

**Department of Computer Applications**

Mini Project Report

DETECTION OF DEPRESSION FROM TEXTUAL DATA IN SOCIAL MEDIA USING MACHINE LEARNING

#### Done by

**Malavika P**

**Reg No:MAC23MCA-2037**

Under the guidance of

**Prof. Biju Skaria**

**2023-2025**

CERTIFICATE



Detection of Depression from Textual Data in Social Media Using Machine Learning

#### Certified that this is the bonafide record of project work done by

**Malavika P**

**Reg No: MAC23MCA-2037**

#### during the third semester, in partial fulfilment of requirements for award of the degree

##### Master of Computer Applications

of

##### APJ Abdul Kalam Technological University Thiruvananthapuram

**Faculty Guide Head of the Department**

#### Prof. Biju Skaria Prof. Biju Skaria

**Project Coordinator Internal Examiners**

#### Prof. Sonia Abraham

**ACKNOWLEDGEMENT**

With heartfelt gratitude, I extend my deepest thanks to the Almighty for His unwavering grace and blessings that have made this journey possible. May His guidance continue to illuminate my path in the years ahead.

I am immensely thankful to Prof. Biju Skaria, Head of the Department of Computer Applications and my mini project guide, and Prof. Sonia Abraham, our dedicated project coordinator, for their invaluable guidance and timely advice, which played a pivotal role in shaping this project. Their guidance, constant supervision, and provision of essential information were instrumental in the successful completion of the mini project.

I extend my profound thanks to all the professors in the department and the entire staff at MACE for their unwavering support and inspiration throughout my academic journey. My sincere appreciation goes to my beloved parents, whose guidance has been a beacon in every step of my path.

I am also grateful to my friends and individuals who generously shared their expertise and assistance, contributing significantly to the fulfillment of this endeavor.

**ABSTRACT**

The "Depression Detection from Social Media Using Machine Learning" project addresses the urgent global need for identifying undiagnosed cases of depression by analyzing textual data from social media platforms. Depression, a leading cause of suicide worldwide, often goes undetected and untreated. Leveraging social media data provides a promising pathway to identifying at-risk individuals earlier, allowing for timely intervention. This study utilizes advanced feature extraction techniques and machine learning classifiers to detect depressive content with high accuracy, ultimately aiming to create a user-friendly and effective tool for both individuals and mental health professionals.

Previous research has demonstrated the potential of various feature extraction and classification techniques in detecting depression. In particular, one study achieved 91% accuracy by integrating Linguistic Inquiry and Word Count (LIWC), Latent Dirichlet Allocation (LDA), and bigrams with a Multilayer Perceptron (MLP) model on Reddit posts, proving effective for identifying depressive language patterns. Another study demonstrated that Bag-of-Words and N-grams, when combined with Logistic Regression (LR) and Random Forest (RF), could identify depressive social media posts with over 90% accuracy, without depending on explicit keywords such as "depression." A third study underscored the effectiveness of Decision Trees in detecting depression in Facebook comments, achieving 60-80% accuracy and tracking depressive trends over time.

In this project, a combination of N-grams and Term Frequency-Inverse Document Frequency (TF-IDF) is employed for feature extraction, which highlights various facets of text data, including emotional processes, linguistic style, and temporal patterns. The classification model incorporates ensemble learning, utilizing MLP and RF for robust predictions, while Decision Trees add complementary insights, optimizing the system's detection capabilities. This approach not only identifies depressive content but also offers recommendations for support, thus addressing both detection and intervention in a holistic manner.

The system is accessible through a user-friendly web interface, built with the Flask framework, enabling real-time analysis of social media posts or comments. Users can input textual content to receive instant feedback on whether the content may indicate depressive tendencies. For those identified as potentially at risk, the system provides support suggestions, thus extending beyond detection to deliver actionable insights for users seeking help or for mental health professionals tracking their patients’ progress.

This model was trained and evaluated on a Kaggle dataset curated by B.B.eye, comprising 10,314 tweets focused on depression analysis. This dataset, with its depth and specificity, allows the system to learn nuanced patterns associated with depressive language and behaviours. By analysing text on social media, this system can facilitate timely support for those at risk, contributing to broader mental health awareness and intervention efforts on a global scale.

**LIST OF TABLES**

2.1 Reference papers summary 5

### LIST OF FIGURES

* 1. Snapshot of ratings.csv 7
  2. Snapshot of movies.csv 8
  3. Snapshot of users.csv 8
  4. Number of Movies, Users and Ratings 9
  5. Data type of Each Attribute 9
  6. Dropped attribute timestamp 10
  7. Merged movie title 10
  8. Handling missing values and checking duplicates 11
  9. Number of ratings per user 11
  10. Number of ratings per movie 12
  11. Occupation distribution of users 15
  12. Ratings distribution 16
  13. Pie chart for movie genre count 16
  14. Boxplot for ratings 17
  15. Example figure of kNN 19
  16. Activity Diagram 21
  17. Project Pipeline 22
  18. User feature dataframe 31
  19. Model knn1 31
  20. Model knn2 32
  21. Code snippet for user based recommendation 32
  22. Recommendation for user 23… 33
  23. Code snippet for item based recommendation 33
  24. Recommendation for the movie ‘Speed’ 34
  25. User movie input 35
  26. Item based recommendation 35
  27. User ID input 35
  28. User based recommendation 36
  29. Index page of movie recommender 38
  30. Input page of movie recommender 38
  31. User id input 39
  32. Similar users in recommendation for user 23 39
  33. Recommendation for user 12 40
  34. Movie title input 40
  35. Recommendation for the movie’12 Angry Men’ 41
  36. Validation for user id 41
  37. Validation for movie title 42
  38. Git history of 3 sprints 43
  39. Git history of sprint 3 43

**CONTENTS**

##### Introduction 1

##### Supporting Literature 2

* 1. Literature Review 2
     1. Summary Table 5
  2. Findings and Proposals 6

##### System Analysis 7

* 1. Analysis of Dataset 7
     1. About the Dataset 7
     2. Explore the dataset 9
  2. Data Pre-processing 10
     1. Data Cleaning 10
     2. Analysis of Feature Variables 13
     3. Analysis of Class Variables 15
  3. Data Visualization 15
  4. Analysis of Algorithm 18
     1. Activity Diagram 21
  5. Project Plan 22
     1. Project Pipeline 22
     2. Project Implementation Plan 23
  6. Feasibility Analysis 24
     1. Technical Feasibility 24
     2. Economic Feasibility 25
     3. Operational Feasibility 26
  7. System Environment 27
     1. Software Environment 27
     2. Hardware Environment 29

##### System Design 30

* 1. Model Building 30
     1. [Model Planning 30](#_TOC_250011)
     2. [Training 34](#_TOC_250010)
     3. [Testing 34](#_TOC_250009)

1. [Results and Discussion 35](#_TOC_250008)
2. [Model Deployment 37](#_TOC_250007)
3. [Git History 43](#_TOC_250006)
4. [Conclusion 44](#_TOC_250005)
5. [Future Work 45](#_TOC_250004)
6. [Appendix 47](#_TOC_250003)
   1. [Minimum Software Requirements 47](#_TOC_250002)
   2. [Minimum Hardware Requirements 49](#_TOC_250001)
7. [References 50](#_TOC_250000)
8. **INTRODUCTION**

Depression is a leading cause of suicide worldwide, with severe consequences for individuals and society. Many cases remain undiagnosed and untreated, often due to social stigma, lack of awareness, and limited access to mental health services. According to the World Health Organization (WHO), depression affects over 300 million people globally, accounting for more than two-thirds of suicides each year. These barriers to treatment highlight the need for alternative approaches to detection and intervention.

Social media platforms have emerged as valuable resources for studying mental health patterns, with platforms like Twitter, Facebook, and Reddit allowing people to freely express their thoughts and feelings. Studies show that individuals with major depressive disorder often share subtle indicators of their condition through language and sentiment in their posts, which can be analyzed to predict the likelihood of depression and offer a potential pathway for early intervention. By examining emotional expression, linguistic style, and temporal indicators, researchers can identify patterns unique to depressive behaviour, providing insights into how individuals manage their symptoms over time.

A recent survey indicates that many people, especially teenagers and young adults, turn to social media to express symptoms of depression and anxiety. However, existing research often relies on specific keywords such as “depression” and “diagnose” when analysing social media data, limiting effectiveness as users may avoid these terms due to stigma. Social media, then, becomes a space for informal expression rather than direct admission of mental health struggles.

This project seeks to address these challenges by developing a machine learning model capable of detecting depressive tendencies in social media posts without relying on explicit keywords. Through advanced text analysis, the project aims to identify nuanced patterns associated with depression and provide early detection, bridging the gap between traditional mental health services and the informal support offered on social media. By making mental health support more accessible and responsive in today’s digital age, this project hopes to contribute meaningfully to early diagnosis and intervention, potentially improving the lives of individuals affected by untreated depression.

1. **SUPPORTING LITERATURE**
   1. **Literature Review**

Paper 1: Tadesse MM, Lin H, Xu B, Yang L. "Detection of depression-related posts in reddit social media forum." Ieee Access. 2019 Apr 4.

The paper titled "Detection of Depression-Related Posts in Reddit Social Media Forum" by Michael M. Tadesse et al. presents a study on identifying depression-related content on Reddit using natural language processing (NLP) and machine learning techniques. The authors address the growing concern of depression as a major contributor to global disability and a significant cause of suicide. They aim to examine Reddit users' posts to detect factors that may reveal depressive attitudes.

The researchers use natural language processing (NLP) and machine learning techniques to train and test their method. They identify words commonly used by people with depression and analyse their connection to mental health. Methods like N-grams (word combinations), LIWC (emotional and psychological content analysis), and LDA (topic identification) help them understand how language on social media can indicate depression.

The study compares single and combined feature sets using various text classification methods like including Logistic Regression, Support Vector Machine, Random Forest, Adaptive Boosting, and Multilayer Perceptron. Results show that proper feature selection and combination lead to higher performance. The best depression detection performance (91% accuracy, 0.93 F1 score) is achieved by combining LIWC, LDA, and bigram features using a Multilayer Perceptron neural network. This paper helps improve detecting depression on social media, like Reddit, by combining language analysis and machine learning, which could lead to earlier support for those at risk.

Table 2.1 Paper 1 Summary

|  |  |
| --- | --- |
| **Title** | Detection of Depression-Related Posts in Reddit Social Media Forum |
| **Dataset** | The dataset was built by Inna Pirina et al. and consists of a list of depressed and non-depressed users. |
| **Algorithm** | LR, SVM, RF, AdaBoost, MLP |
| **Result/Precision** | 91% accuracy and 0.93 F1 score using LIWC+LDA+bigram features with MLP. |
| **Future Proposal** | Examine the link between users personality and depression-related behaviour on social media, Apply the method to other mental health disorders, Test the approach on non-English language data. |

Paper 2: Raghavendra, C. K., and K. C. Srikantaiah. "Similarity based collaborative filtering model for Movie Recommendation Systems." *2021 5th international conference on intelligent computing and control systems*

Raghavendra, C. K., and K. C. Srikantaiah's paper, "Similarity based collaborative filtering model for Movie Recommendation Systems," presented at the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), conducts an in-depth exploration of collaborative filtering techniques in the context of movie recommendation systems. The authors delve into various methodologies, emphasizing the significance of similarity-based approaches for achieving heightened accuracy in movie recommendations. The paper opens with a meticulous literature review, positioning collaborative filtering as a cornerstone in the field of recommender systems. Recognizing the diverse range of approaches, Raghavendra and Srikantaiah focus on similarity-based models, underscoring their pivotal role in refining recommendation accuracy. The review not only provides a comprehensive overview of existing algorithms but also sheds light on the evolving landscape of collaborative filtering. One of the paper's key contributions lies in the nuanced analysis of different similarity measures and their impact on recommendation accuracy. The authors delve into the intricacies of similarity-based models, scrutinizing the effectiveness of measures such as cosine similarity, Pearson correlation, and Jaccard index. The authors go beyond a mere exposition of existing models; they critically evaluate the challenges and limitations inherent in current collaborative filtering approaches. Raghavendra and Srikantaiah highlight the need for personalized recommendations that go beyond traditional user-item interactions, emphasizing the incorporation of user preferences, item features, and contextual information to achieve a more holistic understanding of user behavior. In conclusion, "Similarity based collaborative filtering model for Movie Recommendation Systems" provides a comprehensive and insightful exploration of collaborative filtering techniques in the realm of movie recommendations. The authors' emphasis on similarity- based models, detailed analysis of similarity measures, and identification of challenges contribute significantly to the understanding of this dynamic field. This paper not only informs current practices but also inspires future research to propel movie recommendation systems towards higher levels of accuracy and personalization.

