based recommender system is to calculate the similarity between items. There are a lot of methods to model item and the most famous one is Vector Space Model. The model extracts keywords of the item and calculate the weight by TF-IDF. For example, set ki as the ith keyword of item dj , wij is the weight of ki for dj , then the content of dj can be defined as:

$$Content(dj) = \{w1j, w2j, ...\}$$

As we talked before, content-based recommender system recommends items that are similar with what user liked before. So the tastes of a user can be modeled according to the history of what the user liked. Consider ContentBasedProfile(u) as the preference vector of user u, the definition is:

$$ContentBasedProfile(u) = 1\ |N(u)|\ sumd \in N(u) Content(d)$$

[9]

N(u) is what the user u liked before. After calculating content vector Content(.) and content preference vector ContentBasedProfile(.) of all users, given any user u and an item d, how the user like the item is defined as the similarity between ContentBasedProfile(u) and Content(d): p(u, d) = sim(ContentBasedProfile(u), Content(d)) (2.3) Using keywords to model item is an important step for manu and an item d, how the user like the item is defined as the similarity between ContentBasedProfile(u) and Content(d):

p(u, d) = sim(ContentBasedProfile(u), Content(d))

Using keywords to model item is an important step for many recommender systems. But extracting keywords of an item is also a difficult problem, especially in media field, because it is very hard to extract text keywords from a video. For solving this kind of problem, there are two main ways. One is letting experts tag the items and another one is letting users tag them. The representative of expert tagged systems are Pandora for music and Jinni for movies. Let's take Jinni as an example, the researchers of Jinni defined more than 900 tags as movie gene, and they let movie experts to make tags for them. These tags belong to different categories, including movie genre, plot, time, location and cast. As we can see from the figure, the tags of Kung Fu Panda are divided into ten categories totally, Mood, Plot, Genres, Time, Place, Audience, Praise, Style, Attitudes and Look. These tags contain all aspects of movie information, which can describe a movie very accurately.

2.3 Movie Recommendation System Using Popularity-based Filtering

A movie recommendation system using popularity, also known as a popularity recommender, suggests popular movies to users based on votes and ratings given to each movie. This approach recommends movies to users according to the movies' overall popularity, as indicated by the number of votes and the average rating. Popularity-based recommendation systems are a good starting point for

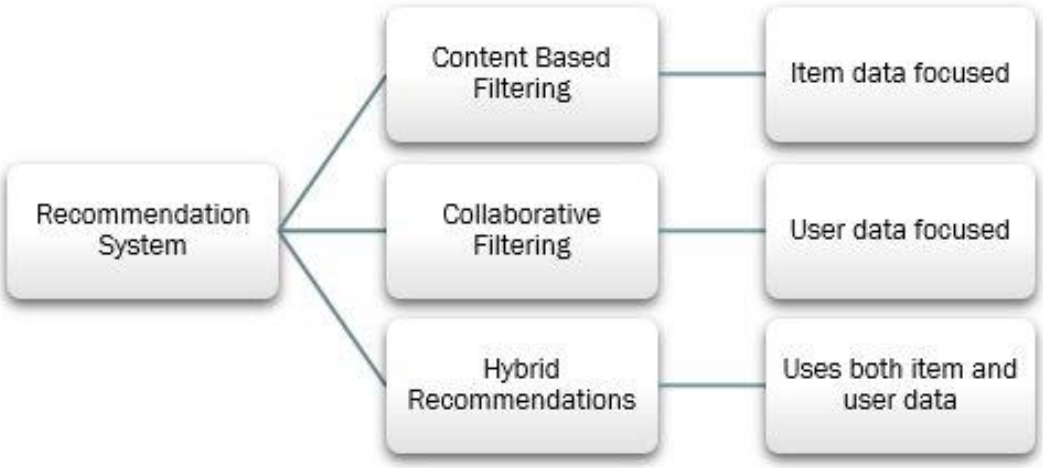
recommendation systems, but they may not provide personalized recommendations and can be biased towards well-known or mainstream movies.

2.4 Movie Recommendation System Using Hybrid-based Filtering

Hybrid Recommender System is more and more popular currently. Combining collaborative filtering and content-based filtering can be more effective by recently research. There are many ways to implement hybrid recommender systems: simply combine the result of CF and CB recommendations, add CF capability to a CB method.

