

RECOMMENDER SYSTEM PROJECT WITH HEROKU DEPLOYMENT

A Major Project Report submitted in partial fulfillment for
the degree of Bachelor of Technology

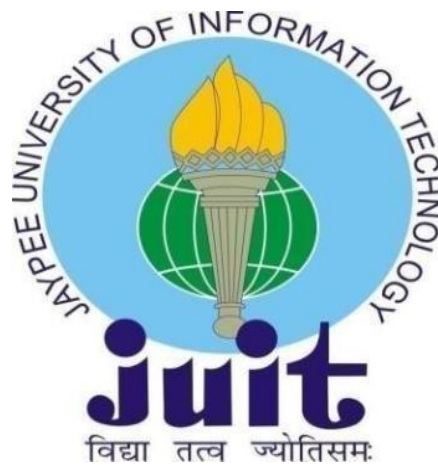
Computer Science and Engineering
By

GAUTMI SINGH (191354)

SHIVAM VERMA (191455)

UNDER THE SUPERVISION OF

DR. JAGPREET SIDHU



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology, Wagnaghat, 173234,
Himachal Pradesh, IN**

CERTIFICATE DECLARATION

I hereby declare that the work presented in this report entitled “Recommender System Project with Heroku Deployment” in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering submitted in the Department of Computer Science & Engineering , Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of Dr. Jagpreet Sidhu, Assistant Professor(SG) , Department of Computer Science and Engineering.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Gautmi Singh, 191354

Shivam Verma, 191455

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

Dr. Jagpreet Sidhu Assistant Professor (SG)
Deptt. of Computer Science and Engineering

Date:

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none">• All Preliminary Pages• Bibliography/Images/Quotes• 14 Words String		Word Counts	
Report Generated on		Submission ID	Character Counts	
			Total Pages Scanned	
			File Size	

Checked by

Name & Signature

.....

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

To begin, I want to express my heartfelt gratitude to almighty God for His heavenly grace, which has enabled us to successfully complete the project work.

Supervisor **Dr. JAGPREET SIDHU**, Assistant Professor (SG), Department of CSE Jaypee University of Information Technology, Wagnaghat, deserves my deepest gratitude. His never-ending patience, intellectual direction, constant encouragement, constant and energetic supervision, constructive criticism, helpful suggestions, and reading numerous poor versions and revising them at every level allowed this project to be completed.

I'd like to thank **Dr. JAGPREET SIDHU**, Department of CSE, for his invaluable assistance in completing my project.

I'd also like to express my gratitude to everyone who has assisted me in making this project a success, whether directly or indirectly. In this unique situation, I'd like to express my gratitude to the different staff members, both teaching and non-teaching, who have provided me with valuable assistance and assisted my project.

Finally, I must express my gratitude for my parents' unwavering support and patience.

Project Group No. : 58

Gautmi Singh ,19354

Shiva Verma ,191455

TABLE OF CONTENT

Sr. No.	Particulars	Page
	Certificate Declaration	I
	Plagiarism Certificate	II
	Acknowledgement	III
	Table of Content	IV-VI
	List Of Abbreviation	VII
	List of Figures	VIII
	Abstract	IX
1	Introduction of Major Project 1.1 Introduction 1.2 Objectives of Project 1.3 Technical Requirements 1.4 Software Requirements 1.5 Deliverables of Project	1-8

2	<p>Literature Survey</p> <p>2.1 Literature Survey</p> <p>2.2 Using K-Means Clustering and KNN</p> <p>2.3 Feasibility Study</p> <p>2.4 Project Definition</p> <p>2.5 Problem Analysis</p> <p>2.6 Solution</p> <p>2.7 Requirements</p> <p>2.8 Objective of Major Project</p>	9-31
3	<p>System Development/ Implementation</p> <p>3.1 Cosine Similarity</p> <p>3.2 Singular Value Decomposition (SVD)</p> <p>3.3 Data Set Used in the Minor Project</p> <p>3.4 Dataset Used in project:</p> <p>3.5 Data Set Features</p>	32-43
4	<p>Performance Analysis</p> <p>4.1 Discussion on Result Achieved</p> <p>4.2 Project Outcome</p>	44
5	<p>Conclusion</p> <p>5.1 Conclusion</p> <p>5.2 Application of Queue Management System</p> <p>5.3 Limitations of Queuing Model</p> <p>5.4 Future Work</p>	45-51

6	References	52-54
7	Appendices	55

LIST OF ABBREVIATION

1. **KNN:** K Nearest Neighbour
2. **SVM :** Support Vector Machine
3. **SVD :** Singular Value Decomposition
4. **DFD:** data flow diagram
5. **RMSE:** Root-mean-square deviation

LIST OF FIGURES

1. Recommendation techniques
2. Content based recommendation system
3. Collaborative filtering
4. Categories table
5. Data flow diagram
6. Movie Metadata_1.csv
7. Movie Metadata_2.csv
8. Movie Metadata_3.csv
9. Credits.csv
10. Movie_metadata.csv after selecting required features
11. Data of movies of year 2017 collected from movies_metadata
12. 'genre_list' evaluated from the JSON object
13. Code snippet_1
14. Code snippet_2
15. CMD_Project Screenshot
16. CMD_Local Host Link
17. Home Page
18. Auto Complete feature
19. Movie Details
20. Particular Movie Detail

ABSTRACT

Nowadays, a recommendation system plays a very important role in various fields including e-commerce and OTT-platforms and has made things easier to find. A recommendation engine recommends the most relevant data to the user by using different algorithms. For watching favourable movies online we can utilise movie recommendation systems, which are more reliable, since searching for 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 SVM 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.

Movie recommendation system is targeted for movie enthusiasts who switch from one platform to another in search of a movie to watch which perfectly fits the choice and mood of the user. The better the recommendation system a platform has, the chance of a user to stay on that platform and do the required transaction over there increases. The main goal of our project is to create a website which helps movie enthusiasts by suggesting what movie to watch without struggling to go through the long process to find a perfect movie from a large list of movies, which is time consuming and confusing too. These types of systems can suggest a set of movies to users based on their interest, previously watched movies or sentiments of the customer. Factors like cast working in the film, genre of the movie, director and so on are also being considered while suggesting a movie. For designing this kind of recommendation system, we would build a model which recommends movies based on a combination of one or more attributes from a dataset and then convert a web app out of it, so that it becomes very user friendly and can be used very easily.

CHAPTER 01

1.1 Introduction

Recommendation systems have emerged as a powerful tool for filtering information by capturing user preferences and making suggestions based on those preferences. These technologies have spread across society and are now often employed in a variety of industries, including music, books, films, movies, cellphones, cuisine, locations, and other utilities. The system displays the best alternatives to the user after making suggestions based on their preferences and past behavior. Our lives would not be complete without films. Documentaries, horror films, cartoons, and action films are just a few of the many styles and formats they come in, ranging from amusing to informative. These movies may be divided into several genres, such as horror, comedy, thriller, emotional, animation, action, sci-fi, and so on. The year of release, the language, the director, and other characteristics can be used to distinguish them. In order to avoid the inconvenience of wasting time looking for our chosen movies, the movie recommendation system assists us in finding our favorite movies among many sources. So that it can provide us the exact and most pertinent movie suggestions based on our tastes, it is imperative that the movie recommendation system be extremely trustworthy and accurate.

The topic of recommendation systems has been extensively investigated over the past 20-25 years, and with quick improvements in this sector, users have learned to demand high-quality suggestions from various services. In fact, if a video streaming app fails to effectively forecast and propose films that match a user's preferences, the user is likely to abandon the service entirely. However, developing effective recommendation systems is a difficult task because each user has different preferences and tastes. Furthermore, an individual's choices can be influenced by a variety of factors, such as the time of year, their current mood, or the activity they are currently engaged in. The range of elements that might impact a user's preferences highlights the difficulty of recommendation systems. For example, the music one enjoys when exercising is very different from the music one prefers while cooking. Furthermore, the recommendation engine must balance exploration and exploitation. On the one hand, it should investigate many domains in order to learn more about the user's preferences; on the other hand, it should make the

best use of the information that is currently accessible.