Paper 3: Anwar, Taushif, and V. Uma. "Comparative study of recommender system approaches and movie recommendation using collaborative filtering." *International Journal of System Assurance Engineering and Management* 12 (2021): 426-436

Taushif Anwar's paper, "Comparative Study of Recommender System Approaches and Movie Recommendation Using Collaborative Filtering," published in the International Journal of System Assurance Engineering and Management in 2021, undertakes a thorough exploration of diverse methodologies within collaborative filtering for movie recommendations. The author strategically examines the nuances of user-based and item-based approaches, shedding light on their respective strengths and limitations. The literature review within the paper unfolds the landscape of collaborative filtering, a widely adopted technique harnessing user behavior data to curate personalized movie recommendations. Anwar meticulously dissects the foundations of collaborative filtering, distinguishing between user-based and item-based methods. User-based methodologies revolve around the identification of similar users, leveraging their preferences to generate recommendations, while item-based approaches center on discerning similarities among items or movies themselves. This dichotomy provides a nuanced understanding of the intricate strategies deployed in collaborative filtering. The review also brings to the forefront the challenges intrinsic to collaborative filtering, with a spotlight on the cold start problem and sparsity issues. Anwar recognizes the complexities that arise when dealing with sparse datasets and the inherent difficulties in initiating recommendations for new users or items. This acknowledgment adds depth to the literature review, presenting a holistic view of the hurdles practitioners face in implementing collaborative filtering models for movie recommendations. In navigating the evolution of recommendation algorithms, Anwar extends the discussion to encompass matrix factorization techniques and hybrid models. These advancements mark a paradigm shift in the quest for enhanced accuracy and effectiveness in movie recommendations. The paper emphasizes the amalgamation of multiple approaches to create hybrid models, showcasing the dynamism of contemporary recommendation systems. In conclusion, "Comparative Study of Recommender System Approaches and Movie Recommendation Using Collaborative Filtering" serves as a compass in navigating the intricate realm of collaborative filtering for movie recommendations. Anwar's meticulous literature review encapsulates the essence of user-based and item-based methods, addresses challenges, and spotlights advancements in recommendation algorithms.

* + 1. **Summary Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PAPER TITLE** | **ALGORITHM** | **DATASET** | **METHODOLOGY** | **PRECISION/ RMSE** |
| Movie Recommender System Using Collaborative Filtering, 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC) | K-Nearest Neighbors (KNN) | Movie Lens  (Kaggle) | Collaborative filtering techniques, such as user-based or item- based similarity calculations, to make movie recommendations based on user ratings and interactions. | PRECISION:  Content-based: 0.501  Collaborative filtering: 0.782 |
| Similarity Based Collaborative Filtering Model for Movie Recommendati on Systems, 2021 5th  International Conference on Intelligent Computing and Control Systems  (ICICCS) | K-Nearest Neighbors | Movie Lens  (Kaggle) | Computing user or item similarities based on movie ratings and using these similarities to make movie recommendations in a Similarity Based Collaborative Filtering Model | PRECISION:  Item Based Collaborative Filtering: 0.84  User Based Collaborative Filtering: 0.76 |
| Comparative study of recommender system Approaches and movie recommendatio n using collaborative filtering, International journal of System Assurance Engineering and Management | K- Nearest Neighbours  SVD SVD++  Co- Clustering | Movie Lens 100k  (Group Lens) | Comparative Study to analyze different recommender system approaches, focusing on collaborative filtering and advancements in recommendation algorithms. | RMSE:  KNN: 0.979  SVD: 0.9358  SVD++: 0.9201  Co-Clustering: 0.9675 |

Table 2.1 Reference papers summary

**2.2. Findings and Proposals**

After a comprehensive review of three noteworthy papers in the domain of movie recommendation systems, it is evident that each contributes significantly to the understanding and advancement of collaborative filtering techniques. In the quest for an optimal recommendation algorithm, the choice of k-Nearest Neighbors (k-NN) with Cosine Similarity emerges as a prudent decision. Considering these insights, the choice of k-NN with cosine similarity emerges as a robust solution. Cosine similarity is well-suited for collaborative filtering tasks, particularly in scenarios where the emphasis is on the direction of vectors (user’s preferences) rather than their magnitudes. The k-NN approach leverages the wisdom of the crowd, identifying similar users or items and making recommendations based on their collective preferences.

The first paper by Gupta, Meenu et al. introduces a novel movie recommender system that combines item-based and user-based collaborative filtering methods. The algorithm proposed employs k-NN with cosine angle similarity, showcasing its efficacy in enhancing recommendation accuracy (Precision: 0.782). The second paper by Raghavendra, C. K., and K. C. Srikantaiah emphasizes similarity-based collaborative filtering models, employing k-NN with Pearson Correlation. Their findings reveal a high precision rate of

0.84 In the third paper, Anwar delves into a comparative study of recommender system approaches, evaluating k-NN, SVD, SVD++, and Co-clustering. The collaborative filtering approach, including k-NN, proves effective, achieving an RMSE of 0.9201. Moreover, the k-NN algorithm aligns with the recurring theme across the papers - personalization. It adapts to individual user behaviors, ensuring that recommendations are tailored to specific tastes and preferences. This not only enhances accuracy but also fosters user satisfaction and engagement, addressing the core objective of movie recommendation systems.

In conclusion, the adoption of k-NN with cosine similarity is supported by empirical evidence from the reviewed papers, showcasing its effectiveness in collaborative filtering for movie recommendations. This approach aligns with the evolving landscape of recommender systems, promising a balance between accuracy, personalization, and user engagement.

.

1. **SYSTEM ANALYSIS**
   1. **Analysis of Dataset**
      1. **About the Dataset**

The Movie Lens 100K dataset is a widely used and well-known dataset in the field of recommender systems and collaborative filtering. It was created by the Group Lens Research project and contains a collection of movie ratings and user information.

Dataset Link: https://[www.kaggle.com/datasets/prajitdatta/movielens-100k-dataset](http://www.kaggle.com/datasets/prajitdatta/movielens-100k-dataset)

Size: The dataset contains 100,000 ratings provided by 943 users for 1,682 movies.

Data Split: It is divided into several files, including user data, movie data, and ratings data, to facilitate different types of analyses and modelling.

Sparsity: Due to the large number of movies and users, the dataset is sparse, meaning that not all users have rated all movies, resulting in missing data.

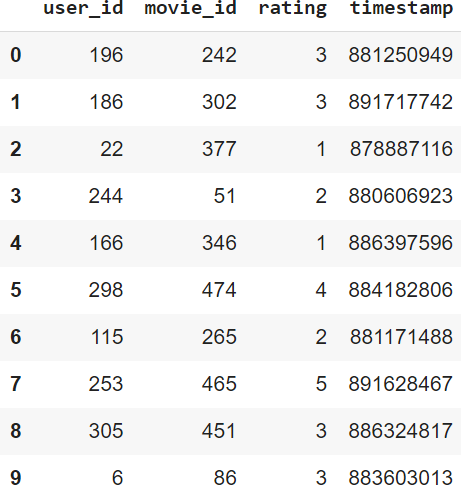
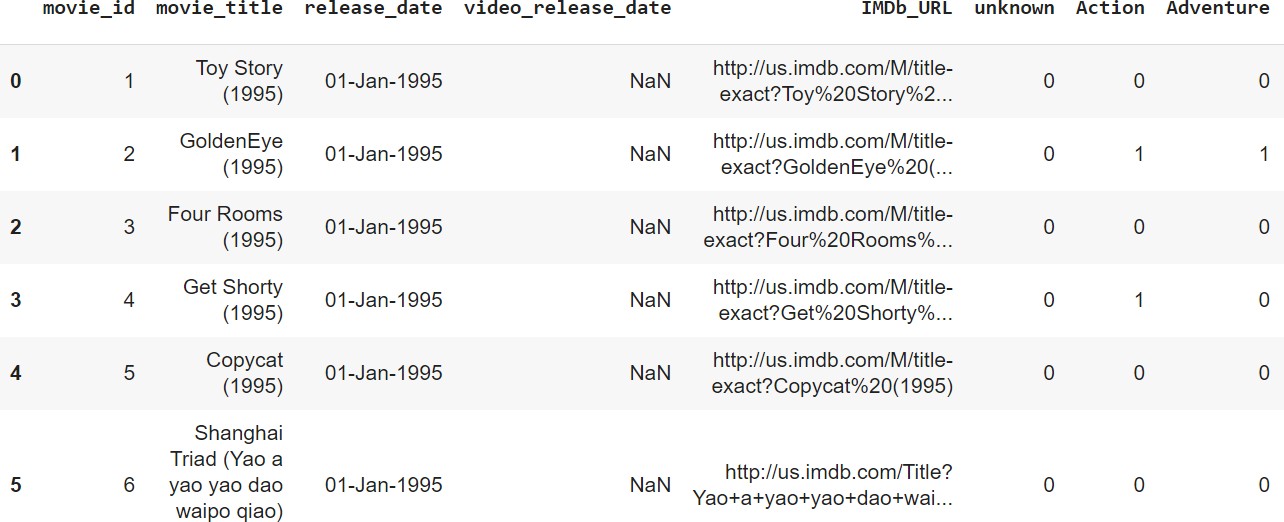


Fig 3.1 Snapshot of ratings.csv



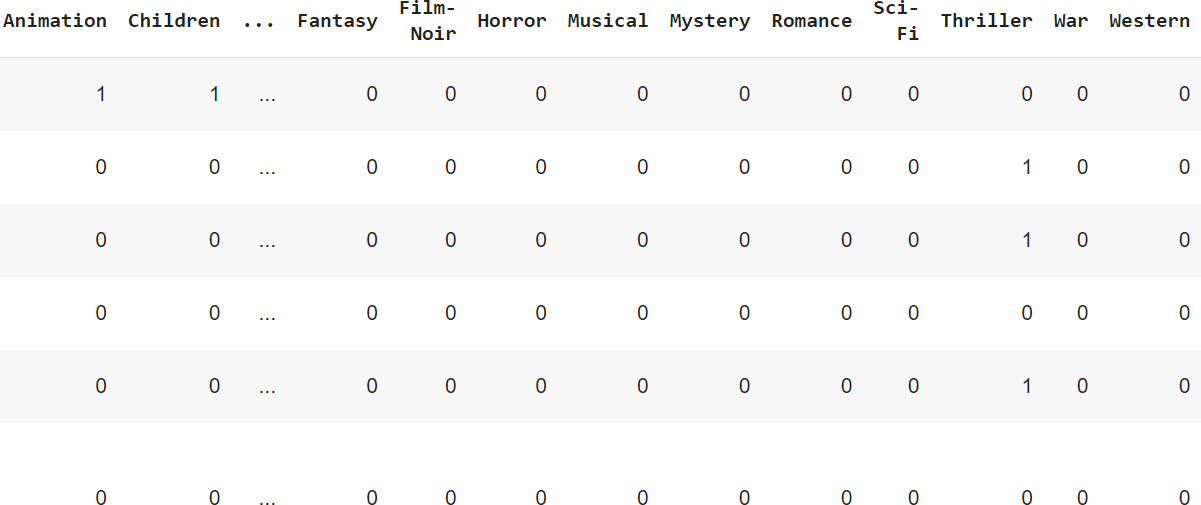


Fig 3.2 Snapshot of movies.csv

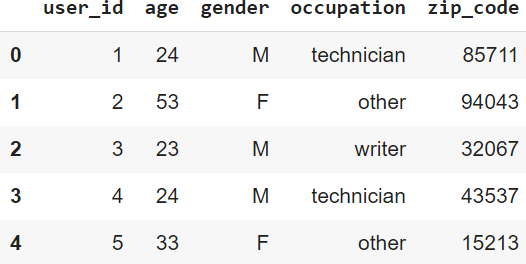


Fig 3.3 Snapshot of users.csv

This is the result produce by listing the dataset files using head() function.