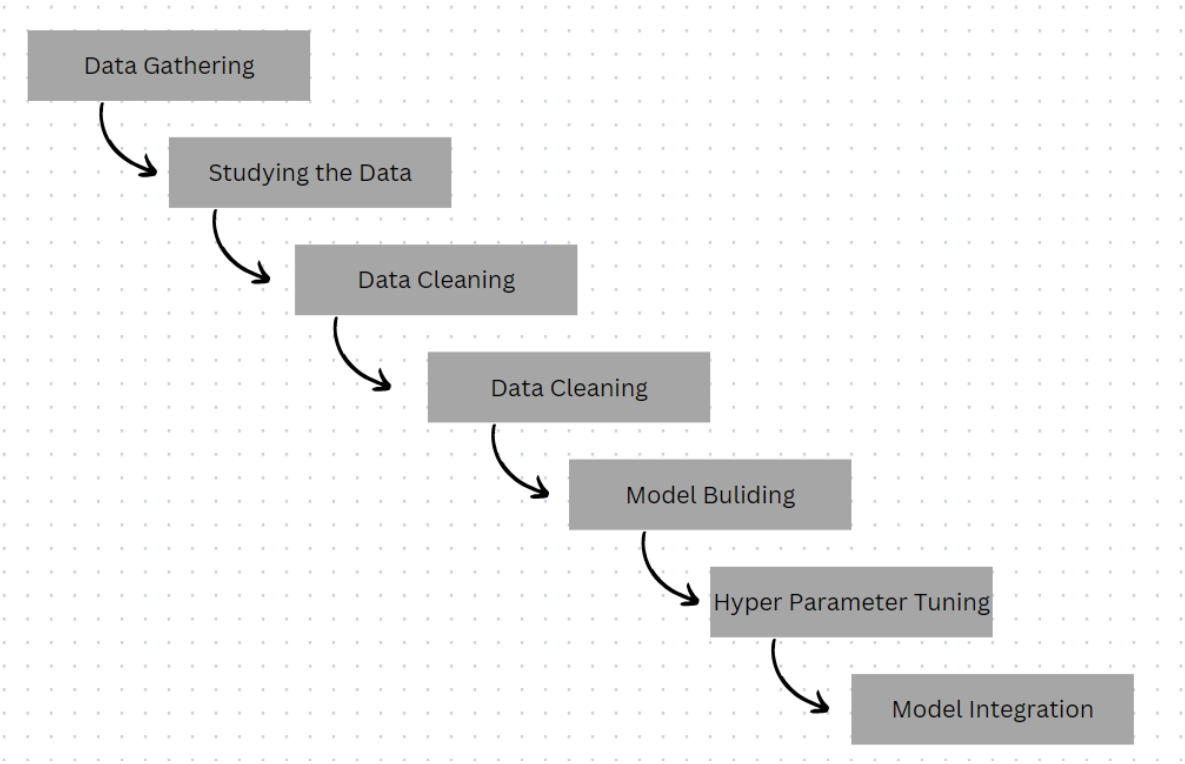
There are seven hybridization methods:

- Weighted: Add scores from different recommender components.
- Switching: Choose methods by switching in different recommender components.
- Mixed: Show recommendation result from different systems.
- Features Combination: Extract features from different sources and combine them as a single input.
- Feature Augmentation: Calculate features by one recommender and put the result to the next step.
- Cascade: Generate a rough result by a recommender technique and recommend on the top of the previous result.
- Meta-level: Use the model generated by one recommender as the input of another recommender technique. Although there are many combinations theoretically, it is not always efficacious for a specific problem. The most important principle of hybrid recommendation is to avoid or make up the weakness of every single recommender technique.



3. OVERALL DESCRIPTION

3.1 Workflow of Project



3.2 Data Description

The Data Set contains:

The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website..

- movies_metadata.csv: The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.

- keywords.csv: Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.
- credits.csv: Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.
- links.csv: The file that contains the TMDB and IMDB IDs of all the movies featured in the Full MovieLens dataset.
- links_small.csv: Contains the TMDB and IMDB IDs of a small subset of 9,000 movies of the Full Dataset.
- ratings_small.csv: The subset of 100,000 ratings from 700 users on 9,000 movies.

3.3 Importing Dataset.

Here we can see that we have categorical and continuous variables, we can also see that we have a lot of null values.

Here the explanation about the variables:

- **cast**: Information about casting. Name of actor, gender and it's character name in movie
- **crew**: Information about crew members. Like who directed the movie, editor of the movie and so on.
- **id**: It's movie ID given by TMDb
- **Keywords**: Tags/keywords for the movie. It list of tags/keywords
- **movieId:** It's serial number for movie
- **imdbId:** Movie id given on IMDb platform
- **tmdbId**: Movie id given on TMDb platform
- **adult:** Indicates if the movie is X-Rated or Adult.
- **belongs_to_collection:** A stringified dictionary that gives information on the movie series the particular film belongs to.

- **budget:** The budget of the movie in dollars.
- **genres:** A stringified list of dictionaries that list out all the genres associated with the movie.
- **homepage:** The Official Homepage of the move.
- **id:** The ID of the movie.
- **imdb_id:** The IMDB ID of the movie.
- **original_language:** The language in which the movie was originally shot in.
- **original_title:** The original title of the movie.
- **overview:** A brief blurb of the movie.
- **popularity:** The Popularity Score assigned by TMDB.
- **poster_path:** The URL of the poster image.
- **production_companies:** A stringified list of production companies involved with the making of the movie.
- **production_countries:** A stringified list of countries where the movie was shot/produced in.
- **release_date:** Theatrical Release Date of the movie.
- **revenue:** The total revenue of the movie in dollars.
- **runtime:** The runtime of the movie in minutes.
- **spoken_languages:** A stringified list of spoken languages in the film.
- **status:** The status of the movie (Released, To Be Released, Announced, etc.)
- **tagline:** The tagline of the movie.
- **title:** The Official Title of the movie.
- **video:** Indicates if there is a video present of the movie with TMDB.
- **vote_average:** The average rating of the movie.
- **vote_count:** The number of votes by users, as counted by TMDB.