The recommendation algorithm suggests the same genre of films to users with similar demographic characteristics and areas of interest. The method is seen to be overly straightforward because each person is unique and has a different set of interests. The idea behind this method is that when a user enters in a movie title in the search field, the system suggests films based on similar genres to the one being searched. This is filtering based on content. By building a matrix data type, we may group similar individuals together and utilize that information to suggest users using a technique called collaborative filtering. In the current fast-paced world when individuals are always pressed for time and have many things to complete within the finite 24 hours, recommendation systems have grown in significance. As a result, recommendation systems are essential for assisting users in selecting the appropriate options without having to engage their cognitive resources.

A recommendation system's main goal is to investigate stuff that could catch someone's attention. In order to achieve this, it takes into consideration a variety of factors and generates customized lists of interesting and useful material for every user. The algorithmic nature of recommendation systems enables them to thoroughly evaluate all available possibilities to provide a customized roster of goods that would appeal to a particular person. This result is based on their profile, search and browsing history, the viewing preferences of others who have their traits or demographics, and the chance that they will watch those films. By utilizing the available data, predictive modeling and heuristics enable this achievement.

The quickest and easiest way to do this is to propose the most popular items. But if we want to significantly enhance the user experience through tailored recommendations, we need specialized recommender systems. From a commercial standpoint, user engagement increases as users find more pertinent goods on the website. A frequent result of this is an increase in platform revenue. According to various reports, recommendations account for up to 35–40% of internet giants' total income.

1.2. Objective of Project

Our goal is to assist users in finding and viewing fresh films that are catered to their tastes and interests. Our system will employ machine learning algorithms to forecast the movies that the user is likely to appreciate by taking into consideration a number of variables, including the user's watching history, ratings, and movie qualities. We intend to solve some of the shortcomings of these individual approaches, such as the cold start problem and the sparsity of data, by combining content-based and collaborative filtering strategies. Additionally, our system will be built to be scalable, enabling it to manage sizable datasets and support a rising user base. The ultimate objective is to offer customers a smooth and personalized movie recommendation experience that increases their pleasure and engagement with the site. We want to reduce data overload and offer consumers more pertinent and helpful suggestions. There is a lot of room for more study and investigation in this area given the significance of movie recommendation systems. However, problems with a pure collaborative approach, such as poor recommendation quality and scalability difficulties, frequently impede the current state of these systems. We can build a more powerful and successful movie recommendation system that better serves consumers by combining the benefits of content-based and collaborative filtering strategies.

This is true; a recommendation system's efficacy mostly depends on its capacity to offer consumers accurate and pertinent recommendations. This is looking at a user's previous actions, preferences, and comments to spot trends and make suggestions for things that are probably going to be of interest. To constantly increase its accuracy, the system must also be able to modify and update its suggestions in response to fresh user interactions and input. A strong recommendation engine may boost customer happiness and engagement, which can eventually spur business expansion. It can be difficult and difficult to navigate the intricate and varied landscape of movie recommendation algorithms; this requires a sensitive and subtle touch. This is caused in large part by the enormous and continuously expanding amount of data that must be painstakingly sorted through and analyzed, as well as the wide diversity of user preferences that must be properly taken into account and assessed. In order to tackle this enormous and Herculean effort, a hybrid strategy that combines the complementary and effective approaches of both content-based filtering and collaborative filtering has been painstakingly developed.

With the letter methodology's analysis of the preferences and predilections of other users with similar and related interests, the former methodology's skillful identification and discernment of items with properties similar to those previously valued by the user is strengthened. Achieving the highest degree of quality and scalability in movie recommendation systems, however, still presents substantial hurdles and impediments that must be skilfully navigated and skillfully overcome, despite how smart and diverse this hybrid technique is.

One of the largest and most challenging issues that movie recommendation systems must address is the "cold start" problem. The other challenge is data sparsity. In the first case, there is no prior information or history about the new person or thing, making it challenging to provide specific advice in the absence of a reference point. The latter is connected to the lack of relevant data, which may make it more challenging for the system to identify and offer items to a user that they would find interesting. To address these challenges, a comprehensive and diverse approach based on cutting-edge techniques and advanced algorithms is necessary. Despite the "cold start" issue and data scarcity, these techniques allow movie recommendation systems to enhance and optimize their efficiency while still providing users with accurate and valuable recommendations..

Despite the many difficulties, a well-designed system for suggesting films has many benefits. A strong recommendation system may help viewers find new and fascinating films that they might not have otherwise seen, which makes for a more engaging and engaging viewing experience. Additionally, by providing individualized and pertinent recommendations, a movie recommendation system has the potential to increase user engagement and loyalty, resulting in a significant boost in revenue and profitability for movie streaming services. With the continued development of hybrid movie recommendation systems, there is a significant opportunity to enhance the quality, accuracy, and scalability of these systems, giving users a more customized and pleasurable movie-viewing experience while also enhancing the value of the movie streaming platforms. However, there are still issues that need to be addressed.

1.3. Technical Requirement

1.3.1 Hardware Requirements

- A PC with Windows/Linux OS
- Processor with 1.7-2.4GHz speed
- Minimum of 8gb RAM
- 2gb Graphic card

1.3.2 Software Specification

- Text Editor (VS-code/WebStorm)
- Anaconda distribution package (PyCharm Editor)
- Python libraries
- Streamlit for Web-app

1.4 Software Requirements

1.4.1 Anaconda distribution: A complete platform created for Python-based scientific computing is called the Anaconda distribution. Anaconda provides a user-friendly approach to manage Python packages and streamline deployment, and is specifically designed for applications in data science, machine learning, large-scale data processing, and predictive analytics. Anaconda is a well-liked option for people and businesses trying to simplify their data science operations due to its emphasis on usability. The package management mechanism of the Anaconda distribution is one of its main advantages. Users can avoid the hassles of handling dependencies and versions manually by depending on anaconda to handle package versions. Users may concentrate on their job rather than worrying about technological concerns thanks to this. Spyder, a potent Integrated Development Environment (IDE) included with Anaconda, is one such example. It has a number of capabilities, including debugging, syntax highlighting, and code completion. Jupyter Notebook, a web-based interactive computing environment that enables users to create and share documents that incorporate real-time code, mathematical equations, graphic representations,

and narrative prose, is also a component of Anaconda.

1.4.2 Python libraries: Data scientists and analysts may easily execute sophisticated computations and analyzes thanks to the use of Python modules. These libraries include a broad selection of tools and functions that may be used to handle, examine, and visualize data as well as create sophisticated models and algorithms. In data science and analytics, among the most often used Python libraries include SKlearn, Numpy, pandas, Matplotlib, and Flask framework.

SKlearn: SKlearn is a Python library that offers a range of classification, regression, and clustering algorithms, such as support vector machines, random forests, gradient boosting, k-means, and DBSCAN. It is designed to integrate seamlessly with NumPy and SciPy, two widely-used Python libraries for numerical and scientific computing.

NumPy and Pandas: NumPy is a versatile array-processing package that provides a high-performance, multi-dimensional array object and an array of tools for working with these arrays. As a result, it is the go-to package for scientific computing in Python.

Pandas is another popular Python library in data science that provides user-friendly structures and tools for data analysis. While NumPy offers objects for multi-dimensional arrays, Pandas provides an in-memory 2D table object called Dataframe.

Streamlit: Streamlit, on the other hand, is an open-source Python app framework that facilitates the creation of web applications for machine learning and data science quickly. It is compatible with a broad range of Python libraries, such as scikit-learn, Keras, PyTorch, SymPy (Latex), NumPy, pandas, Matplotlib, and more. With Streamlit, widgets are treated as variables, eliminating the need for callbacks, and data caching streamlines and accelerates computation pipelines.

Streamlit can track updates of the linked Git repository and application, making it an efficient tool for web app development.

1.5 Deliverables of Project

The project's deliverables cover every stage of the data science workflow, from early data exploration through final model deployment and report authoring. They are diverse and extensive. The benefits of having a good movie recommendation system are apparent, despite the many complexities involved in creating one. Not only can it improve users' overall movie-watching experiences by introducing them to fresh and interesting films they would not have otherwise explored, but it can also increase user engagement and loyalty by making suggestions that are specifically catered to each person's likes and inclinations. Additionally, this individualized method of movie recommendations may result in higher profits and revenue for movie streaming services. By utilizing hybrid recommendation systems, we can go one step further by improving the quality, accuracy, and scalability of the recommendation engine, which can give users an even more customized and pleasurable viewing experience while also generating value for movie streaming platforms. However, in order to make sure that the recommendation system works properly, it is imperative to address any unresolved problems. To do this, a thorough analysis of the data collection must be conducted in order to spot any patterns, trends, or linkages. Understanding the data and detecting any potential problems that would need to be fixed during data munging or modeling are both made possible by EDA. To prepare data for analysis, data munging entails cleaning and altering the data. This might include addressing missing data, coping with outliers, and encoding categorical variables, among other things. Data mining, which employs statistical and machine learning techniques to find patterns and correlations in the data, may be applied with the cleaned and processed data.