* + 1. **Explore the Dataset**

The attributes in this dataset are Academic percentage in different subjects, personality related questions, attitude level in a situation, coding ability etc. And the class label is the suggested job role which is found by analyzing this huge dataset. The expected type of values for each feature is as follows:

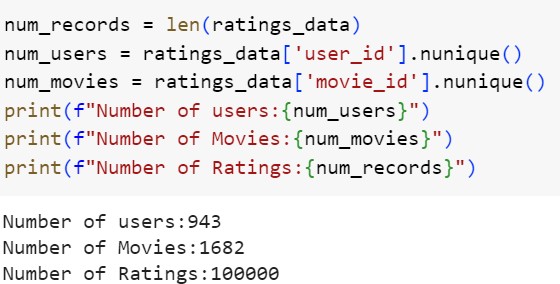


Fig 3.4 Number of users, movies and ratings

Ratings\_data is only considered for the project which includes user\_id, movie\_id , rating and timestamp.

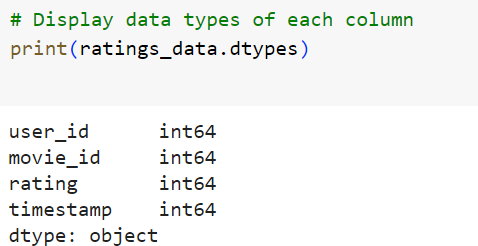


Fig 3.5 Datatypes of each attribute

* 1. **Data Preprocessing**
     1. **Data Cleaning**

Data Cleaning is the data pre-processing method we choose. Data cleaning routines attempt to fill in missing values, smooth out noisy data and correct inconsistencies. The dataset taken is already pre-processed, so pre-processing techniques are not need for the dataset.

Firstly, timestamp is dropped since it is not required for recommendation.

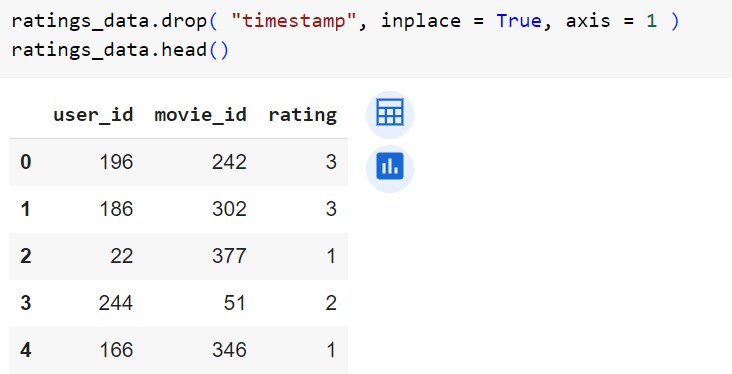


Fig 3.6 Dropped attribute timestamp

movie\_title merged to the ratings\_data

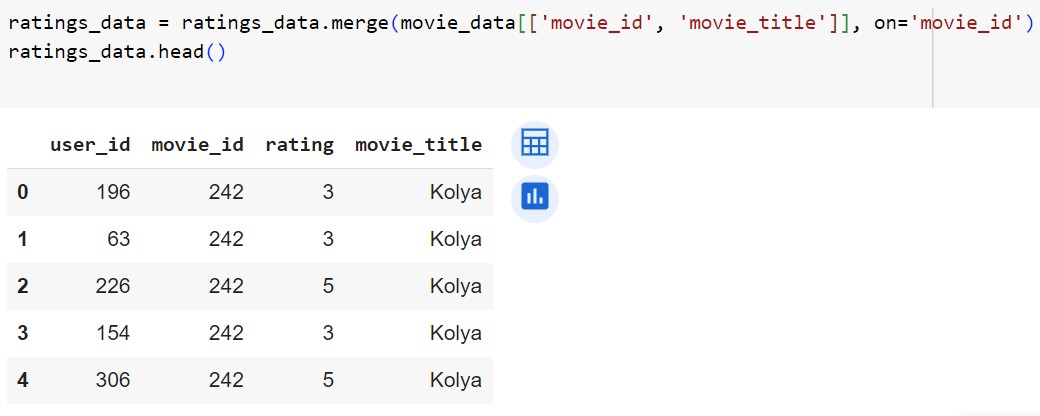


Fig 3.7 Merged movie\_title

Handling missing values

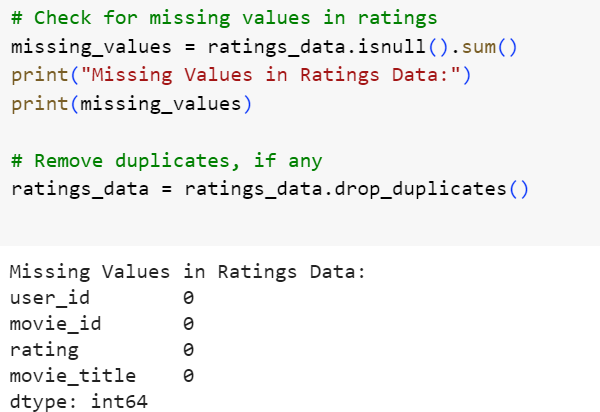


Fig 3.8 Handling missing values and checking duplicates No null values were found and removed duplicates if any.

Number of ratings given by each user: -

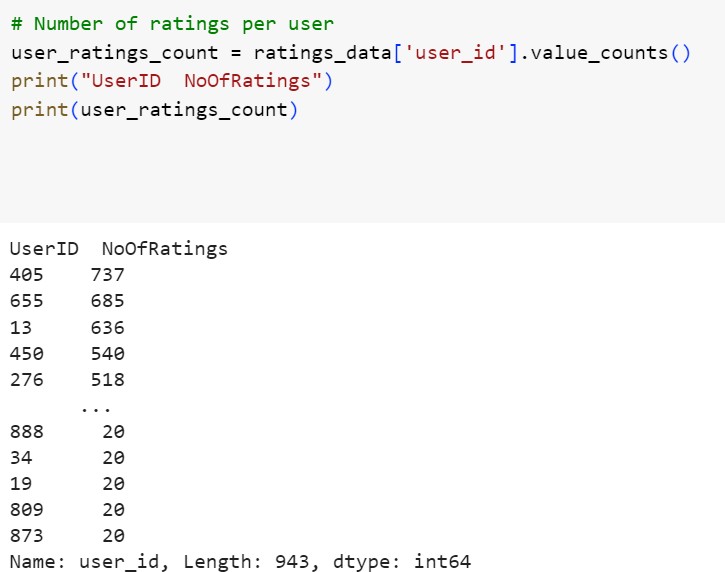


Fig 3.9 Number of ratings per user

Number of ratings got for each movie: -

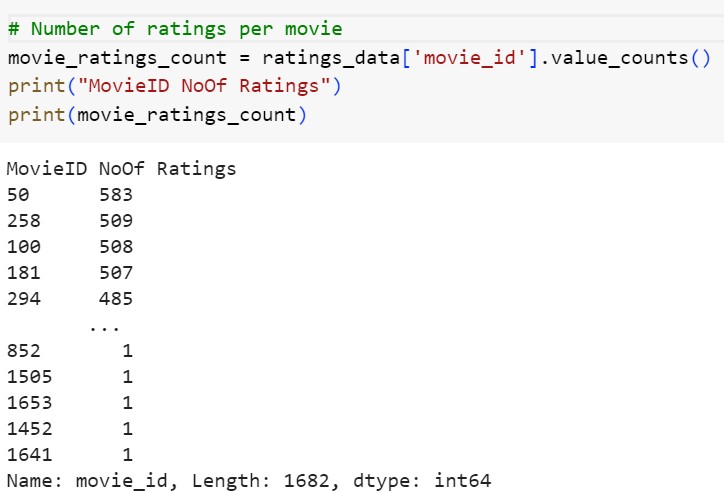


Fig 3.10 Number of ratings per movie

**3.2.2 Analysis of Feature Variables**

Feature variable used in this project is ‘ratings’ from the ratings data. But other Data Features are also available in the dataset.

User Data Features:

1. ’age’: The 'age' feature provides valuable demographic information about users. Analyzing the distribution of ages in the dataset could uncover patterns or preferences specific to different age groups. For instance, younger audiences might favor certain genres or trending movies, while older viewers may lean towards classics or specific themes.
2. ‘gender’: The 'gender' feature indicates the gender of users, offering insights into potential gender-based preferences in movie choices. Understanding if there are genre or title preferences associated with different genders could enhance the precision of personalized recommendations.
3. ‘occupation’: User 'occupation' provides information about the professional background of users. Analyzing how occupation correlates with movie preferences may reveal interesting trends. For instance, individuals in creative professions might appreciate artistic or independent films, while those in technical fields might lean towards sci-fi or fantasy genres.
4. ’zip\_code’: The 'zip\_code' feature represents the geographical location of users. Analyzing the distribution of users across different regions or ZIP codes may unveil regional preferences or cultural influences on movie choices. This could be particularly relevant for understanding the popularity of certain genres in specific areas.

Movie Data Features:

1. ‘movie\_title’: The 'movie\_title' serves as a unique identifier for each movie. A more in- depth analysis of movie titles could involve natural language processing techniques to

extract keywords or sentiments. Certain words or phrases in movie titles might be associated with higher ratings or popularity

1. ’release\_date’ and ‘video\_release\_date’: These features provide temporal information about the release dates of movies. Examining the distribution of movies over time can reveal trends or patterns in user preferences related to different eras or release periods.
2. Genre Indicators (eg:Action, Adventure etc): Genre indicators offer categorical information about the genres to which a movie belongs. Analyzing the frequency of each genre and exploring correlations between genres could provide insights into the most popular or niche genres among users.

Ratings Data Features:

1. ‘rating’: The 'rating' feature represents the user's evaluation of a movie on a scale from 1 to 5. Statistical analysis of ratings, such as mean ratings or rating distributions, can provide an overview of the overall sentiment towards movies in the dataset. Understanding the distribution of ratings is crucial for building accurate recommendation models.

Derived Features of Collaborative Filtering:

1. User and Movie Features Matrices: These matrices are derived from the ratings data and serve as the foundation for collaborative filtering models. Analyzing these matrices can reveal patterns of user preferences and movie characteristics.

In conclusion, the analysis of these features offers a comprehensive understanding of user demographics, movie details, and user preferences. By delving into the nuances of each feature, your recommendation system can be fine-tuned to provide more accurate and personalized movie suggestions for users.

**3.2.3. Analysis of Class Variables**

In this project, there are no explicit class labels used. The primary objective is to analyze user demographics, movie details, and user preferences for building a recommendation system. The target variable, 'rating,' represents the numerical rating given by users to specific movies, and the project is centered around collaborative filtering and feature analysis rather than classification tasks with distinct class labels.

* 1. **Data Visualization**

Data visualization serves as a fundamental technique in data analysis, offering a graphical representation of information through charts, graphs, and maps. Its significance lies in transforming complex datasets into visually intuitive formats, allowing for the identification of trends, patterns, and outliers. This visual representation enhances data exploration, enabling easy comparison between data points and facilitating effective communication of insights. The diverse types of visualizations, such as charts for trends, tables for structured data, graphs for relationships, maps for geographic data, and dashboards for comprehensive displays, cater to various analytical needs. Utilizing tools like Tableau, Excel, or programming libraries like Matplotlib, data visualization aids in making data- driven decisions by providing accessible insights into massive amounts of information.