- **userId:** It is id for User
- **movieId:** It is TMDb movie id.
- **rating:** Rating given for the particular movie by specific user
- **timestamp:** Time stamp when rating has been given by user

3.4 Project Initiation:

- Define the project scope, objectives, and deliverables.
- Set up project management tools and resources.

3.5 Data Acquisition:

- Obtain the Movie dataset from the appropriate source, ensuring compliance with data privacy and security regulations.
- Understand the structure and format of the dataset, including the features, target variable, and any data dictionaries or documentation provided.

3.6 Data Exploration and Understanding:

- Perform initial data exploration to gain insights into the dataset's characteristics, including the distribution of variables, missing values, outliers, and potential data quality issues.
- Visualize key features using descriptive statistics, histograms, box plots, and correlation matrices.

3.7 Data Pre-processing:

- Handle missing data by imputation, removal, or interpolation techniques.
- Remove HTML tags:

  The attributes present in our dataset like title and body have HTML tags like Encode categorical variables using techniques such as one-hot encoding or label encoding.

- Removing StopWords :

  The stop words present in our dataset don’t have any significance when we

train our model. We don't lose any information while training the data. But removing stopwords will make the size of our dataset small and this reduces training

3.8 Model Development:

- Split the dataset into training, validation, and test sets to evaluate model performance effectively.
- Select appropriate machine learning algorithms for the predictive task, such as logistic regression, decision trees, random forests, gradient boosting, or neural networks.
- Train initial models using default hyperparameters and evaluate their performance using appropriate evaluation metrics.
- Perform hyperparameter tuning using techniques such as grid search, random search, or Bayesian optimization to optimize model performance.
- Experiment with ensemble methods or model stacking to further enhance predictive accuracy.

3.9 Model Evaluation:

- Assess the performance of trained models using a comprehensive set of evaluation metrics, including accuracy, precision, recall, F1-score, ROC AUC, and calibration plots.
- Conduct sensitivity analysis to understand the robustness of the models across different scenarios and thresholds.
- Compare the performance of different models and select the best-performing one based on predefined criteria.

3.10 Model Interpretation:

- Interpret model predictions and feature importance to understand the factors driving vehicle loan repayment behaviour.

- Communicate findings to stakeholders in a clear and interpretable manner, highlighting actionable insights and recommendations.

3.11 Model Deployment:

- Deploy the selected model into production environments, integrating it into the movie recommendation process to assist in decision-making.
- Implement monitoring mechanisms to track model performance and recalibrate the model as needed over time.

3.12 Documentation and Reporting:

- Document the entire project workflow, including data pre-processing steps, model development process, evaluation results, and deployment procedures.
- Prepare a comprehensive report summarizing the project findings, insights.