The process of choosing the optimal model involves comparing several algorithms based on assessment measures including accuracy, precision, and recall. Since evaluation involves evaluating the model's performance and identifying potential areas for improvement, it is a crucial step in the data science workflow.

This might entail adjusting hyperparameters, investigating different methods, or dealing with bias or overfitting concerns. The recommendation system is included into a web application or other software environment as the final step in model deployment. This

might entail developing an API to provide model access or incorporating the model into a web-based user experience. It is crucial to record all actions done, decisions made, and the justifications for those choices throughout the project. A project report that offers a comprehensive and in-depth analysis of the full project lifecycle must be included in every data science project. One of the most crucial components of the report is the data source, which must be thoroughly covered to provide context for the project. While the exploratory data analysis section should draw attention to any noteworthy trends or patterns in the data, the code implementation section should outline the numerous steps taken to clean, preprocess, and transform the data into a format suitable for modeling.

The report should describe the numerous methods utilized to identify the most relevant and instructional aspects for the recommendation system, which is another key aspect of the project. Information about the many models that were taken into consideration as well as the criteria used to choose the final model should be included in the section on model selection. A complete evaluation of the chosen model should be provided in the model performance section, taking into consideration factors like accuracy, precision, recall, and F1-score. Any obstacles or setbacks encountered during the project must be mentioned in the report, along with the solutions used to overcome them.

The report should also highlight any areas that require more scrutiny or study and suggest potential lines of inquiry or development for the future.

The project report must, in the end, include a comprehensive and in-depth evaluation of the whole data science project, including the data source, exploratory data analysis, coding, feature selection, model selection, and model performance.

CHAPTER 02

2.1 Literature Survey

Although many different recommendation systems with collaborative, content-based, or hybrid filtering methods have been developed over time, the efficiency of these systems ultimately depends on the big data and machine learning algorithms used. The accuracy and customization of suggestions sent to users may be greatly improved with the proper use of these algorithms, which can also lessen the possibility of data bias and increase system scalability.

Additionally, the incorporation of cutting-edge technologies, such as deep learning and natural language processing, has the potential to improve recommendation system performance even more while providing users with a more personalized and interesting experience.

Methods currently used to recommend films

1. In 1998, the first significant study on the emerging subject of recommender systems was released. A significant amount of work has now emerged, elucidating a number of factors that contribute to the dependability of these systems. Numerous factors have been discussed that improve the fidelity and trust in recommender systems. For instance, John O'Donovan and Barry Smyth created the concept of trust, which enables one to assess the proportion of accurate predictions a profile has made on an overall basis (profile-level trust) or specifically with regard to a certain item (item-level trust). Our thoroughly considered proposal aims to provide customers with informed film options depending on their tastes. The system must promptly deliver users with pertinent videos after scanning the database for video data in order to do this. Automatic product and service recommendations must be able to draw insights from prior user behaviors for recommendation systems to work well. These systems may be powered by a variety of algorithms, and it might be challenging to select the appropriate

algorithm among the many options available. The prediction error may be decreased by 22% by using the most efficient method, nevertheless.

2. Lops et al. released an innovative item-based collaborative filtering recommendation algorithm in 2011 that deftly circumvents the irksome issues with the rating system. The method subverts the conventional user-to-user collaborative filtering approach by using the rating distribution per item rather than for users. This reconfiguration eliminates the requirement for frequent model rebuilds by producing a model with a far more robust rating distribution. The use of this smart strategy allowed the system to achieve a great accuracy of 75%, demonstrating its dependability and efficiency.
3. Collaborative and content-based filtering are both common strategies in recommendation systems. However, it has been discovered that these methods have two critical flaws, namely the sparsity problem and the scalability problem. In order to solve these problems, Burkey (2007) proposed a hybrid recommendation system that combines the benefits of both methods. The system's proven capacity to achieve a notable accuracy of 70% has made it feasible to develop suggestions in a more effective and efficient manner.
4. Hongli Lin and his colleagues created content-boosted collaborative filtering (CBCF), a fresh method for recommendation systems, in 2008. The creation of this algorithm involves two basic phases. In the first stage, content-based filtering is utilized to raise the ratings of trainee instances that have previously been gathered. Next, collaborative filtering creates the final forecasts. The CBF-CF approach's drawbacks are combined with their benefits in the CBCF

algorithm, which also removes their weaknesses. There are several types of recommender systems that employ diverse approaches, some of which may be classified as content-based filtering systems (CBF-based systems). Astonishing accuracy rates of 75% are reportedly achieved by these systems.

5. Eugene Seo and Ho-Jin Choi published a method in 2009 for anticipating ratings and reviews using the k-means clustering algorithm. Their findings demonstrate that this strategy outperforms other current algorithms in terms of accuracy. The k-means clustering approach has actually been shown to have a remarkable accuracy of about 75%, making it a preferable option for predicting ratings and reviews.

6. Costin-Gabriel Chiru and his group came up with a mechanism for offering tailored movie recommendations. It uses information about individual users. This approach addresses the issue of broad suggestions, which arises from ignoring user-specific data. In order to provide suggestions based on data it has acquired about the user's psychological profile, watching history, and movie ratings from other websites, the system uses aggregate similarity calculations. A hybrid algorithm employed by the Movie Recommender system resulted in a remarkable accuracy of 79%, demonstrating the potential of personalized recommendation systems to increase consumer satisfaction.

We provide a complete analysis of the literature on recommendation systems that appeared in scholarly journals between 2001 and 2010 in this paper. Our objective is to get vital insights on the historical development and evolution of recommendation systems. The structure of the document is described as follows:

- The research methodology for the study is outlined.

- For papers on recommendation systems, categorization criteria are supplied.
- We review the literature on suggestions and offer our classification findings.
- The study's shortcomings are explored and conclusions are offered.