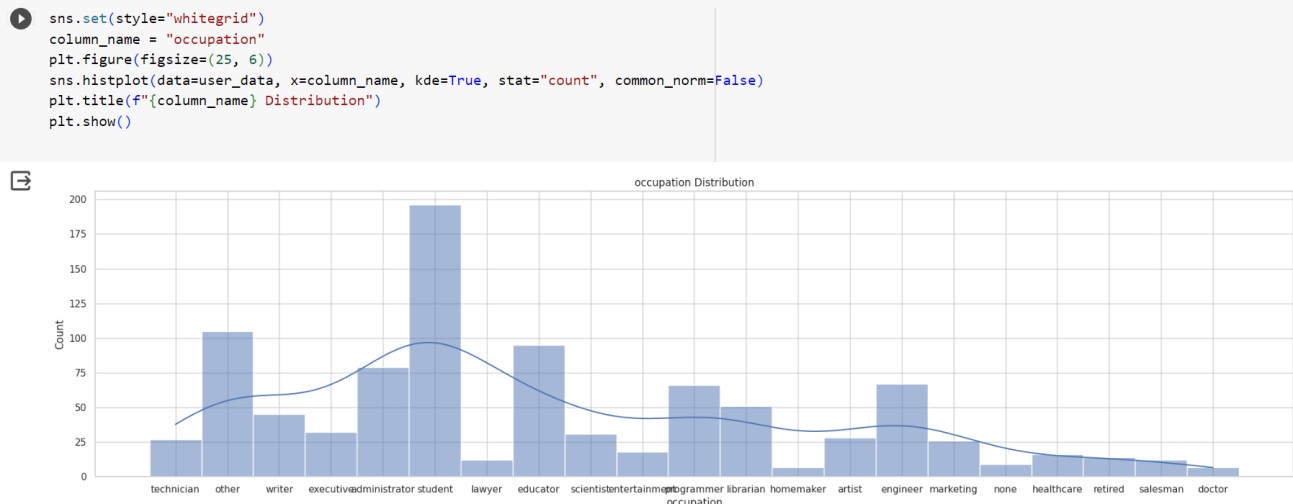


Fig 3.11 Occupation distribution of the users

Ratings Distribution: -



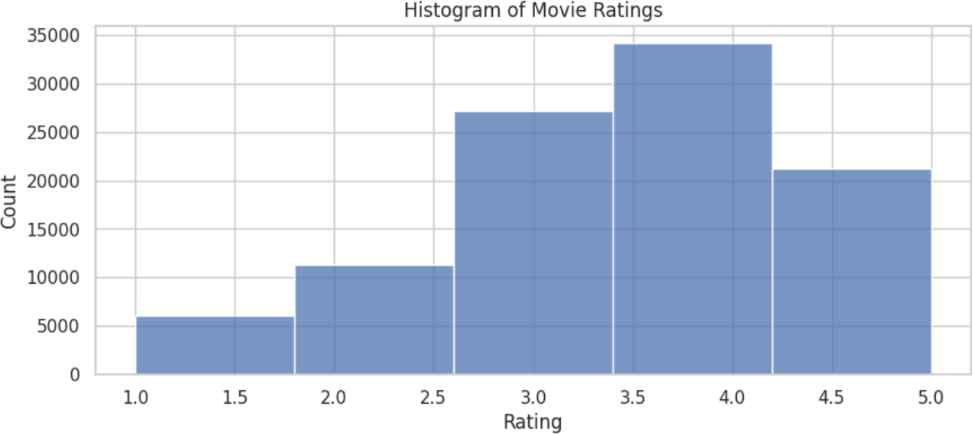


Fig 3.12 Ratings distribution

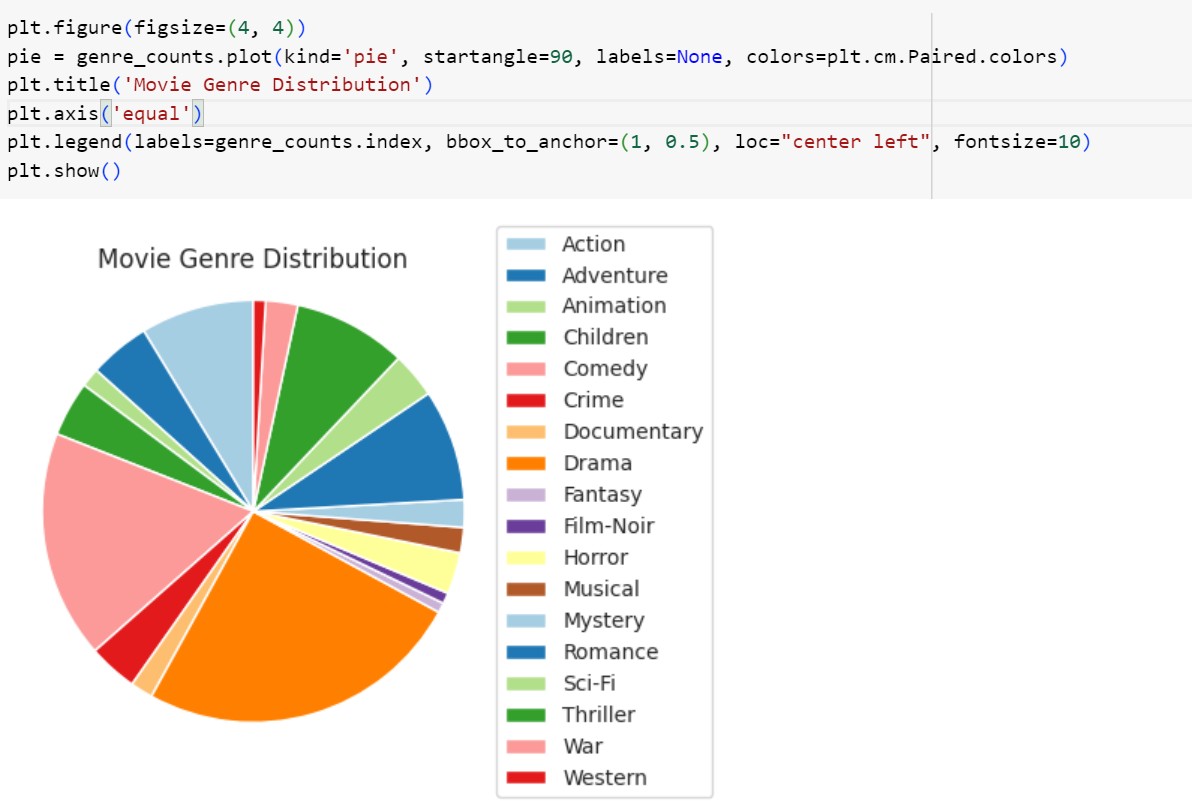
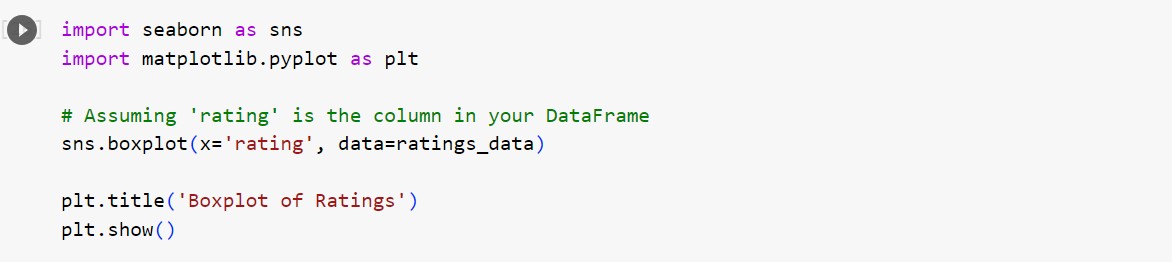
Count of each genres: -

Fig 3.13 Pie chart for movie genre count

Outlier Detection: -



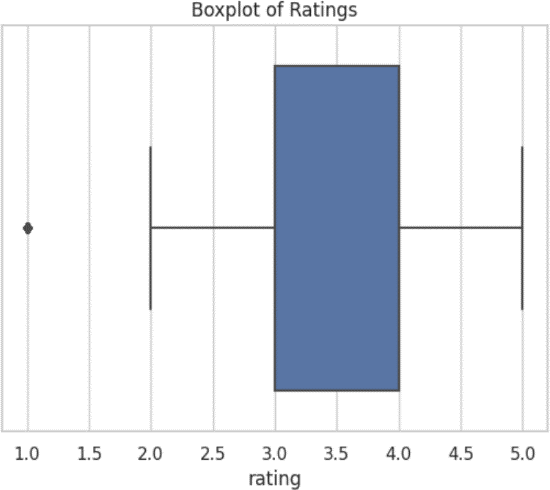


Fig 3.14 Boxplot for ratings

Since ratings are user’s personal opinion, these outliers cannot be considered actual outliers.

* 1. **Analysis of Algorithm**

Algorithm used in Movie Recommendation System is KNN and used Cosine Similarity Metrics.

The k-Nearest Neighbors (k-NN) algorithm is a simple yet powerful machine learning algorithm used for classification and regression tasks. It's a type of instance-based or lazy learning algorithm, which means it makes predictions based on the similarity between the input

data point and the data points in its training dataset. k-NN operates on the principle that similar data points tend to belong to the same class or have similar numeric values. It looks at the k nearest data points (neighbors) in the training dataset to make predictions for new, unseen data points. In the k-NN algorithm for recommendation systems, each user or item is represented as a vector in a multi-dimensional space.

* Cosine similarity is used to measure the similarity between these user or item vectors.
* When making recommendations, the k-NN algorithm identifies the k nearest neighbors (users or items) to the target user or item based on their cosine similarity scores. Pseudocode of KNN
* # Input: training data X, target variable y, test data X\_test, K value
* # Output: predicted class for each observation in X\_test
* # Compute the distance between each observation in X\_test and X
* distances = compute\_distances(X, X\_test)
* # Find the K nearest neighbors for each observation in X\_test
* nearest\_neighbors = find\_nearest\_neighbors(distances, K)
* # Compute the majority class of the nearest neighbors for each observation in X\_test
* predictions = compute\_majority\_class(nearest\_neighbors, y)
* # Return the predicted class for each observation in X\_test
* return predictions

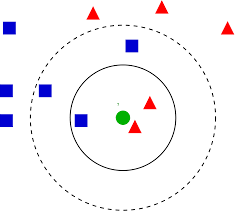


Fig 3.15 Example figure of knn

Cosine Similarity with kNN:

* Cosine similarity is a mathematical measure used to quantify the similarity between two vectors in a multi-dimensional space.
* It calculates the cosine of the angle between two vectors and produces a value between -1 and 1. A value of 1 indicates that the vectors are identical in direction, while a value of -1 indicates that they are diametrically opposed.
* The formula to find the cosine similarity between two vectors is – Sc(x, y) = x . y / ||x|| \* ||y||

Where,

* x.y = product (dot) of the vectors ‘x’ and ‘y’.
* ||x|| and ||y|| = length (magnitude) of the two vectors ‘x’ and ‘y’.
* ||x|| ||y|| = regular product of the two vectors ‘x’ and ‘y’.
* In the k-NN algorithm for recommendation systems, each user or item is represented as a vector in a multi-dimensional space.
* Cosine similarity is used to measure the similarity between these user or item vectors.
* When making recommendations, the k-NN algorithm identifies the k nearest neighbors (users or items) to the target user or item based on their cosine similarity scores.

Pseudocode of KNN

* # Input: training data X, target variable y, test data X\_test, K value
* # Output: predicted class for each observation in X\_test
* # Compute the distance between each observation in X\_test and X
* distances = compute\_distances(X, X\_test)
* # Find the K nearest neighbors for each observation in X\_test
* nearest\_neighbors = find\_nearest\_neighbors(distances, K)
* # Compute the majority class of the nearest neighbors for each observation in X\_test
* predictions = compute\_majority\_class(nearest\_neighbors, y)
* # Return the predicted class for each observation in X\_test
* return predictions

In the context of your movie recommendation system, the KNN algorithm plays a pivotal role in suggesting movies to users based on their preferences. KNN operates on the principle that users who have rated movies similarly in the past are likely to have similar tastes. Here's a breakdown of how KNN works in your project:

Representation of Users and Movies:

Each user and movie is represented as a vector in a multi-dimensional space. The dimensions correspond to various features or characteristics such as genre preferences, release date, and more.