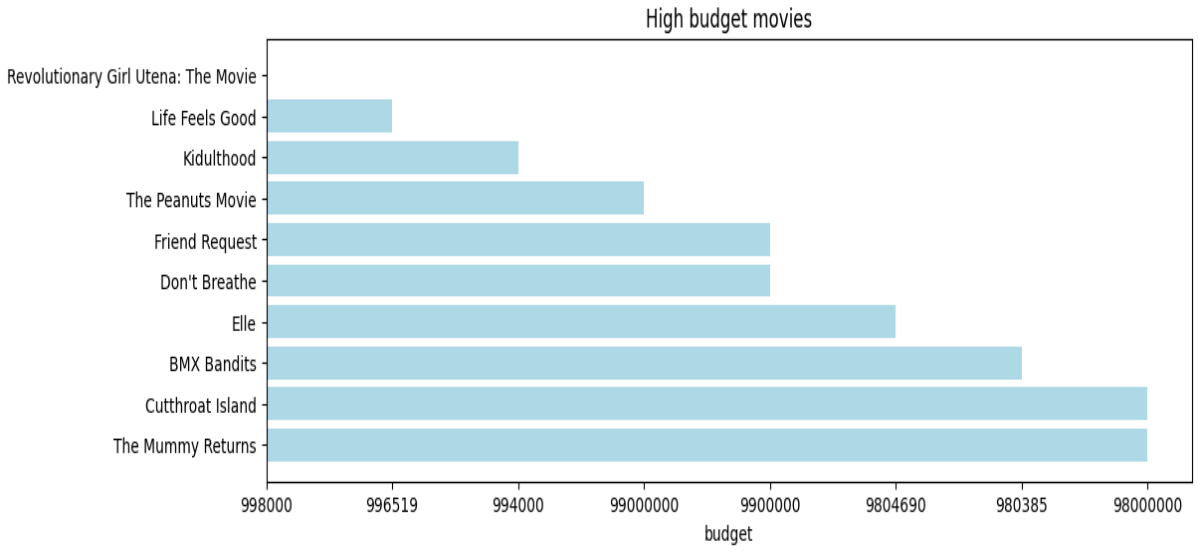


Fig 1: high budget movie barchart

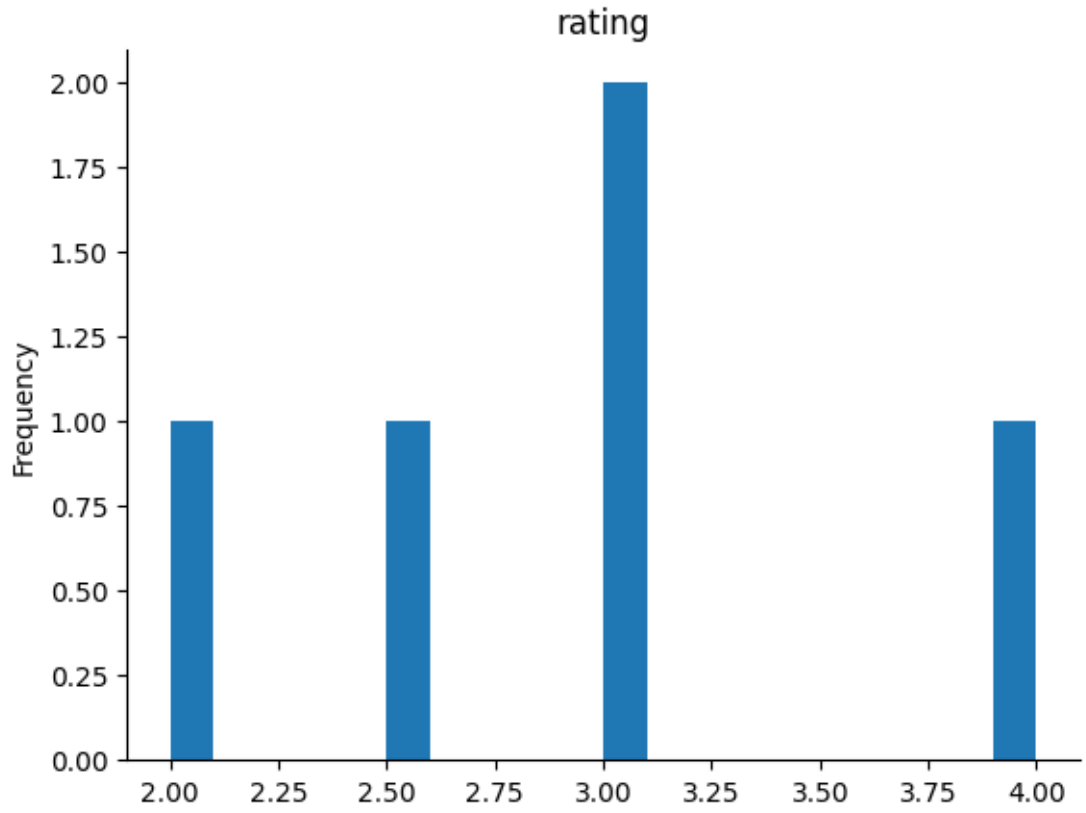


Fig2: Rating bar chart

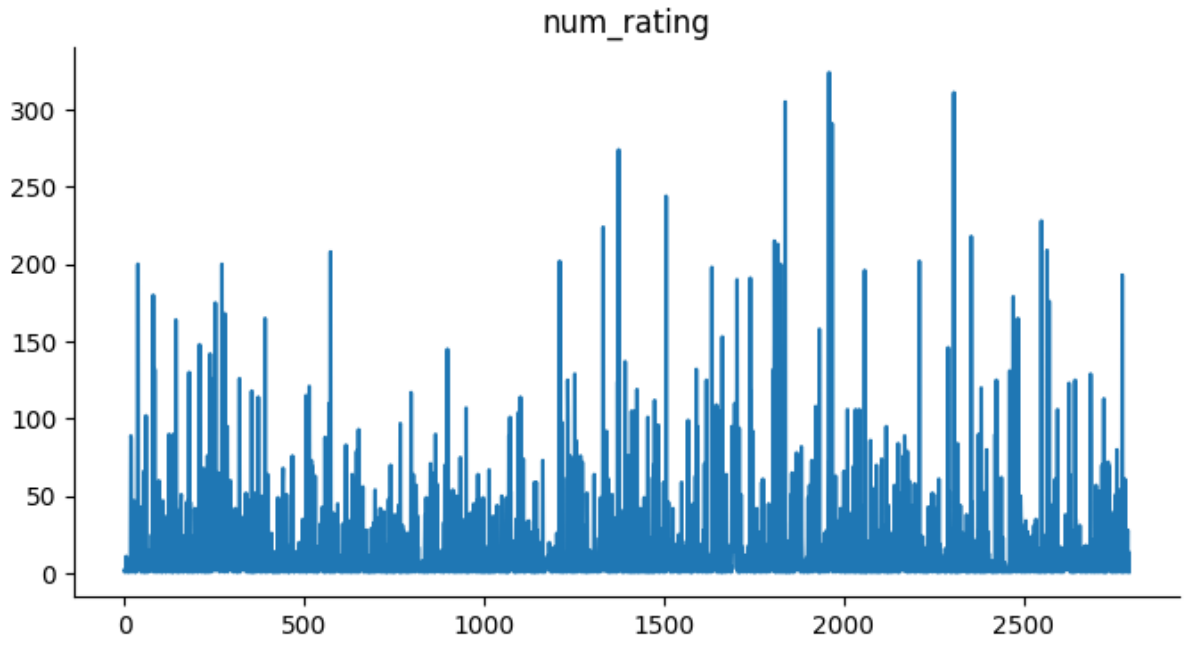


Fig 3: num_rating bar graph

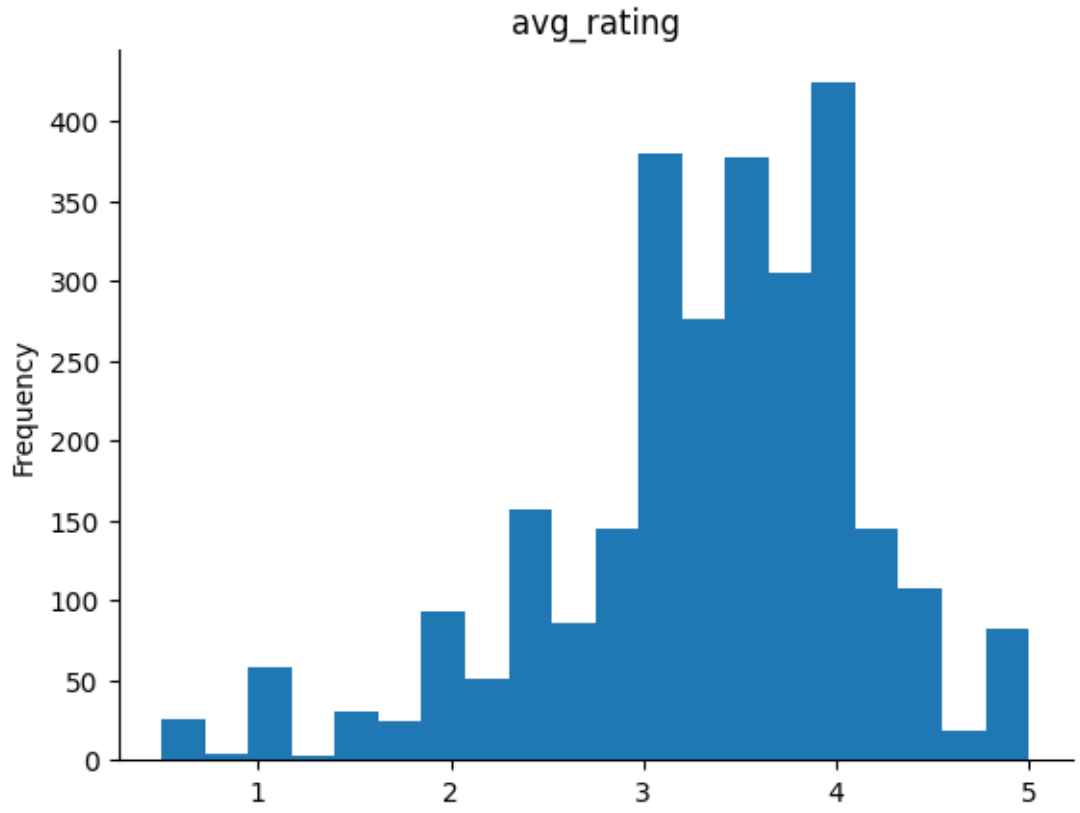


Fig 4: Avg_rating Histogram

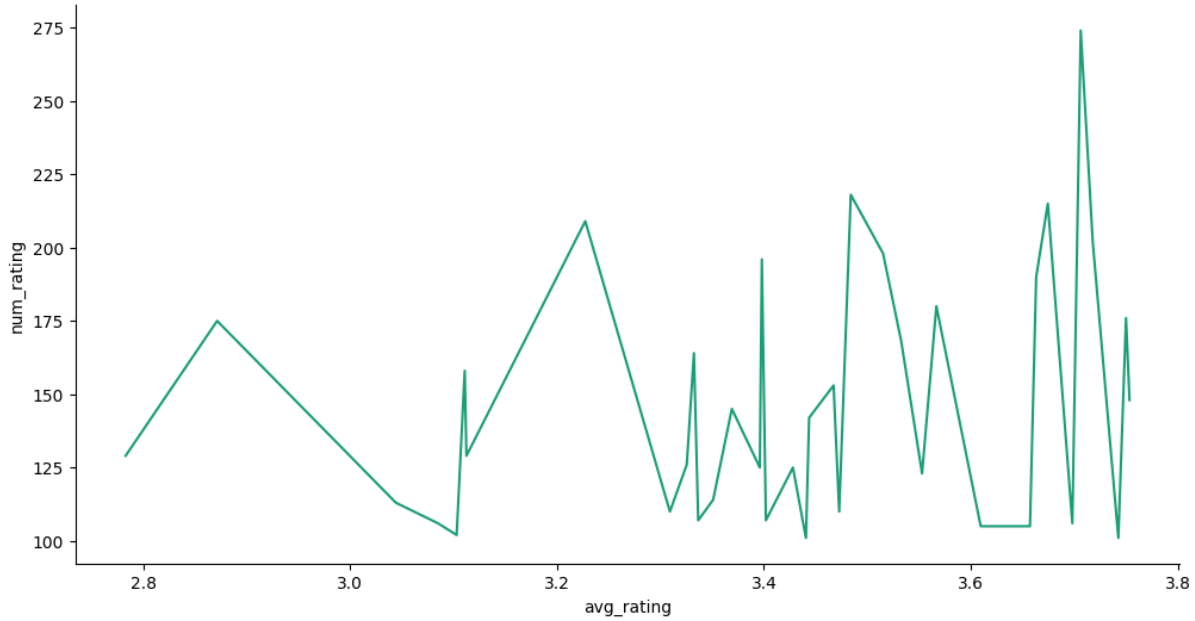


Fig 5: avg_rating and num_rating line chart

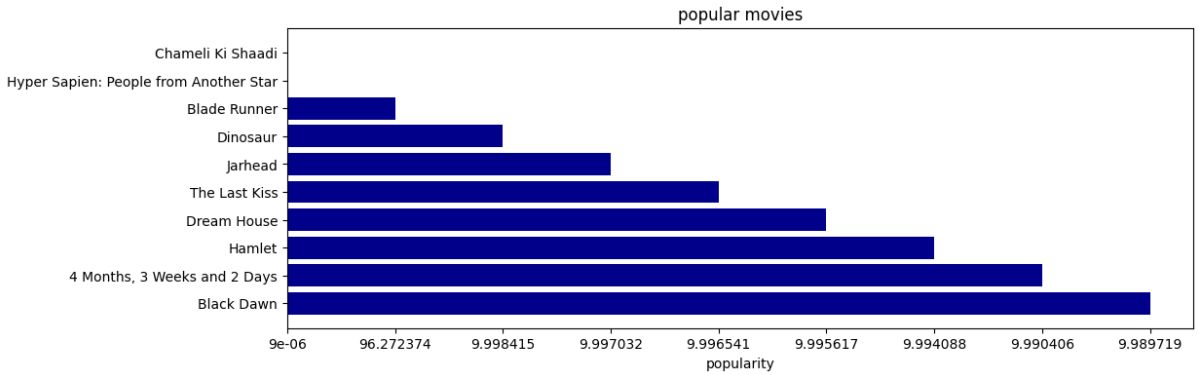


Fig6 : Bar chart of popular movies

4. REQUIREMENTS SPECIFICATION

4.1 Hardware Requirement:

- 500 GB hard drive (Minimum requirement)
- 8 GB RAM (Minimum requirement)
- PC x64-bit CPU

4.2 Software Requirement:

- Windows/Mac/Linux
- Collab