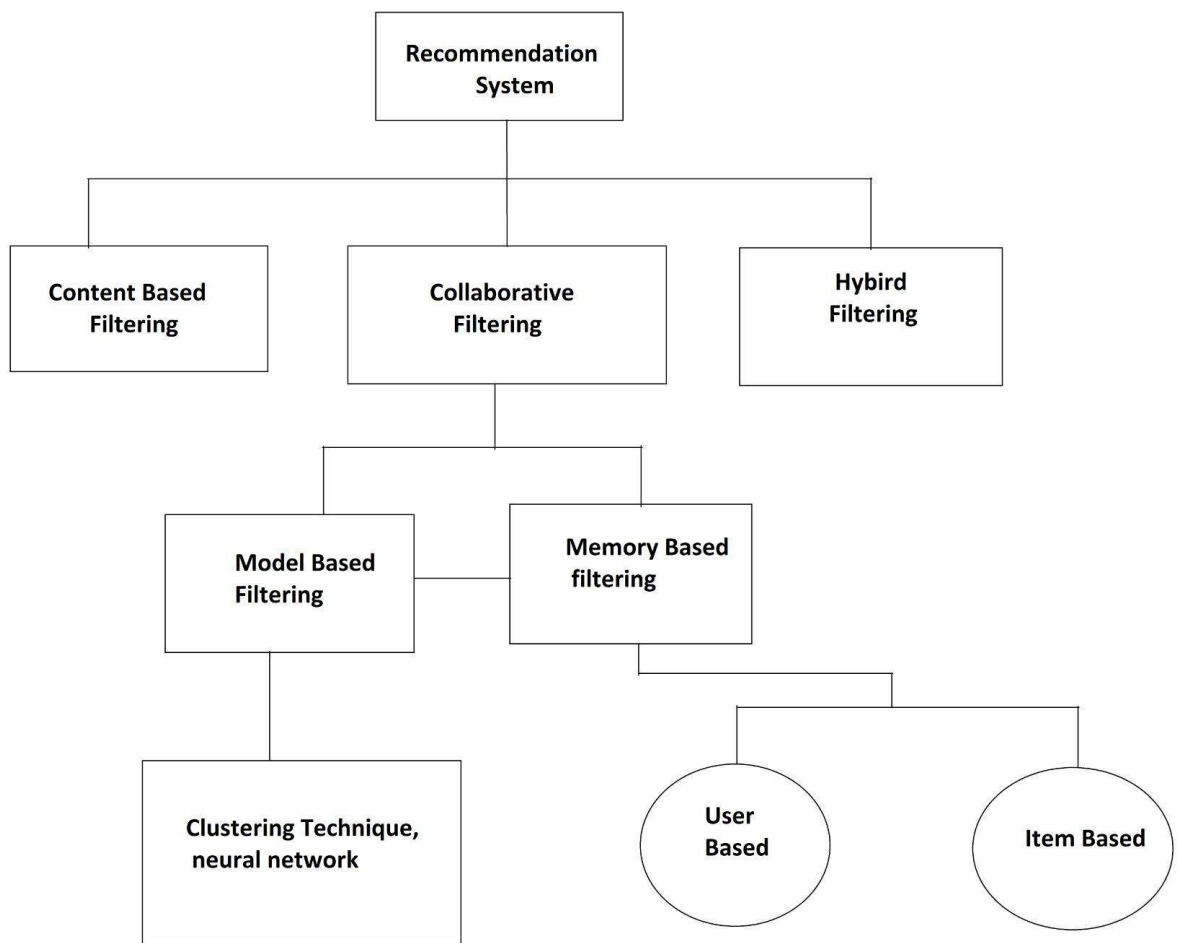


Figure 1: Recommendation techniques

2.2 Movie Recommendation System by K-Means AND KNN

A recommendation system is a computerized tool that gathers information about consumers' interests for various products, such as movies, either explicitly or implicitly. While explicit acquisition makes use of a user's past ratings or viewing history, implicit acquisition depends on the user's actions while watching films. Clustering is a supporting approach that groups a set of items so that they are more similar to one another than to those in other clusters.

K-Means Clustering and K-Nearest Neighbour are applied to the movie lens dataset to get the best-optimized output. The suggested strategy gathers data and produces fewer clusters than the current method, which results in an optimisation of the movie recommendation process. The current method scatters data and produces a significant number of clusters. The suggested recommender system makes predictions about a user's desire for a movie based on a variety of criteria, working on the premise that individuals have similar tastes or preferences, and that these tastes may affect one another. The procedure is therefore improved, and the root mean square error (RMSE) is decreased.

The discrepancy between the anticipated values and the actual values is measured by the RMSE (Root Mean Squared Error). A lower RMSE suggests that the model is doing better. There are various methods that may be used to lower RMSE. One of these methods is feature selection, where just the most important characteristics are chosen in order to cut down on noise and increase prediction accuracy. An alternative method is cross-validation, which assesses the model's performance on several subsets of the data in order to find and fix problems that might be contributing to a larger RMSE. In order to lessen overfitting and provide a lower RMSE, regularization techniques like L1 and L2 regularization can also be applied. Cleaning the data by eliminating outliers and pointless data can help increase the model's accuracy. Last but not least, fine-tuning the model's hyperparameters, such as the learning rate, number of hidden layers, and number of neurons, can assist to increase the model's accuracy and lead to a lower RMSE.

Popular and effective movie recommendation systems employ the K-Means and K-Nearest Neighbours (KNN) algorithms because they can precisely identify comparable

films based on their attributes and user preferences. Related films are linked up based on genre, director, cast, and rating using the clustering technique K-Means. The algorithm can then offer similar films to the user depending on the cluster to which their preferred film belongs once the films have been clustered. As a consequence, K-Means is effective in locating films with similar characteristics and genres, producing trustworthy choices.

KNN, in contrast, is a classification algorithm that assesses a movie's attributes and user preferences to identify its genre or category. The algorithm chooses the k movies that most closely resemble the user's favorite movie and proposes those movies. KNN is effective in finding films that match the user's preferred criteria and qualities, yielding accurate suggestions. K-Means and KNN combined can improve the accuracy of movie recommendations by grouping movies based on their qualities and then using KNN to recommend related movies inside the same cluster. By ensuring that suggestions are based on user preferences as well as comparable qualities, this technique ensures that recommendations are more accurate and specifically catered to the user.

Overall, K-Means and KNN algorithms are effective in locating related films based on their characteristics and user preferences, producing precise and customized recommendations.

2.2.1 Content-Based Movie Recommendation Systems

The commonality of movie properties is the foundation of content-based techniques. If a user uses this kind of recommendation system and watches one movie, similar movies are suggested to them. For instance, the algorithm will suggest films in the same genre or featuring the same actor, or both, to a user after they watch a comedy starring Adam Sandler. In light of this, movie qualities are the input for developing a content-based recommender system.

The benefit of content-based movie recommendation systems is that they may provide recommendations based on a person's tastes and interests, rather than on previous viewing behavior. Users who want to explore new genres or who are new users who might not have a large viewing history will find this to be especially helpful.

Extraction of pertinent features from the films is the first step in developing a model that can learn to recognise similarities across films based on those characteristics. This model is then used to create a content-based recommendation system. For this, one may utilize machine learning methods like k-nearest neighbors, decision trees, and neural networks.

The quality and quantity of the features taken from the movies determines how accurate content-based movie recommendation systems are. As a result, it's crucial to choose the attributes that are pertinent for suggesting related films with care. In conclusion, content-based movie recommendation systems are a useful resource for movie buffs and movie databases to offer viewers customized and pertinent recommendations. These systems are anticipated to improve in accuracy and usefulness over time as technology develops.

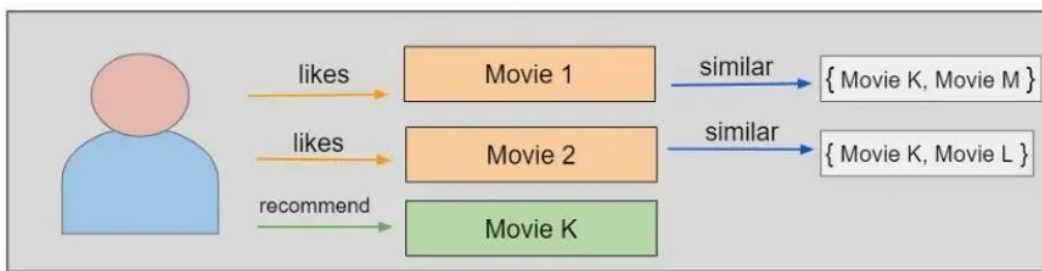


Figure 2: Content based recommendation system

2.2.2 Movie Recommendation System Using Collaborative Filtering:

Movie recommendation systems are becoming increasingly popular in the era of digital streaming, where viewers have access to a vast collection of movies and TV shows. One of the most popular and effective techniques for creating movie recommendation systems is collaborative filtering. Collaborative filtering is a machine learning technique that uses the behavior of users to recommend movies. Based on user evaluations of films, the system analyzes user preferences and suggests films that other users who share those tastes have enjoyed. Collaborative filtering is based on the idea that individuals with similar likes in films are more likely to have those same interests.

User-based and item-based collaborative filtering are the two primary varieties. User-based collaborative filtering compares users who have similar tastes in films and suggests titles that they have given high marks for. Item-based collaborative filtering recognises comparable films based on their characteristics and suggests films that are comparable to those that a user has already rated highly.

The steps listed below are commonly used to construct a collaborative filtering movie recommendation system:

1. Collecting data: Gathering a plethora of information on user evaluations and movie characteristics is the first step in creating a successful movie recommendation system. A powerful machine learning model that can generate highly customized and personalized suggestions is then trained using this data. The end product can provide movie fans with a special and pleasurable viewing experience that is catered to their particular interests and preferences. This cutting-edge recommendation system represents a significant advancement in the field of contemporary movie technology thanks to its capacity to seamlessly integrate user feedback and ratings.

2. Pre-processing data: Preprocessing the data after it has been gathered is the next step to make sure that it is in a format that the machine learning algorithm can use. Several techniques are used in this process, including data cleaning, handling missing values, and encoding categorical variables. In order to avoid biased and erroneous suggestions, the data must be cleaned of any mistakes, duplication, or inconsistencies. Imputation or deleting the rows with missing values are two methods for handling missing data. Categorical variables, such user preferences or movie genres, are encoded into numerical values that the machine learning system can quickly understand. The machine learning model is subsequently trained using the preprocessed data to produce precise and individualized movie suggestions.