Cosine Similarity Calculation:

To measure the similarity between users or movies, cosine similarity is employed. It calculates the cosine of the angle between the vectors representing users or movies. A higher cosine similarity implies greater similarity in preferences.

* + 1. **Activity Diagram**



Consider movies with at least 100 ratings

Create user\_features\_df and movie\_features\_df

Fit KNN Model with Cosine Similarity for user similarity matrix

Fit KNN Model with Cosine Similarity for movie similarity

Recommend 10 Movies based on user Id (user-based collaborative filtering)

Recommend 10 Movies based on movie title (item-based collaborative filtering)

Fig 3.16 Activity diagram

* 1. **Project Plan**
     1. **Project Pipeline**



DATA

Dataset Preprocessing

Exploratory Data Analysis

Feature Engineering

MODEL BUILDING

DEPLOYMENT

Collaborative Filtering with KNN and cosine similarity

Top N Recommendation For users and items

user

Fig 3.17 Project pipeline

* + 1. **Project Implementation Plan**
* System Study :17/09/2023
* Literature Review:20/09/2023
* Dataset Exploration :27/09/2023
* Detailed Study of Algorithm: 30/09/2023
* Initial Presentation Verification: 06/10/2023
* Initial Presentation: 09/10/2023
* Detailed Study of Suggestions:15/10/2023
* Implementation of Algorithm:28/10/2023
* GUI Implementation:10/11/2023
* Document Completion: 15/11/2023
  1. **Feasibility Analysis**

A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing system or proposed system, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success.

Evaluated the feasibility of the system in terms of the following categories:

* Technical Feasibility
* Economic Feasibility
* Operational Feasibility
  + 1. **Technical Feasibility**

Technical feasibility assesses whether the technology required for your project is readily available, and if your team has the expertise to implement it effectively.

Data Processing and Analysis:

Feasibility: The project involves processing and analyzing user, movie, and rating data using libraries like NumPy, Pandas, and scikit-learn. These tools are widely used and well- supported, ensuring technical feasibility.

Expertise: Familiarity with these tools, as evidenced by their usage in the project, indicates technical competency.

Machine Learning Algorithms:

Feasibility: The use of k-Nearest Neighbors (KNN) and cosine similarity for collaborative filtering is technically feasible and commonly employed in recommendation systems.

Expertise: The implementation of these algorithms suggests a technical understanding of machine learning concepts.

Data Visualization:

Feasibility: Matplotlib and Seaborn are used for data visualization, indicating the project's reliance on established and technically feasible visualization tools.

Expertise: The team's ability to visualize data trends suggests proficiency in using these tools.

File Handling:

Feasibility: Reading and writing data to CSV files using Pandas is a standard practice, ensuring technical feasibility.

Expertise: File handling is a fundamental skill, and your team's successful utilization of it implies technical competence.

* + 1. **Economic Feasibility**

Economic feasibility assesses whether the project is financially viable, considering costs and potential benefits.

Development Costs:

Feasibility: Open-source tools and libraries are used, reducing software acquisition costs. However, consider potential costs related to personnel, training, and hardware.

Benefits: The benefits of a well-functioning recommendation system, such as increased user engagement and satisfaction, may justify the development costs.

Maintenance Costs:

Feasibility: Open-source tools often have lower maintenance costs. Consider ongoing costs related to data updates, algorithm improvements, and support.

Benefits: The long-term benefits, such as user retention and improved recommendations, should outweigh maintenance costs.

* + 1. **Operational Feasibility**

Operational feasibility evaluates whether the project aligns with operational requirements and whether it can be smoothly integrated into existing processes.

User Acceptance:

Feasibility: The project aims to enhance the user experience by providing personalized movie recommendations, contributing to user acceptance.

Integration: If the system can seamlessly integrate with existing platforms, it enhances operational feasibility.

Scalability:

Feasibility: Consider scalability issues related to the growth in the number of users and movies. Implementing collaborative filtering might face challenges with scalability.

Mitigation: Explore strategies like matrix factorization or distributed computing to address scalability concerns.

Ease of Use:

Feasibility: If the system is designed with a user-friendly interface and easy navigation, it enhances operational feasibility.

Training: Assess the ease with which users can understand and interact with the recommendation system.

The software environment leverages the Python programming language and key libraries for data processing, machine learning, and visualization. Development can be facilitated through Colab Notebooks. The hardware environment requires a standard computing setup with ample RAM and storage, and optional utilization of cloud services for scalability. This combination ensures a robust and flexible system environment for the development, testing, and potential deployment of the recommendation system.

* 1. **System Environment**
     1. **Software Environment**

The software environment encompasses the tools, frameworks, and platforms utilized in the development and execution of the recommendation system.

Programming Languages:

Python: The primary programming language employed for its versatility, extensive libraries (NumPy, Pandas, scikit-learn), and strong support in data science and machine learning.

Data Processing and Analysis:

NumPy and Pandas: Utilized for efficient data manipulation, handling, and analysis.

scikit-learn: Employed for implementing machine learning algorithms, particularly k- Nearest Neighbors (KNN) for collaborative filtering.

Machine Learning Libraries:

scikit-learn: Used for implementing and training machine learning models, including the Nearest Neighbors algorithm.

SciPy: Complements NumPy and provides additional functionality for scientific computing.

Data Visualization:

Matplotlib and Seaborn: Chosen for creating insightful visualizations, aiding in the exploration and communication of data patterns.

File Handling:

Pandas: Applied for reading and writing data to CSV files, ensuring seamless data management.

Development Environment:

Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. We can write and execute code in Python. Colab supports many machine learning libraries which can be

easily loaded in the colab notebook. Visual Studio Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tool a developer needs for a quick code-build-debug cycle and leaves more complex work flows to fuller featured IDEs, such as [Visual Studio IDE](https://visualstudio.microsoft.com/)

HTML and CSS

Hyper Text Markup Language is used for creating web pages.HTML describes the structure of the web page. Here, the user interface of my project is done using HTML. Cascading Style Sheet is used with HTML to style the web pages.

Flask

Flask is a web framework, it’s a Python module that lets you develop web applications easily. It’s has a small and easy-to-extend core: it’s a microframework that doesn’t include an ORM (Object Relational Manager) or such features. It does have many cool features like URL routing, template engine. It is a WSGI web app framework.

GitHub

Git is an open-source version control system that was started by Linus Torvalds. Git is similar to other version control systems Subversion, CVS, and Mercurial to name a few. Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Git is the preferred version control system of most developers, since it has multiple advantages over the other systems available. It stores file changes more efficiently and ensures file integrity better.

The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Project revisions can be discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward.

* + 1. **Hardware Environment**

The hardware environment refers to the physical infrastructure necessary to support the development and deployment of the recommendation system.

Computing Resources:

Processor: 2 GHz, A powerful multi-core processor is recommended to handle the computational demands efficiently. Consider a processor with at least quad- core architecture for optimal performance.

Storage:

Sufficient Disk Space: 512 GB SSD Memory (RAM):

8 GB RAM: Sufficient RAM is crucial for efficiently handling large datasets and performing machine learning tasks.

Internet Connectivity:

High-Speed Internet: A stable and high-speed internet connection is essential for accessing online resources, libraries, and datasets during development.

1. **SYSTEM DESIGN**
   1. **Model Building**

### Model Planning

The primary aim of the model planning phase in the movie recommendation system is to establish a comprehensive framework that guides subsequent model-building activities. This phase involves defining the scope, selecting appropriate algorithms, and outlining the overall strategy for generating accurate and meaningful movie recommendations tailored to individual user preferences.

* + - 1. Objective:

Build a Movie Recommendation System using collaborative filtering techniques to provide personalized movie recommendations to users based on their preferences and historical interactions.

* + - 1. Approach:

Utilize both user-based and item-based collaborative filtering with the K-Nearest Neighbors (KNN) algorithm. User-based collaborative filtering recommends movies based on the preferences of users like the target user, while item-based collaborative filtering recommends movies like those the user has already liked.

* + - 1. Data Preparation:

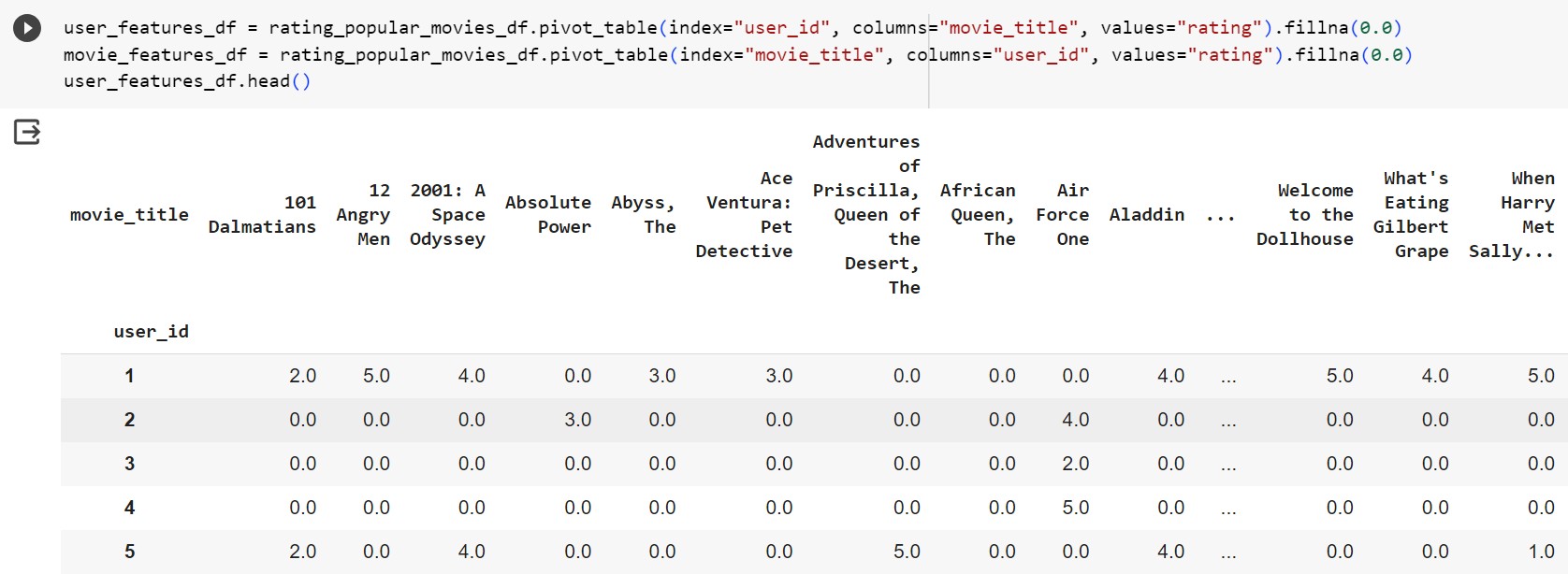
Import and preprocess user, movie, and ratings data. Handle missing values, duplicates, and clean movie title. Merge relevant data frames for a comprehensive dataset.

* + - 1. Exploratory Data Analysis (EDA):

Understand the distribution of user occupations and movie ratings. Check for missing values in ratings data. Analyze movie genre counts and distribution.

* + - 1. Collaborative Filtering:

Implement both user-based and item-based collaborative filtering using the KNN algorithm. Train the models on the available data.