Libraries:

- Numpy 1.18.2 : Used for working with arrays
- Pandas 1.2.1 : Used for data analysis.
- Matplotlib 3.3.3 : Used for visual representation like plotting graphs.

- Scikit-learn 0.24.1 : Used for making use of Machine learning tools
- AST: This module helps python application to process trees of the python abstract syntax grammar.

5. IMPLEMENTATION

5.1 Cosine Similarity:

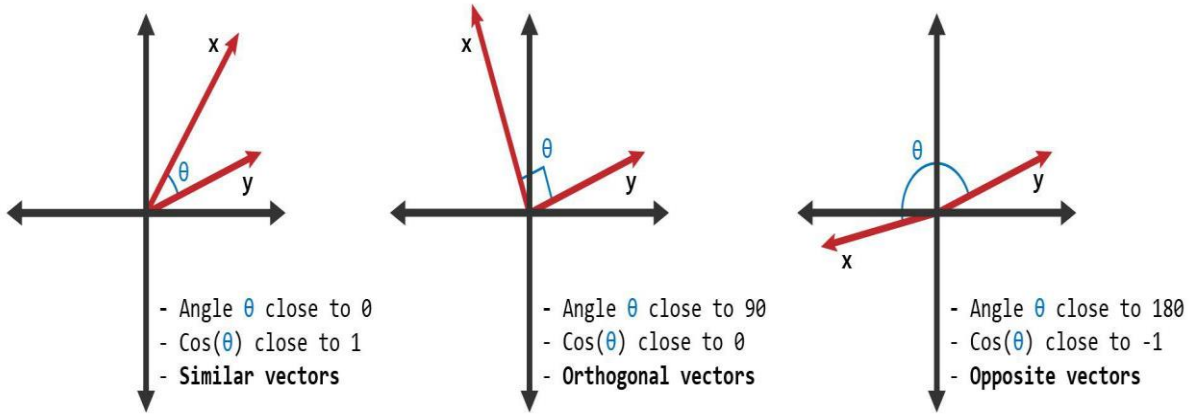
Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

Cosine similarity is a metric used to measure the similarity of two vectors. Specifically, it measures the similarity in the direction or orientation of the vectors ignoring differences in their magnitude or scale. Both vectors need to be part of the same inner product space, meaning they must produce a scalar through inner product multiplication. The similarity of two vectors is measured by the cosine of the angle between them.

$$similarity(A, B) = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

where

- θ is the angle between the vectors,
- $A \cdot B$ is dot product between A and B and calculated as $A \cdot B = A^T B = \sum_{i=1}^{n} A_i B_i = A_1 B_1 + A_2 B_2 + \ldots + A_n B_n$,
- $\|A\|$ represents the L2 norm or magnitude of the vector which is calculated as $\|A\| = \sqrt{A_1^2 + A_1^2 \ldots A_1^n}$.