3. Building the model: The construction of a complex machine learning model using cutting-edge collaborative filtering methods like user- or item-based filtering is necessary to create a highly effective recommendation system. The pre-processed data is carefully selected and utilized to train the model, which is then painstakingly optimized to generate very exact and accurate movie suggestions. The perfect customization of the final

suggestions to each individual user's specific movie interests and preferences depends on the careful selection of the most appropriate collaborative filtering approaches and the skilled training of the machine learning model.

4. Making recommendations: Once the model has been trained, it can be used to produce user-specific recommendations based on their preferences in movies and reviews. Numerous formats, such as a custom feed or a list of suggested films, are available for the delivery of the recommendations. The model makes use of sophisticated machine learning algorithms to offer each user more accurate and individualized recommendations, resulting in a more pleasurable and interesting movie-watching experience. The capacity to provide customized suggestions in a range of formats boosts customer loyalty and engagement, increasing revenue and profitability for movie streaming services.

Collaborative filtering has the benefit of being able to offer highly customized recommendations based on a user's particular interests and preferences. This may lead to an improved user experience and greater platform engagement. Collaborative filtering does have certain drawbacks, such as the "cold start" issue, which makes it challenging to provide correct suggestions to new users who haven't yet reviewed any films. Additionally, because collaborative filtering favors popular films over lesser-known ones more frequently, it can be biased.

Collaborative filtering is still one of the most well-liked and efficient methods for developing movie recommendation systems in spite of these drawbacks. By making personalized and pertinent suggestions to consumers, streaming services and movie databases may enhance engagement and client loyalty.

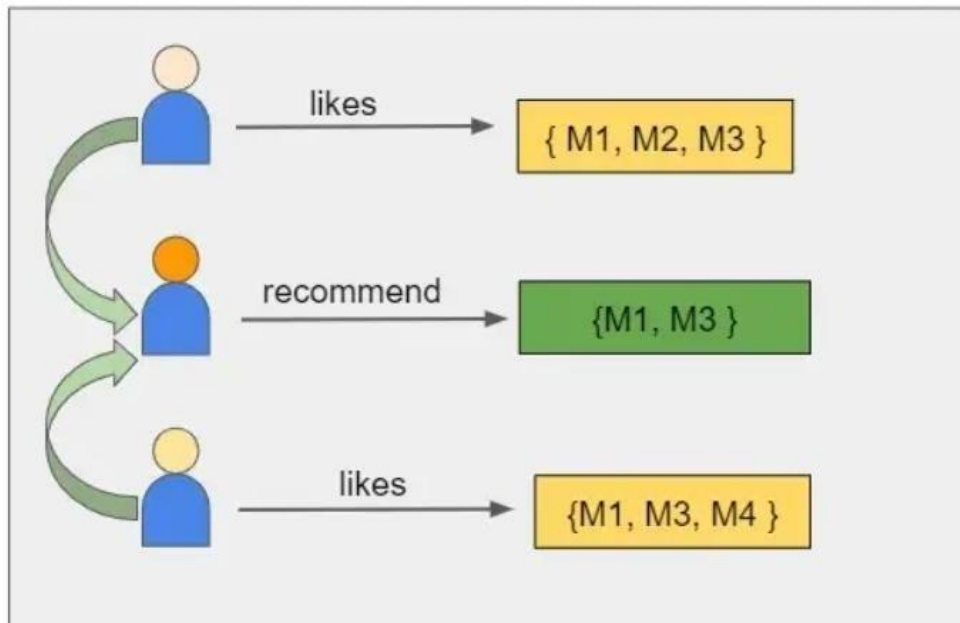


Figure 3: Collaborative Filtering

2.2.3 User-based filtering: A common method used in collaborative filtering is user-based filtering, which suggests films to users based on the preferences of other users who share their preferences. The idea behind user-based filtering is that people who have comparable movie ratings are probably similar in their interests and would thus likely like the same movies. This strategy is frequently employed in movie recommendation systems, which aim to suggest films that a user is likely to appreciate based on the ratings of other users who share their characteristics. A matrix of user ratings for each film in the dataset is initially created as part of the user-based filtering algorithm's operation. Each column denotes a movie, and each row denotes a user. The value in the matrix is null if a user hasn't given a film a rating. Following the creation of the matrix, each pair of users' similarity is determined using a similarity metric, such as cosine similarity or Pearson correlation. The similarity measure determines how similar two users' assessments are, and it gives them a number between 0 and 1.

The method determines the users who are most similar to the target user based on their prior movie ratings once the similarities between all pairs of users have been

computed. The algorithm then suggests films to the target user based on the films that other users who have similar interests have given high ratings. User-based filtering has the benefit of allowing for the creation of highly customised suggestions based on each user's particular interests and preferences. This strategy works especially well when recommending well-liked films that have a lot of ratings. The "cold start" issue that might arise with item-based filtering, where it is challenging to deliver correct suggestions for new films that have not yet been reviewed by any users, is similarly well-solved by user-based filtering.

User-based filtering has the drawback of being computationally demanding, especially when working with huge datasets. When the dataset contains millions of ratings, computing the similarity between each pair of users can take some time. To solve this problem, a variety of methods, including clustering and k-nearest neighbors (KNN), can be applied to lessen the computing complexity of the procedure.

The "popularity bias" issue, where popular films are recommended more frequently than lesser-known films, is another drawback of user-based filtering. This could happen because algorithms are more inclined to propose popular films because they frequently receive high ratings. The effect of well-known films can be lessened and more balanced suggestions can be made by using a variety of strategies, including weighing and normalization.

User-based filtering is an effective method for developing tailored movie recommendation systems overall. The algorithm may make precise and pertinent suggestions that boost consumer happiness and engagement by recognising comparable customers based on their prior movie ratings. However, it's crucial to be aware of the user-based filtering's limitations and to use the right strategies to deal with these problems.

2.2.4 Item-based filtering: Item-based filtering is a kind of collaborative filtering that suggests products (in this example, films) based on how closely they resemble previous products that a user has given high ratings. thing-based filtering is predicated on the idea that customers who give one thing a high rating are likely to give other, comparable goods a high rating as well. This method is frequently

employed in movie recommendation systems, which aim to suggest films that a user is likely to love in light of their prior viewing behavior.

The matrix of user ratings for each film in the dataset is first created in order for the item-based filtering method to function. Each column denotes a movie, and each row denotes a user. The value in the matrix is null if a user hasn't given a film a rating. Using a similarity measure, such as cosine similarity or Pearson correlation, the similarity between each pair of videos is determined once the matrix has been generated. A number between 0 and 1 is given based on how similar the ratings for two films are, according to the similarity measure. The method selects the films that are most comparable to those that a user has previously rated highly after calculating the similarities between all pairs of films. The system then recommends the most similar movies to the user based on their past viewing history.

Item-based filtering has the benefit of being able to offer highly customized suggestions based on a user's particular viewing preferences. This strategy works especially well when promoting obscure or lesser-known films that might not have a lot of ratings. The "cold start" issue that might arise with user-based filtering, where it is challenging to provide correct suggestions for new users who have not yet evaluated any films, is also avoided with item-based filtering.

Item-based filtering has the drawback of being computationally demanding, especially when working with huge datasets. When the dataset contains millions of ratings, computing the similarity between each pair of films can take some time. Numerous methods, including matrix factorization and neighborhood-based methods, can be used to address this problem and lessen the computational complexity of the algorithm.

The "sparsity problem," or a large number of missing values in the user-item matrix, is another drawback of item-based filtering. This can happen when people haven't given many movies ratings or when a lot of movies haven't received any ratings at all. It might be challenging to find comparable films in these situations and provide reliable suggestions.

Item-based filtering is a potent method for developing tailored movie recommendation systems overall. The algorithm may offer precise and pertinent

suggestions that boost engagement and customer happiness by finding films that are comparable to those that a user has previously rated highly. Item-based filtering has its limitations, too, so it's critical to be aware of them and adopt the right strategies to deal with them.

2.3 Feasibility Study

Different recommendation systems have been developed over time using a variety of techniques, including content-based, collaborative, and hybrid approaches. These recommendation systems had been developed using machine learning and big data technologies. A movie recommendation system called MOVREC was created and constructed by D.K. Yadav and colleagues using a collaborative filtering technique. In order to give suggestions to the user in collaborative filtering, we strive to group like people together by developing a matrix data type. Collaboration and content-based filtering are two conventional recommender systems that "Luis M Capos et al" created..