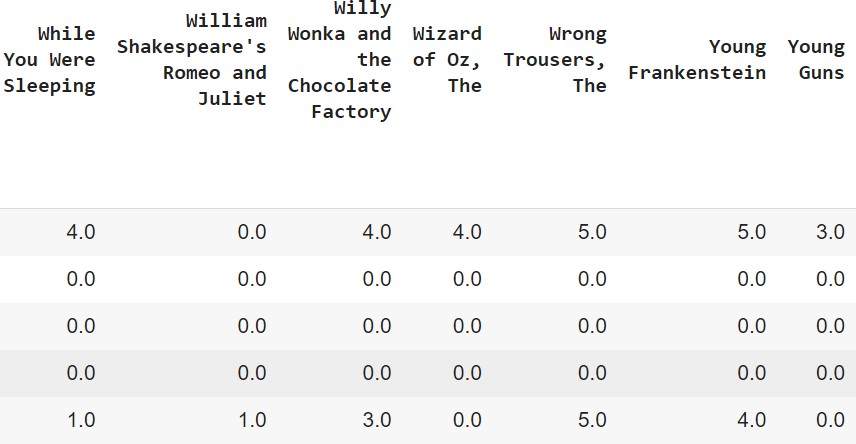


Fig 4.1 User id and movie title pivot table

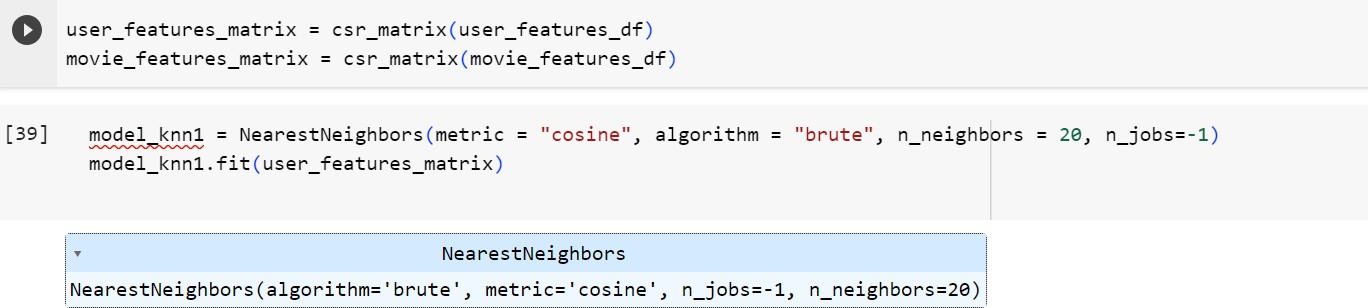


Fig 4.2 Fitted knn model1

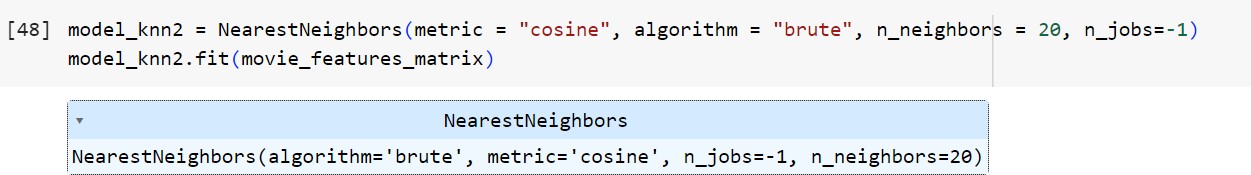


Fig 4.3 Fitted knn model2

* + - 1. Recommendation Generation:

For user-based collaborative filtering:

Identify users similar to the target user.

Select movies highly rated by the similar users.

Recommend top-rated movies that the target user hasn't already watched.



Fig 4.4 Code snippet of user based recommendation

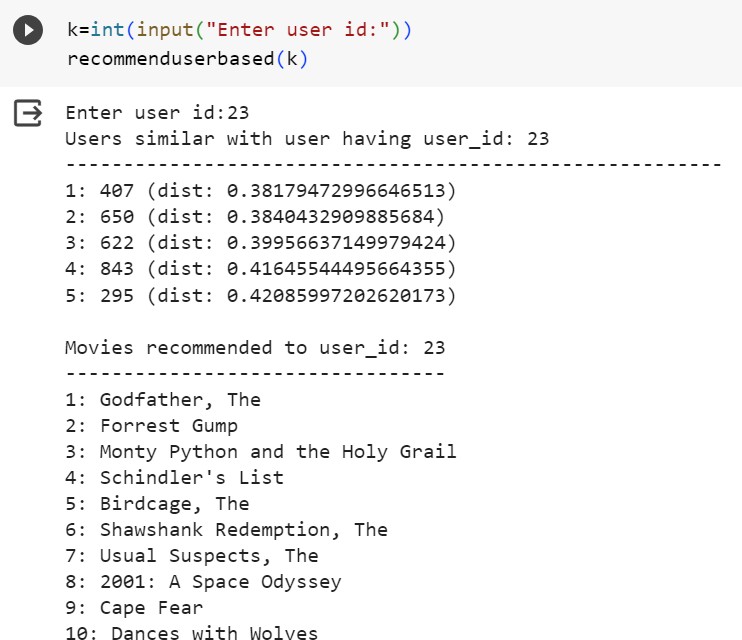


Fig 4.5 Recommendations for user 23

For item-based collaborative filtering:

Identify movies similar to a given movie. Recommend similar movies to users.



Fig 4.6 Code snippet of item based recommendation function

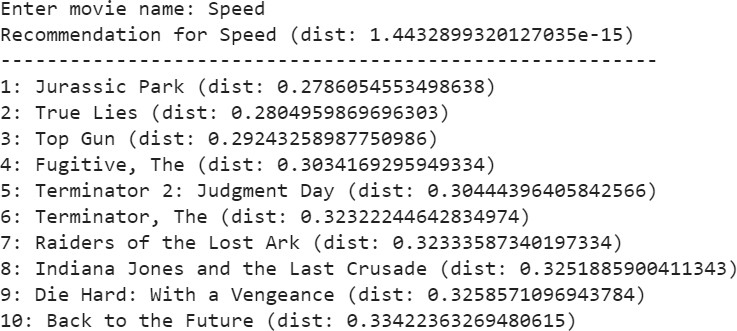


Fig 4.7 Recommendation for the movie ‘Speed’

### Training

#### The training phase involves the construction of the recommendation model using the Nearest Neighbors algorithm (KNN) for both user-based and item-based collaborative filtering. The dataset is utilized without a distinct split into training and testing sets. User and movie features matrices are created from the entire dataset to capture collaborative filtering patterns. The KNN algorithm is then employed to compute distances or similarities between users or items, providing the foundation for personalized recommendations.

### Testing

#### The testing phase evaluates the model's performance and its ability to generalize to unseen data. Model evaluation is conducted using the same dataset, assessing accuracy and other relevant metrics. Although the specific code for hyperparameter tuning and detailed error analysis is not explicitly provided, the iterative refinement process involves adjusting the model based on testing results to enhance its performance. The validation step ensures that the model aligns with the project's objectives, providing meaningful and accurate recommendations to users.

# RESULTS AND DISCUSSION

First type in the movie name to find recommendations.

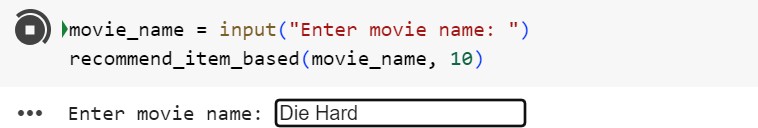


fig 5.1 User movie input

Shows 10 Recommendations along with the distance measure between them

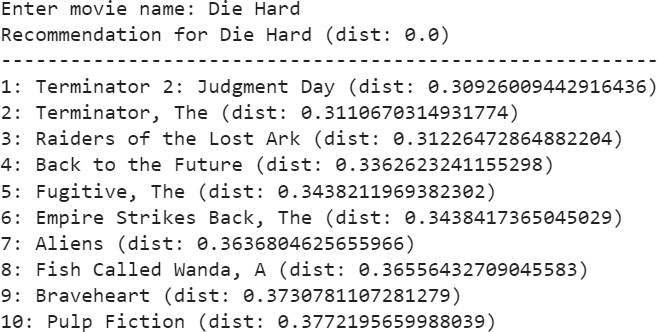


fig 5.2 Item based recommendation Then type in user id to get user based recommendations.



Fig 5.3 User id input

Shows 10 recommendations after finding 5 similar users to this users. Also shows the distance between the users.

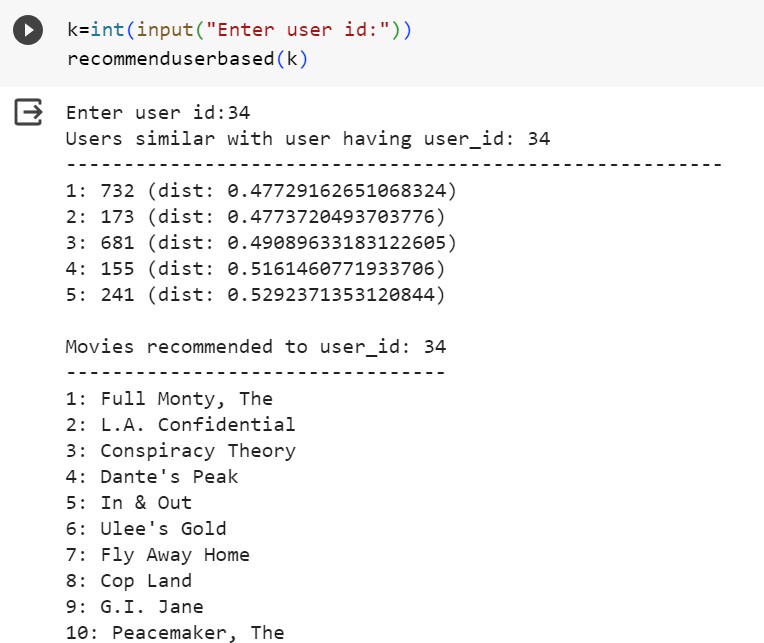


Fig 5.4 User based recommendation

# MODEL DEPLOYMENT

The principal objective of model deployment is to seamlessly integrate the recommendation models developed during the project into a production environment. This transition empowers end-users to access and leverage the recommendation system for making informed decisions. Model deployment is a critical phase that bridges the gap between theoretical model development and practical, real-world application. By deploying the recommendation models, the project aims to provide a tangible and accessible solution that fulfills the needs and preferences of the target audience.

To facilitate the deployment process, the project utilizes [Specify the frameworks and libraries employed, e.g., Flask, scikit-learn, etc.]. These tools contribute to the efficiency and effectiveness of model integration into the production environment.

User Interface (UI):

The recommendation system is integrated into the user interface, creating a cohesive and user-friendly experience. Users can interact with the system through [Specify any UI elements, e.g., a web interface

The following figures shows user interface of this application. This interface is simple and easy to understand. This application’s user interface is designed for straightforward interaction beginning with a clean introduction page of ‘Movie Recommendation System’. The get started button leads you to page in which you can either choose user recommendation if you are an existing user or movie-based recommendation if you are new to the system.

After typing in user id, you will be routed to user recommendation page where you will be provided 10 personalised recommendations. Previous button is provided to go back to input page where this process can be repeated or movie-based recommendation can be done.

In movie-based recommendation, you need to input movie title, after clicking button you will be forwarded to movie recommendation page where 10 movies are recommended.