5.2 Singular Value Decomposition (SVD):

Let A be an n*d matrix with singular vectors v1, v2, . . . , vr and corresponding singular values σ1, σ2, . . . , σr. Then ui = (1/σi)Avi , for i = 1, 2, . . . , r, are the left singular vectors and by Theorem 1.5, A can be decomposed into a sum of rank one matrices a

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u_i} \mathbf{v_i}^T.$$

We first prove a simple lemma stating that two matrices A and B are identical if Av = Bv for all v. The lemma states that in the abstract, a matrix A can be viewed as a transformation that maps vector v onto Av

Importing libraries

```
[ ]  import pandas as pd
     import numpy as np
     %matplotlib inline
     import matplotlib.pyplot as plt
     import seaborn as sns
     import ast
     from scipy import stats
     from ast import literal_eval
     from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
     from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
```

Loading Dataset

```
[ ]  credits = pd.read_csv("credits.csv")
```

```
[ ]  movies = pd.read_csv("movies_metadata.csv",low_memory = False)
```

```
[ ]  ratings = pd.read_csv("ratings.csv")
```

```
[ ]  ratings_small = pd.read_csv("ratings_small.csv")
```

```
[ ]  links = pd.read_csv("links.csv")
```

```
[ ]  links_small = pd.read_csv("links_small.csv")
```

```
[ ]  keywords = pd.read_csv("keywords.csv")
```

Cleaning Missing Values

```
mdata_ohe.isnull().sum()
```

```
id           0
title        4
overview    995
genres        0
keywords      0
cast          0
crew          0
dtype: int64
```

```
mdata_ohe.isnull().sum()
```

```
id          0
title       0
overview    0
genres      0
keywords    0
cast        0
crew        0
dtype: int64
```

Importing Abstract Syntax Tree

```
[ ] import ast
```

```
[ ] def convert(obj):
        l=[]
        counter=0
        for i in ast.literal_eval(obj):
            if counter!=3:
                l.append(i['name'])
                counter+=1
            else:
                break
        return l
```

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

```
mdata_ohe['genres'].head(10)
```

```
0      [Animation, Comedy, Family]
1     [Adventure, Fantasy, Family]
2                [Romance, Comedy]
3          [Comedy, Drama, Romance]
4                          [Comedy]
5            [Action, Crime, Drama]
6                 [Comedy, Romance]
7        [Action, Adventure, Drama]
8    [Action, Adventure, Thriller]
9    [Adventure, Action, Thriller]
Name: genres, dtype: object
```

```
[ ] mdata_ohe['keywords']=mdata_ohe['keywords'].apply(convert)
```

Top 50 Popular Books

```
[ ] pop_data
```

	title	num_rating	avg_rating	id	genres	overview	cast	crew
0	Lost in Translation	129	2.782946	153	[Drama]	[Two, lost, souls, visiting, Tokyo, --, the, y...	[BillMurray, ScarlettJohansson, AnnaFaris]	[]
1	Bang, Boom, Bang	175	2.871429	344	[Crime, Action, Comedy]	[Bank, robber, Kelle, Grabowski, escapes, from...	[OliverKorittke, MarkusKnüfken, RalfRichter]	[PeterThorwarth]
2	Who Killed Bambi?	113	3.044248	1917	[Thriller]	[Isabelle,, a, beautiful, nursing, student,, i...	[SophieQuinton, LaurentLucas, CatherineJacob]	[]
3	Tough Enough	106	3.084906	434	[Drama, Thriller]	[From, the, youth, directed, novel, of, the, s...	[DavidKross, JennyElvers, ErhanEmre]	[]
4	Tough Enough	106	3.084906	38556	[Action, Drama, Romance]	[An, aspiring, country/western, singer,, whose...	[DennisQuaid, CarleneWatkins, StanShaw]	[RichardFleischer]
5	A Clockwork Orange	102	3.102941	185	[ScienceFiction, Drama]	[Demonic, gang-leader, Alex, goes, on, the, sp...	[MalcolmMcDowell, PatrickMagee, AdrienneCorri]	[StanleyKubrick]
6	Syriana	158	3.110759	231	[Drama, Thriller]	[The, Middle, Eastern, oil, industry, is, the,...	[GeorgeClooney, MattDamon, JeffreyWright]	[]
7	Wag the Dog	129	3.112403	586	[Comedy, Drama]	[During, the, final, weeks, of, a, presidentia...	[DustinHoffman, RobertDeNiro, AnneHeche]	[BarryLevinson]
8	Titanic	209	3.227273	597	[Drama, Romance, Thriller]	[84, years, later,, a, 101-year-old, woman, na...	[KateWinslet, LeonardoDiCaprio, FrancesFisher]	[]
9	Titanic	209	3.227273	16535	[Drama, Action, Romance]	[Unhappily, married,, Julia, Sturges, decides,...	[CliftonWebb, BarbaraStanwyck, RobertWagner]	[]
10	Titanic	209	3.227273	2699	[Action, Drama, Romance]	[A, story, of, the, romances, of, two, couples...	[PeterGallagher, GeorgeC.Scott, CatherineZeta-...	[]
11	Dave Chappelle's Block Party	110	3.309091	292	[Comedy, Documentary, Music]	[The, American, comedian/actor, delivers, a, s...	[DaveChappelle, ErykahBadu, Common]	[MichelGondry]
12	Big Fish	126	3.325397	587	[Adventure, Fantasy, Drama]	[Throughout, his, life, Edward, Bloom, has, al...	[EwanMcGregor, AlbertFinney, BillyCrudup]	[]
13	All the Way Boys	164	3.332317	1721	[Adventure, Action, Comedy]	[The, "Trinity", crew, makes, another, modern,...	[AlexanderAllerson, BudSpencer, TerenceHill]	[]