He discovered that each of the two recommender systems had drawbacks of its own, therefore he suggested a new system that combined the collaborative filtering technique with the Bayesian network. Additionally, "Harpreet Kaur et al." introduces a hybrid system that combines collaborative filtering and content-based filtering [3]. By using the chameleon, Utkarsh Gupta et al. joined the object-specific data or the consumer-specific records to create a cluster. He suggested this effective strategy, which mostly relies on hierarchical grouping. Voting methods are employed to forecast the film's rating. The clusters created in this suggested approach are better for related items, and the inaccuracy is also minimal. The authors "Urszula Kulewska et al." also offered two cluster computing techniques, using centroid-based sol and memory-based filtering as a basis for comparing the efficiency of the various approaches. And as a consequence, the generated suggestions' accuracy has significantly increased. Costin-Gabriel Chiru presented a system that leverages the user's profile information to make movie suggestions.

The project's main goal was to address the difficulty of making personalised suggestions, which frequently leads to the omission of user-specific data. The system gathers and examines information on user watching patterns, psychological profiles, and movie evaluations from numerous websites in order to address this problem. The sum of these

variables is computed to assess similarity. A hybrid model method that combines content-based filtering with collaborative filtering approaches has been created to accomplish this goal.

The difficulty level of each case for each learner may be predicted using a technique called content boosted collaborative filtering (CBCF), according to Hongli Lin et al. His technique consists of two stages: the content-based filtering that enhances the data on trainee case ratings already available, and the collaborative filtering that generates the final forecasts. This algorithm takes advantage of collaborative learning as well as content-based filtering.

Given that our goal is a mechanism for suggesting films. A collaborative filtering method, content-based filtering, or a combination of the two can be used to create a movie recommendation system. In our project, we created a hybrid strategy that combines collaborative filtering and content filtering. There are benefits and drawbacks to both strategies. Only these types of films will be suggested to the user in content-based filtering based on user ratings or likes. Benefits include ease of design and speedy computation. Cons: The model can only suggest items based on the user's current interests. To put it another way, the model has a limited capacity to build upon the consumers' already established interests. Comparing comparable users is how recommendations in collaborative filtering are made. Benefits: Because the embeddings are automatically learnt, domain expertise is not necessary. Users who use the model may find new interests. In a vacuum, the ML system might not be aware that the user is interested in a certain item, but the model might still suggest it since other users who share that interest might be. Dis-advantages: The dot product of the associated embeddings represents the prediction of the model for a particular (user, item) pair. As a result, if an item isn't observed during training, the system can't incorporate it and can't use it to query the model. The cold-start problem is another name for this phenomenon.

2.4 Problem Definition

The "Movie Recommendation System" project's main goal is to offer consumers personalized movie suggestions based on their own tastes and movie ratings. Utilizing the capabilities of machine learning algorithms, the objective is to provide relevant and

interesting content recommendations from a large library of films. The importance of this project rests in the fact that consumers may find it difficult to explore the vast and varied collection of films available in the contemporary film business. Users may see films that fall short of their expectations as a result of finding it difficult to find fresh films that match their preferences. The goal of the movie recommendation system is to address this issue by offering consumers accurate recommendations that will improve their movie-watching experience.

The recommendation system uses machine learning methods like content-based filtering and collaborative filtering to deliver precise recommendations by analyzing the user-provided movie ratings. The method takes into account the user's previous movie selections and ratings to suggest films that are relevant to their interests. Think about the examples of Housefull 1, Housefull 2, and Housefull 3 to demonstrate this. The algorithm can suggest Housefull 3 if a user gave Housefull 1 and Housefull 2 good ratings, thinking that the user will likely love it as well. The system may provide customers precise and pertinent recommendations by utilizing the capabilities of machine learning algorithms, improving their entire movie-watching experience.

In conclusion, the "Movie Recommendation System" project attempts to employ machine learning algorithms to provide customers with customized movie suggestions. The system may propose relevant and interesting content from a huge library of films by examining user ratings and preferences, improving the user's entire movie-watching experience.

2.5 Problem Analysis

We want an ideal dataset that has actors name, director name, genres, movie title, and movie rating, but it is very difficult to find the dataset with only these attributes. Also, if the attributes in the dataset are not enough, then the system will show the incorrect recommendation. For example, if you search for a movie "Liar Liar" and the system predicts movies with other genres like comedy, then you know that the system is not doing its job.

Users rely on movie recommendation systems to provide them individualized choices for films and TV series, making them a crucial component of the entertainment business. These methods do not, however, come without difficulties and issues. In this part, we'll

examine some of the major issues that movie recommendation systems could encounter.

1. **Cold Start Problem:** The cold start issue is one of the main issues with recommendation systems. When a new user joins the platform or a new film is uploaded to the database without any prior data to base suggestions on, this happens. The algorithm cannot make reliable suggestions without any data, which may result in a bad user experience.

2. **Sparsity Problem:** The sparsity issue with recommendation systems is another difficulty. This happens when there aren't enough reviews for a film, which makes it challenging to give reliable suggestions. In these circumstances, the system might suggest well-known films or unrelated films that may not be to the user's taste.

3. **Popularity Bias:** It's more common for people to recommend well-known films than unknown ones. This is due to the fact that more people have rated popular films, and the system is set up to provide suggestions based on previous user ratings. This may result in a bias towards popular selections in recommendations and keep people from learning about new films that could suit their likes.

4. **Scalability:** With more users and films in the database, scaling recommendation systems might become more difficult. Recommendation systems' algorithms can become progressively more complicated, making it challenging to analyze vast datasets effectively.

5. **Data Quality:** The accuracy of the suggestions depends heavily on the quality of the data utilized in the recommendation systems. Inaccurate or inappropriate suggestions can emerge from data quality problems such as missing or wrong data, outliers, and biases, which can have a negative impact on the user experience.

Movie recommendation systems can utilize a variety of methods, including collaborative filtering, content-based filtering, and hybrid approaches, to solve these issues. By combining the ratings of comparable users to provide recommendations, collaborative filtering algorithms can assist in resolving the cold start issue. The sparsity issue may be addressed using content-based filtering, which makes movie recommendations based on factors like genre, actors, and director. These methods can be combined in hybrid ways to deliver more precise and individualized suggestions.

In conclusion, although revolutionizing the way viewers find and watch films, movie

recommendation systems are not without significant difficulties. To create efficient recommendation systems that offer customers accurate and individualized recommendations, it is essential to comprehend these difficulties. The user experience and consumer engagement of movie recommendation systems may be improved by utilizing the right methodologies and addressing the main issues.

2.6 Solution

2.6.1 Accuracy: If the database for the recommender systems has few movies, the system will have higher accuracy. If the database is large, there tends to be a lower accuracy because the pool of information searched is too large. To counter the problem, the K-means algorithm reduces the computational time by restricting the computation to a certain number of iterations or selecting only the top-N number of movies for recommendations. However, if some of the movies have never been rated, they are likely to be biased in the searches. To increase the system accuracy in making the recommendations, some of the algorithms have employed sophisticated search criteria that will conduct a thorough search and match the product features to user and item characteristics.

In addition, two or more algorithms are combined to allow the perfect user and feature analysis and come up with the desired output. Some of the classification processes such as the PCA-SOM conduct the logarithmic computations offline so that they give recommendations easily when they are online. It reduces the computational time and increases the recommender system accuracy. In modern recommender systems, cold-start problems and the accuracy is solved by having a dialog box where the users can type in the features they need, and recommendations will be given according to what matches the search words.

2.6.2 Diversity: In the intricate and multifaceted realm of recommender systems, it has been observed that the number of movies contained within the database has a profound effect on the accuracy of the system. When the database contains only a meager selection of films, the accuracy tends to soar to greater heights. Conversely, a larger database has a tendency to impede accuracy, as the search pool becomes too extensive, and the system's computational power becomes taxed

beyond its limits.

However, to circumvent this issue, the K-means algorithm, a powerful tool in the arsenal of recommender systems, implements a cunning strategy. It effectively sets a limit on the number of iterations or curates a list of only the top-N movies, with the ultimate aim of drastically reducing computational time. While this approach is certainly beneficial, it does come with a caveat. If certain movies within the database have not been rated, the system can become rather biased, which can have a deleterious effect on the accuracy of the recommendations made.