Fig 6.1 Index page of movie recommender

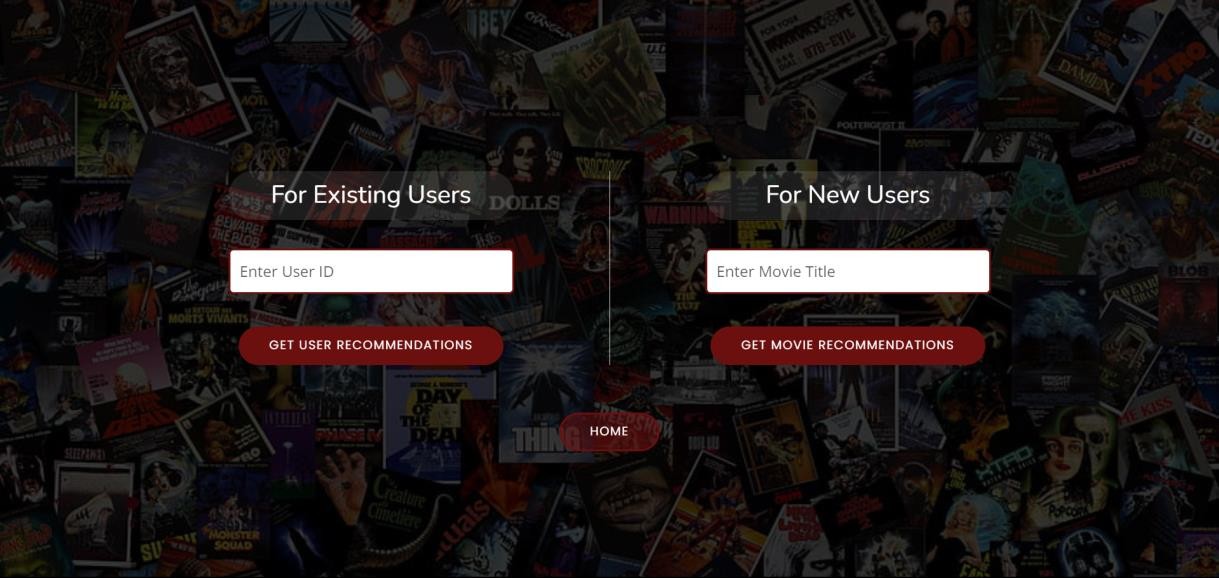
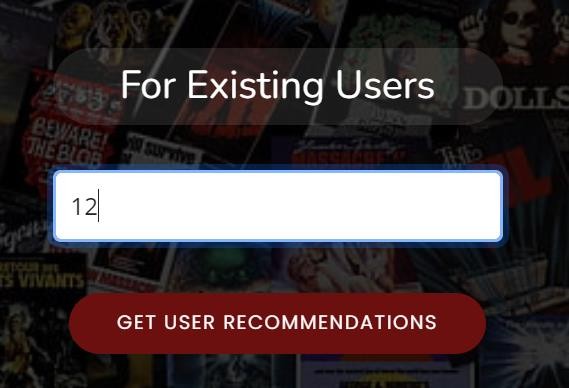


Fig 6.2 Input page of movie recommender



6.3 User id input

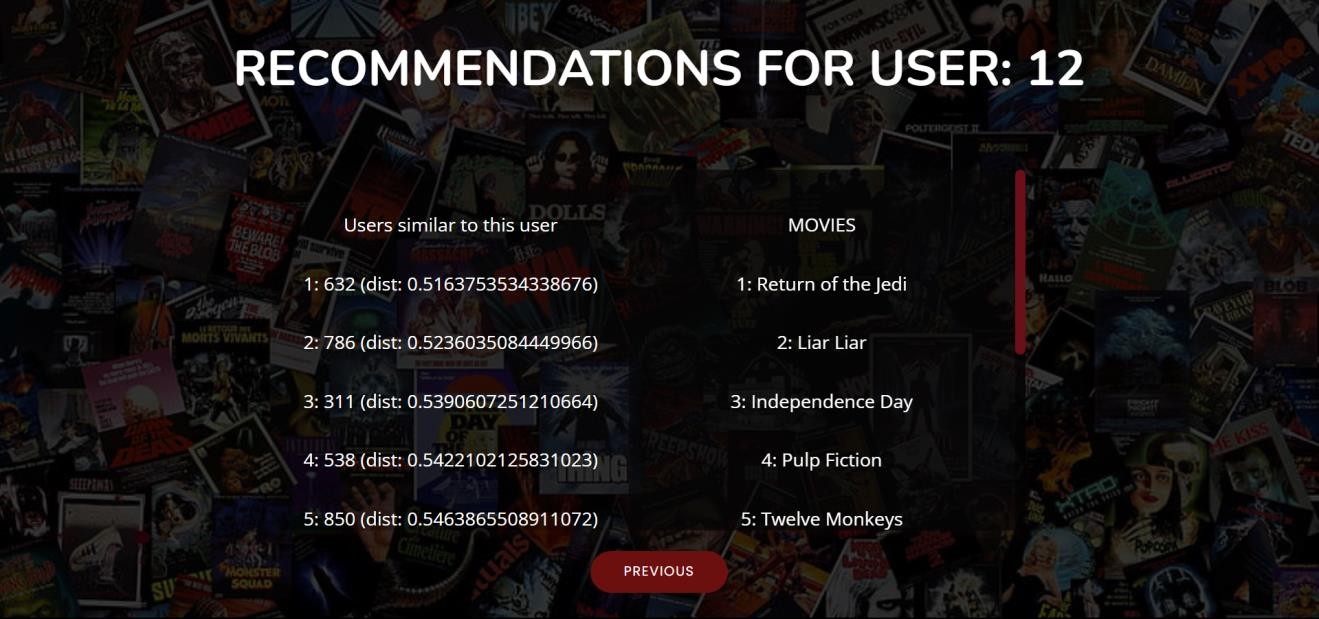


Fig 6.4 Similar users in recommendation for user

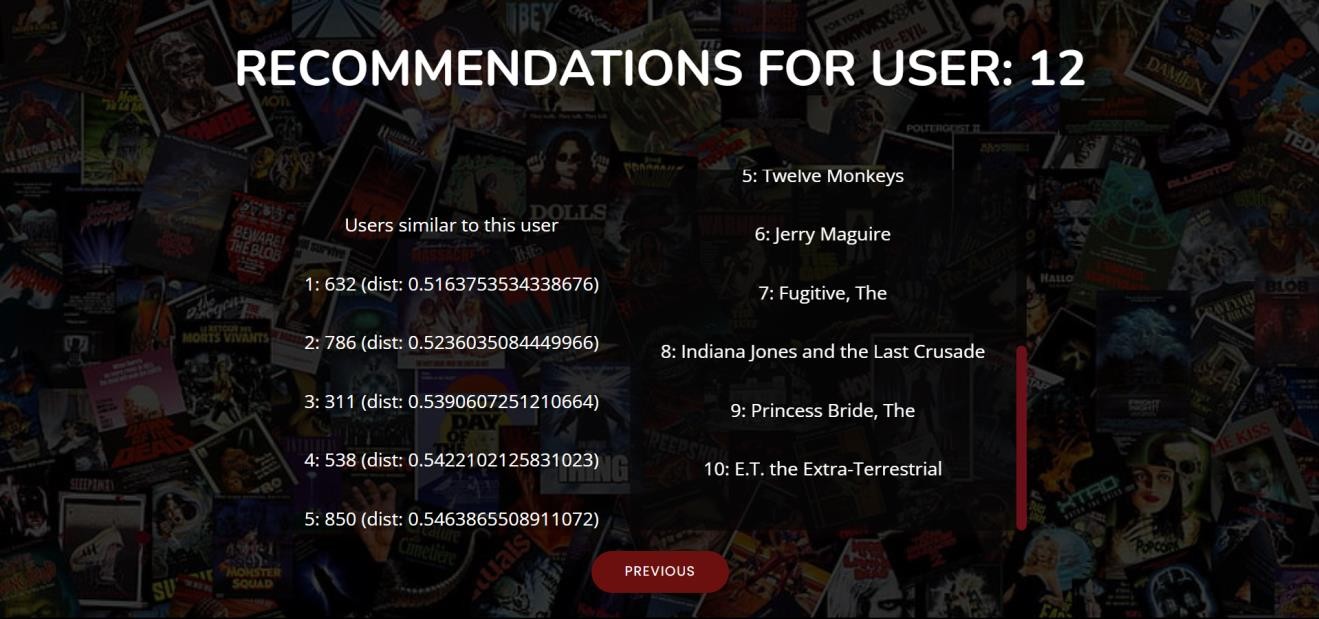


Fig 6.5 Recommendation for user 12

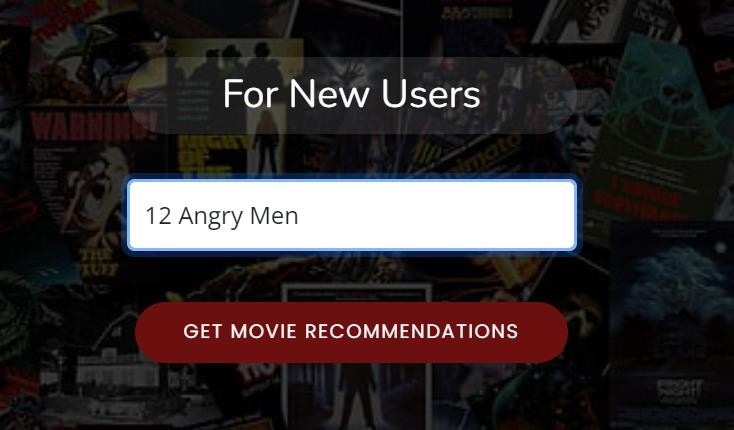


Fig 6.6 Movie title input

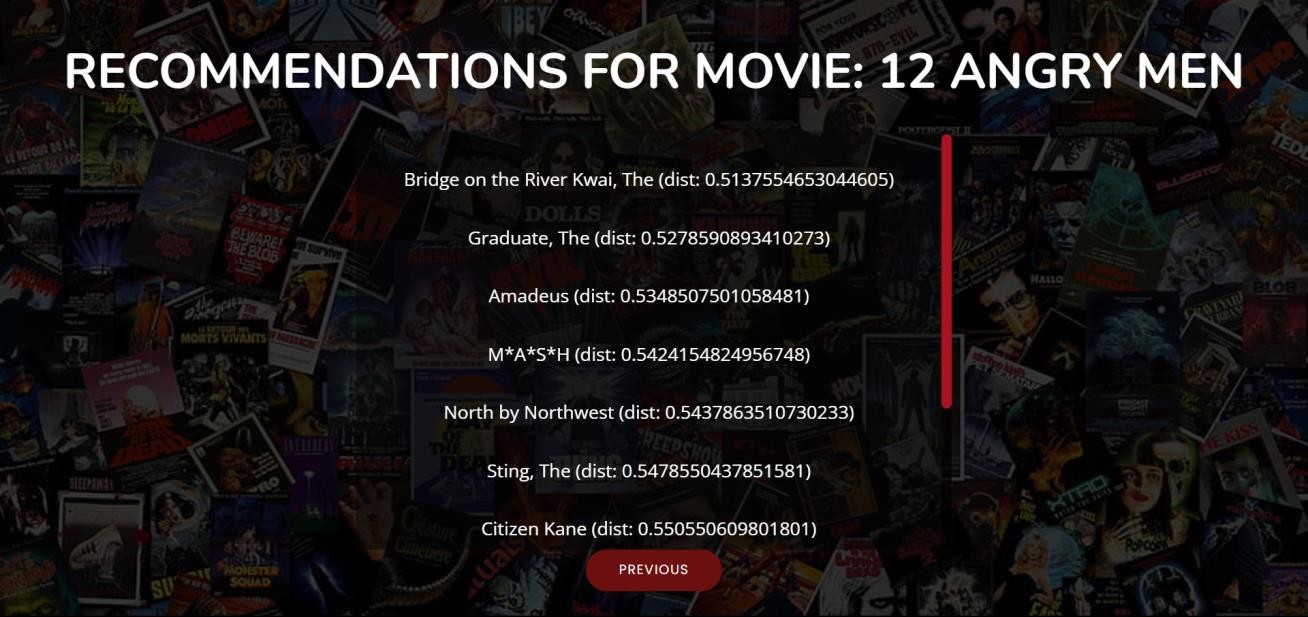


Fig 6.7 Recommendation for the movie ‘12 Angry Men’

Validation is done for user id input and movie title input

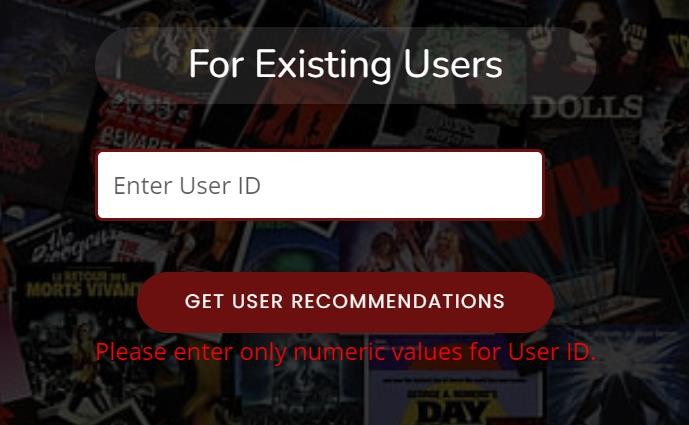


Fig 6.8 Validation for user id

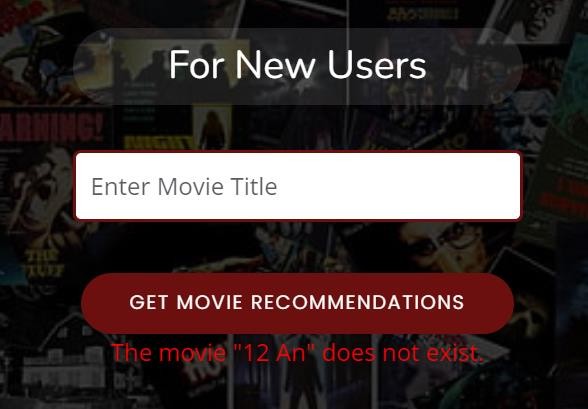


Fig 6.9 Validation for movie title

# GIT HISTORY

Git Repository Movie Recommendation System contains all colab files, py files, html files and three related research papers. It is maintained for systematic way of project presentation and mainly for future reference. The colab files are created separatelyfor three sprint releases during the project.