Importing CountVectorizer

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

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

```
[ ] vector
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

Cosine Similartity

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

```
[ ]  similarity = cosine_similarity(vector)
```

```
[ ]  similarity
```

```
array([[1.        , 0.04445542, 0.05057217, ..., 0.        , 0.03143473,
        0.        ],
       [0.04445542, 1.        , 0.07756315, ..., 0.        , 0.03214122,
        0.        ],
       [0.05057217, 0.07756315, 1.        , ..., 0.        , 0.03656362,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.38138504,
        0.        ],
       [0.03143473, 0.03214122, 0.03656362, ..., 0.38138504, 1.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        1.        ]])
```

Content based recommender

```
def recommend(movie):
    index = new_data[new_data['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])
    for i in distances[1:11]:
        print(new_data.iloc[i[0]].title)
```

```
[ ]  recommend('Waiting to Exhale')
```

```
Worth Winning
Snow days
Jasminum
If You Love
Chilly Scenes of Winter
A Good Marriage
Next Stop Wonderland
Little Black Book
About Last Night...
French Cancan
```

Collaborative filtering Based

```
[ ]  def recommendor(movie):
        # index fetch
        index = np.where(pt.index==movie)[0][0]
        similar_items = sorted(list(enumerate(similarity_score[index])),key=lambda x:x[1],reverse=True)[1:11]

        data = []
        for i in similar_items:
            item = []
            temp_df = mdata_ohe[mdata_ohe['title'] == pt.index[i[0]]]
            item.extend(list(temp_df.drop_duplicates('title')['title'].values))
            item.extend(list(temp_df.drop_duplicates('title')['cast'].values))
            item.extend(list(temp_df.drop_duplicates('title')['genres'].values))


            data.append(item)

        return data
```

Recommends 10 similar movies

```
[ ]  recommendor('A Brief History of Time')
```

```
[['Street Kings',
  ['KeanuReeves', 'ForestWhitaker', 'ChrisEvans'],
  ['Action', 'Crime', 'Drama']],
 ['The Breakfast Club',
  ['EmilioEstevez', 'AnthonyMichaelHall', 'JuddNelson'],
  ['Comedy', 'Drama']],
 ['Cold Mountain', ['JudeLaw', 'NicoleKidman', 'RenéeZellweger'], ['Drama']],
 ['Carry On Screaming',
  ['HarryH.Corbett', 'KennethWilliams', 'JimDale'],
  ['Comedy']],
 ['Notes on a Scandal',
  ['JudiDench', 'CateBlanchett', 'BillNighy'],
  ['Drama', 'Romance']],
 ['Lonely Hearts',
  ['JohnTravolta', 'JamesGandolfini', 'SalmaHayek'],
  ['Drama', 'Thriller', 'Crime']],
 ['Nostalgia',
  ['OlegYankovskiy', 'ErlandJosephson', 'DomizianaGiordano'],
  ['Drama', 'Romance']],
 ['Say Anything...',
  ['JohnCusack', 'IoneSkye', 'JohnMahoney'],
  ['Comedy', 'Drama', 'Romance']],
 ['The Chronicles of Riddick: Dark Fury',
  ['VinDiesel', 'RhianaGriffith', 'KeithDavid'],
  ['Action', 'Animation', 'ScienceFiction']],
 ['Once Were Warriors',
  ['RenaOwen', 'TemueraMorrison', 'MamaengaroaKerr-Bell'],
  ['Drama']]]
```

6. CONCLUSION

In this project, to improve the accuracy, quality and scalability of movie recommendation system, a Hybrid approach by unifying content based filtering and collaborative filtering; using Singular Value Decomposition (SVD) as a classifier and Cosine Similarity is presented in the proposed methodology. Existing pure approaches and proposed hybrid approach is implemented on three different Movie datasets and the results are compared among them. 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.

7. FUTURE SCOPE

In the proposed approach, It has considered Genres of movies but, in future we can also consider age of user as according to the age movie preferences also changes, like for example, during our childhood we like animated movies more as compared to other movies. There is a need to work on the memory requirements of the proposed approach in the future. The proposed approach has been implemented here on different movie datasets only. It can also be implemented on the Film Affinity and Netflix datasets and the performance can be computed in the future.

REFERENCES

- https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
- https://gemini.google.com
- https://www.w3schools.com
- https://github.com/jalajthanaki/Movie_recommendation_engine
- https://scikit-learn.org/stable/