To tackle this dilemma, certain algorithms within the system employ intricate and sophisticated search criteria, which facilitate a thorough and exhaustive search, taking into account a plethora of product features that match user and item characteristics. This enables the system to generate highly accurate recommendations that are more in tune with the user's preferences.

In order to achieve optimal output, it is not uncommon for two or more algorithms to be combined in order to facilitate comprehensive user and feature analysis. Among the many classification processes available, the PCA-SOM approach is a powerful tool that conducts logarithmic computations offline, effectively reducing computational time and enhancing system accuracy. In modern recommender systems, where accuracy and cold-start problems are critical issues, a dialog box has been incorporated, allowing users to input their desired features, which in turn generates recommendations based on matching search terms.

2.6.2 Diversity: Recommendation systems often fail to suggest new movies to users, which can result in excellent films going unnoticed due to a lack of user ratings. Additionally, some great movies may not receive any ratings, leaving the recommendation system uncertain about their quality for specific audiences. To address these challenges, recommender systems have introduced diversity to prioritize new or unrated movies, increasing the likelihood that users will notice and rate them. Based on user feedback, the system can then decide whether to continue recommending the movie or not. Additionally, by tracking the number of views, the system can categorize the movie based on its features or appeal to different user groups.

2.6.3 Scalability: While sparsity and diversity aim to increase the chances of movies with new features appearing in top searches, scalability aims to solve the problem of increased computational time and increase the performance of the recommender system. Scalability ensures that there is a balance obtained between accuracy and computational time. If it is necessary, some of the classification computations are performed beforehand so that by the time the user comes to select an item to watch, the system makes an almost immediate recommendation with high levels of efficiency.

2.6.4 Sparsity: The goals of sparsity and diversity are to promote the inclusion of movies with novel characteristics in the top search results. Meanwhile, scalability addresses the challenge of longer computational times and seeks to enhance the performance of the recommender system. Scalability aims to achieve a balance between accuracy and computational efficiency. To accomplish this, certain classification computations may be performed in advance, allowing the system to make almost instantaneous recommendations when the user selects an item to watch, thus maximizing efficiency.

2.6.5 Discussions: Movie recommendation systems operate in a complex landscape where a vast amount of data must be considered before making accurate recommendations. These systems face significant challenges due to the diverse range of user and context information. To address these challenges, movie recommendation systems have evolved over time, and current technologies require analyzing content, context, and user characteristics across various social media platforms to recommend appropriate movies to customers.

In the past, the MovieLens dataset was created when there was little to no development in the use of social media platforms where users share movie information to generate interest. However, today's technologies have advanced, and social media platforms play a crucial role in the functioning of movie recommendation systems. Companies have integrated analytics into their recommender system algorithms, allowing them to analyze user activity on social media platforms like Twitter, YouTube, and Meta to recommend the best movies.

By connecting to these platforms, they can also analyze the user's previous history to recommend appropriate movies, significantly reducing the cold-start problem. Context-based filtering is also gaining traction in movie recommendation systems, similar to product recommendations on e-commerce platforms. By integrating time stamps, recommendation systems can recommend the best movies in various contexts, such as children's lullaby movies at bedtime or educational movies during the day. Such personalized recommendations enhance the user experience and lead to greater user engagement.

There are also advances in the use of blockchain technology, which enhances user privacy but can impact the efficacy of collaborative filtering algorithms that depend on user information. In such cases, advanced methods such as context and content-based filtering can be used to maintain the accuracy of recommendations. Despite these challenges, movie recommendation systems continue to evolve, driven by emerging technologies, changing user needs and preferences, and an ever-growing pool of data.

2.7 Requirements

2.7.1 Functional Requirements

A recommendation system is a powerful model used to filter and present information to users based on their preferences and interests. A movie recommendation system is a particular type of recommendation system that helps users search for their preferred movies across various sources, reducing the time and effort required to find suitable movies. However, it is crucial that the movie recommendation system is reliable and accurate in its recommendations, providing users with suggestions that align with their preferences. The system should be able to analyze user data and ratings and provide recommendations that are the most relevant and matched according to the user's preferences. One of the main challenges in developing a movie recommendation system is handling the vast and diverse collection of movies available.

The system should be able to filter out irrelevant movies and present only those that align with the user's preferences. To achieve this, machine learning algorithms such as content-based filtering and collaborative filtering are commonly used. Content-based filtering involves analyzing the features of movies such as genre, director, and actors, and recommending movies that have similar features to those that the user has rated positively in the past. Collaborative filtering, on the other hand, involves analyzing the ratings and preferences of other users who have similar tastes and recommending movies that they have rated positively. In conclusion, a reliable and accurate movie recommendation system is crucial for enhancing the movie-watching experience of users. By leveraging machine learning algorithms such as content-based filtering and collaborative filtering, the system can provide personalized recommendations that align with the user's preferences and reduce the time and effort required to find suitable movies.

User	Recommendations
Student	Get movie recommendations related to adventures.
	Get the animated movie recommendations.
Adult	Get movie recommendations related to erotic.
	Get movie recommendations related to comedy.
Old	Get the religious movie recommendation.
	Get the movie recommendations related to biopics.

Figure 4: Categories table

2.7.1 Non-Functional Requirements

Performance of our website depends on the time taken by the recommendation engine to get the results with the movie details, cast details and the recommendations/suggestions of similar types of movies.

E-R Diagram / Data-Flow Diagram (DFD) ER Diagram

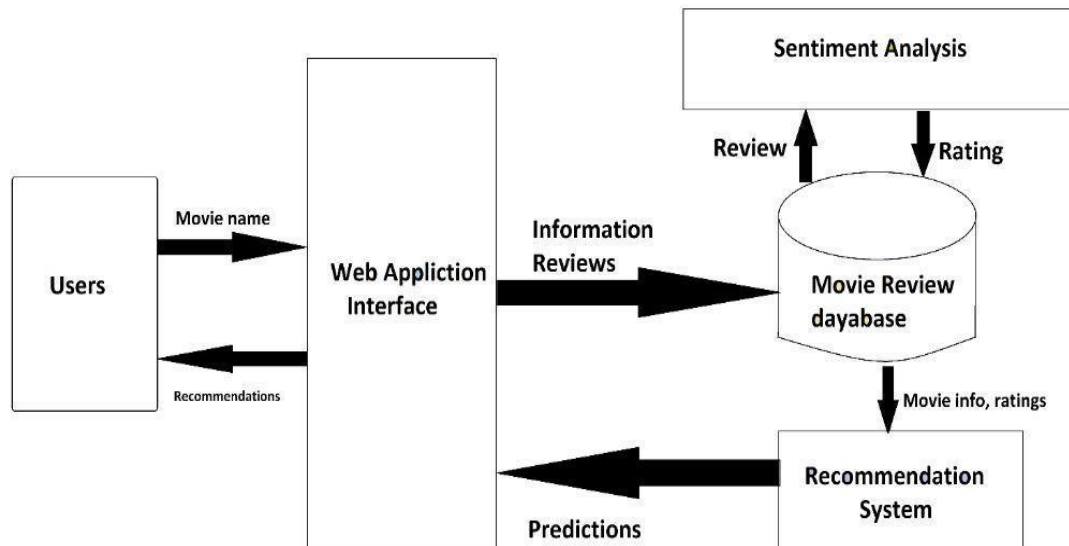


Figure 5: Data Flow Diagram

2.8 Objective of Major Project:

Movie recommendation systems are an integral part of websites and e-commerce applications, providing a mechanism to assist users in classifying their interests and preferences. These systems are designed to analyze vast amounts of user data and provide personalized recommendations based on their viewing history, preferences, and behavior. In this project, the first step is to load the necessary datasets required to build a movie recommendation model. These datasets, including movies.csv, rating.csv, and users.csv, are available on Kaggle.com and contain valuable information that can be leveraged to train and develop effective recommendation models. The project involves building two different models, namely content-based filtering and collaborative filtering.

The content-based filtering model is designed to analyze the features of movies such as genre, director, and actors and recommend movies that have similar features to those that the user has rated positively in the past. The collaborative filtering model, on the other hand, analyzes the ratings and preferences of other users who have similar tastes and recommends movies that they have rated positively. To provide the most accurate recommendations, both models are combined to produce a single final list of movies that are recommended to the particular user based on their viewing history and preferences. By leveraging machine learning algorithms and big data analytics, the project aims to provide a reliable and accurate movie recommendation system that enhances the movie-watching experience of users and reduces the time and effort required to find suitable movies.