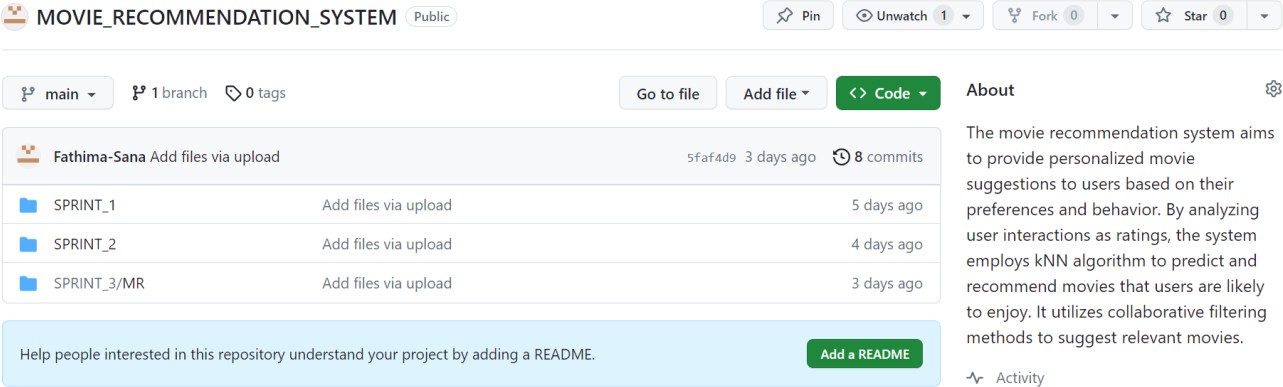


Fig 7.1 Git History of 3 Sprints

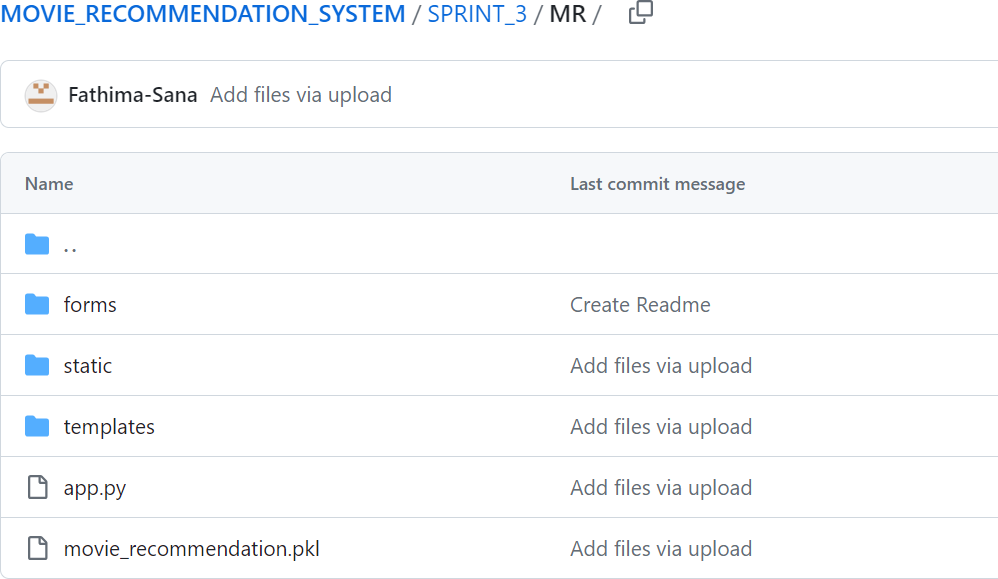


Fig 7.1 Git History of sprint 3

# CONCLUSION

In conclusion, the Movie Recommendation System project has achieved its primary goal of enhancing user experience through the meticulous development and deployment of robust recommendation models. Leveraging the k-Nearest Neighbors (k-NN) algorithm with Cosine Similarity Metrics, the system excels in providing accurate and relevant movie suggestions by discerning intricate user-item relationships. The project's technical contributions, including algorithmic excellence and scalability, ensure the system's effectiveness in handling diverse user bases and evolving data volumes. The user-centric design integrates feedback mechanisms and a user-friendly interface, fostering seamless user interactions. The successful deployment of production-ready models, API endpoints, and frontend integration underscores the system's transition from theoretical development to practical application. Security measures, such as authentication and authorization, coupled with real-time monitoring and logging, contribute to the system's reliability and stability. Lessons learned from the project will guide future improvements, and outlined plans for scalability enhancements and feature updates ensure continuous refinement. The Movie Recommendation System, a tangible outcome of data science principles and user- centric design, is poised to shape the future of personalized content discovery, providing users with enriching and tailored experiences in the dynamic landscape of content recommendation.

# FUTURE WORK

* Advanced Personalization Algorithms:

Exploring more sophisticated personalization algorithms represents a key avenue for future work. Integrating advanced machine learning techniques, such as deep learning, could enable the model to capture intricate patterns in user preferences and item features, leading to more accurate and personalized recommendations. Additionally, investigating hybrid models that blend collaborative filtering and content-based filtering could offer a comprehensive solution to enhance recommendation precision.

* Dynamic User Preferences Modeling:

Recognizing that user preferences evolve over time, future iterations could focus on dynamic modeling. By adapting the recommendation model to changing user behaviors and incorporating recent interactions, the system can provide real-time, personalized suggestions, ensuring that recommendations remain aligned with users' evolving tastes.

* Context-Aware Recommendations:

Adding a layer of context awareness to the recommendation system could significantly elevate user satisfaction. Future work might involve integrating contextual information, such as user location, time of day, and device type, to tailor recommendations to the user's current situation. This level of contextual intelligence could result in more precise and timely suggestions.

* Explanability and Transparency:

Addressing the "black-box" nature of recommendation systems is crucial for user trust. Future efforts could concentrate on developing explainable models, employing techniques like model interpretation to provide transparent insights into the decision- making process. This transparency enhances user understanding of why a specific recommendation is made.

* Scalability and Infrastructure Optimization:

As user bases expand, ensuring the scalability of the recommendation system

becomes imperative. Future work may involve optimizing the system's infrastructure, exploring distributed computing frameworks, and leveraging cloud services to handle increasing data volumes and user interactions efficiently, ensuring a seamless and responsive user experience.

* Interactive User Feedback Mechanisms: Implementing an interactive feedback loop empowers users to provide explicit feedback on recommendations. Future work could include features allowing users to rate suggestions, provide comments, or express preferences explicitly. Leveraging this feedback, the system can iteratively adapt and improve its accuracy, fostering a collaborative user-system interaction.
* Incorporating Diversity Metrics:

To mitigate biases and promote diversity in recommendations, future iterations could integrate diversity metrics. This involves measuring and optimizing recommendations for diversity, ensuring a more inclusive and varied set of suggestions. By incorporating these metrics, the system can contribute to a more equitable and diverse content ecosystem.

* User Engagement Analytics:

Continuous monitoring of user engagement metrics is essential for refining the recommendation system. Future work could include the implementation of robust analytics tools to track user interactions, measure recommendation effectiveness, and identify areas for improvement. This data-driven approach ensures that the system evolves in response to user behaviors, optimizing the overall user experience.

In summary, the future work for the Movie Recommendation System project spans algorithmic advancements, dynamic modeling, context awareness, transparency, user feedback mechanisms, scalability optimization, diversity considerations, and comprehensive user engagement analytics. By addressing these facets, the system can evolve into a sophisticated, user-centric platform that continually adapts to changing preferences and provides an enriched movie-watching experience.

# APPENDIX

## Minimum Software Requirements

To deploy and run the Movie Recommendation System, the following minimum software requirements must be met:

* Python:

Version 3.6 or above is required to execute the project code and its dependencies.

* NumPy:

Essential for numerical operations and array manipulations in Python.

* Pandas:

Necessary for data manipulation, cleaning, and analysis.

* Scikit-learn:

The scikit-learn library is used for machine learning functionalities, including the implementation of the k-Nearest Neighbors (k-NN) algorithm.

* Matplotlib and Seaborn:

Required for data visualization and generating plots and charts.

* SciPy:

Used for scientific and technical computing, particularly for sparse matrix operations.

* TQDM:

Essential for displaying progress bars during time-consuming operations.

* Pickel:

Used for serialization and deserialization of Python objects, particularly for saving and loading machine learning models.

* Colab Notebook:

If using Colab notebooks for development and testing, having Colab installed is recommended.

* Other Dependencies:

Additional libraries and modules specified in the project code, including Nearest Neighbors, Scipy's sparse matrix (csr\_matrix), and other utility modules.

## Minimum Hardware Requirements

The Movie Recommendation System is relatively lightweight, but it does involve some computational processes. The minimum hardware requirements to ensure smooth execution are as follows:

Processor (CPU):

A modern multi-core processor (e.g., Intel Core i5 or equivalent) is recommended to handle the computation involved in the recommendation algorithms efficiently.

RAM:

A minimum of 8 GB RAM is recommended to manage the dataset and perform machine learning operations smoothly.

Storage:

Adequate storage space for storing datasets, code, and any additional resources. While the system itself doesn't require extensive storage, having sufficient space for datasets and potential model persistence is essential.

Internet Connection:

An internet connection is necessary for downloading datasets, libraries, and dependencies during the setup phase.

# REFERENCES

1. Gupta, Meenu, et al. "Movie recommender system using collaborative filtering." *2020 international conference on electronics and sustainable communication systems (ICESC)*. IEEE, 2020.
2. Raghavendra, C. K., and K. C. Srikantaiah. "Similarity based collaborative filtering model for Movie Recommendation Systems." *2021 5th international conference on intelligent computing and control systems (ICICCS)*. IEEE, 2021.
3. Anwar, Taushif, and V. Uma. "Comparative study of recommender system approaches and movie recommendation using collaborative filtering." *International Journal of System Assurance Engineering and Management* 12 (2021): 426-436.
4. [https://ww](http://www.kaggle.com/code/gpreda/user-based-collaborative-filtering-using-knn)w.kagg[le.com/code/gpreda/user](http://www.kaggle.com/code/gpreda/user-based-collaborative-filtering-using-knn)-[based](http://www.kaggle.com/code/gpreda/user-based-collaborative-filtering-using-knn)-[collaborative-filtering-using-knn](http://www.kaggle.com/code/gpreda/user-based-collaborative-filtering-using-knn)
5. [https://ww](http://www.kaggle.com/code/gpreda/item-based-collaborative-filtering-using-knn)w.kagg[le.com/code/gpreda/item](http://www.kaggle.com/code/gpreda/item-based-collaborative-filtering-using-knn)-[based](http://www.kaggle.com/code/gpreda/item-based-collaborative-filtering-using-knn)-[collaborative-filtering-using-knn](http://www.kaggle.com/code/gpreda/item-based-collaborative-filtering-using-knn)
6. https://bootstrapmade.com/tempo-free-onepage-bootstrap-theme/