CHAPTER 03

SYSTEM DEVELOPMENT

Implementation

The Proposed System Make Use Different Algorithms and Methods for the implementation of Hybrid Approach

3.1 Cosine Similarity:

In order to compare two vectors, such as user-item ratings or item-item attributes, the cosine similarity metric is frequently employed in recommendation systems. The dot product of the two vectors divided by the product of their magnitudes is used to determine the cosine similarity. Its values fall between -1 and 1, with -1 denoting completely different vectors and 1 denoting identical vectors.

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

Where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$ is the dot product of the two vectors.

3.2 Singular Value Decomposition (SVD):

Let A be an n*d matrix with singular vectors v_1, v_2, \dots, v_r and corresponding singular values $\sigma_1, \sigma_2, \dots, \sigma_r$.

Then $u_i = (1/\sigma_i) A v_i$, for $i = 1, 2, \dots, r$, are the left singular vectors and by Theorem 1.5, A can be decomposed into a sum of rank one matrices a

$$A = \sum_{i=1}^r \sigma_i u_i v_i^t$$

First, we provide a straightforward lemma that states two matrices A and B are equivalent if $Av = Bv$ for all v . The lemma claims that a matrix A can be conceptualized as a transformation that translates a vector onto another vector, Av , in the abstract.

3.3 Data Set Used in the Minor Project

In this project we have used various datasets which includes:

- a. IMDB 5000 Movies Datasets:
- b. The Movies Dataset:

It contains the following files:

- `movies_metadata.csv` : This is the main Movies Metadata file which contains
- `credits.csv` : It consists of information about the cast and crew for all our movie
- We are also extracting features of movies of various years from wikipedia.

3.4 Dataset Used in project:

- A. IMDB 5000 Movies Dataset:

```

Saving: tmdb_5000_movies.csv to tmdb_5000_movies.csv

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

movies.head(2)

movies.shape
(4803, 20)

```

ge	original_title	overview	popularity	production_companies	production_countries	release_date	revenue	runtime	spoken_languages	status	tagline	title	vote_average	vote_count
en	Avatar	In the 22nd century, a paraplegic Marine is d...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...}]	[{"iso_3166_1": "US", "name": "United States o..."}]	2009-12-10	2787965087	162.0	[{"iso_639_1": "en", "name": "English"}, {"iso_639_1": "en", "name": "English"}]	Released	Enter the World of Pandora.	Avatar	7.2	11800
en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"name": "Universal Studios", "id": 1}]]	[{"iso_3166_1": "US", "name": "United States o..."}]	2007-05-19	961000000	169.0	[{"iso_639_1": "en", "name": "English"}]	Released	At the end of the world, the adventure begins.	Pirates of the Caribbean: At World's End	6.9	4500

Figure 6: movie_metadata_1.csv

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	runtime	spoken_language
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}]	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id": 1464, "name": "culture clash"}, {"id": 1465, "name": "culture clash"}]	en	Avatar	In the 22nd century, a paraplegic Marine is d...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...}]	162.0	[{"iso_639_1": "en", "name": "English"}, {"iso_639_1": "en", "name": "English"}]
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 10, "name": "Family"}]	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "name": "na..."}]	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"name": "Universal Studios", "id": 1}]]	169.0	[{"iso_639_1": "en", "name": "English"}]
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 10, "name": "Family"}]	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name": "na..."}]	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"name": "United Artists", "id": 1}]]	148.0	[{"iso_639_1": "en", "name": "English"}, {"iso_639_1": "fr", "name": "Fran\u00e7ais"}]
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "name": "Fantasy"}, {"id": 12, "name": "Adventure"}]	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853, "name": "na..."}]	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[{"name": "Legendary Pictures", "id": 923}, {"name": "Warner Bros. Entertainment", "id": 1}]]	165.0	[{"iso_639_1": "en", "name": "English"}]
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 10, "name": "Family"}]	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id": 819, "name": "na..."}]	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[{"name": "Walt Disney Pictures", "id": 2}]]	132.0	[{"iso_639_1": "en", "name": "English"}]

Figure 7: movie_metadata_2.csv

B. The Movies Dataset:

It contains the following files:

movies_metadata.csv : This is the main Movies Metadata file which contains

```

Code + Markdown | ▶ Run All | Clear Outputs of All Cells | Outline ... Python 3.10.2 64-bit
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
movies.head()

```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "..."}, {"id": 726, "na...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	[{"id": 849, "name": "dc comics"}, {"id": 853...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 818, "name": "based on novel"}, {"id":...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813ea3", "de...

```

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

```

Figure 8: movie_metadata_3.csv

credits.csv: It consists of information about the cast and crew for all our movies. It is available in the form of a stringified JSON Object.

```

movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
movies.head()

```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "..."}, {"id": 726, "na...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	[{"id": 849, "name": "dc comics"}, {"id": 853...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 818, "name": "based on novel"}, {"id":...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813ea3", "de...

```

import ast
def convert(text):
    L = []
    for i in ast.literal_eval(text):

```

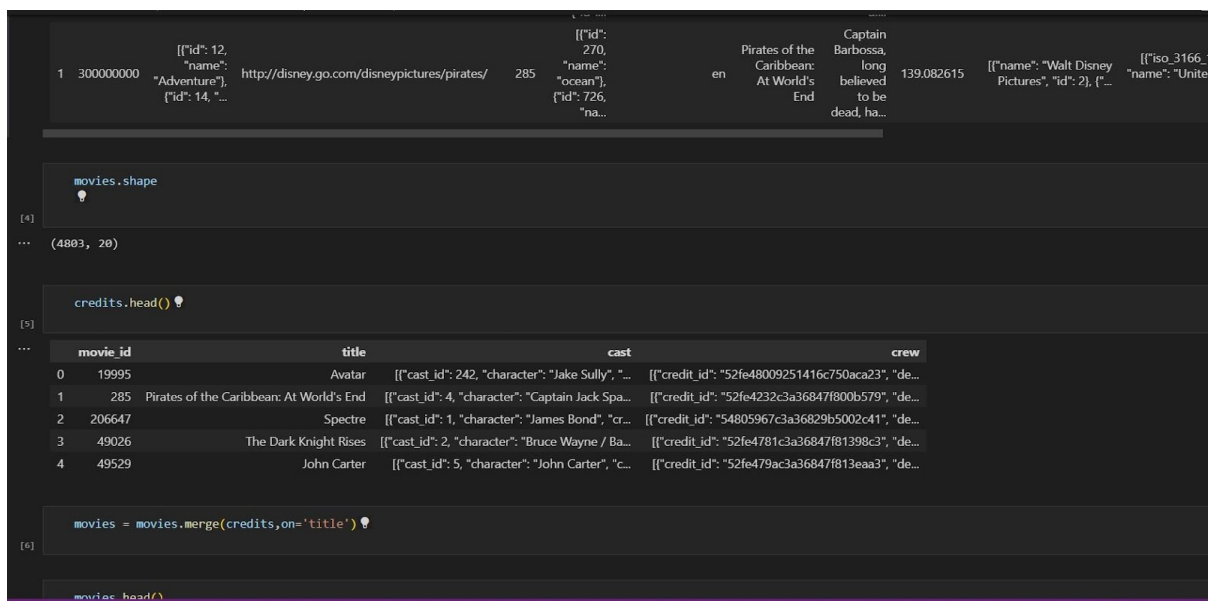
Figure 9: credits.csv

3.2 Data Set Features

Since, we are having various datasets with many different features, thus we have extracted various features from all datasets and have appended them and have created a final dataset. To do this , we have firstly selected a few features on which our recommendation will be done. These features include 'director_name', 'actor_1_name', 'actor_2_name', 'actor_3_name', 'genres', 'movie_title'.

```
data = data.loc[:,['director_name', 'actor_1_name', 'actor_2_name', 'actor_3_name','genres', 'movie_title']]
```

The above line of code extracted the required features from movie_metadat.csv (IMDB 5000 Movies Dataset). In this dataset we got data of movies till the year 2016. So, we will extract data of movies of next year from other datasets that we have.



The screenshot shows a Jupyter Notebook interface with the following content:

```
1 300000000 [{"id": 12, "name": "Adventure"}, {"id": 14, "..."} [{"id": 270, "name": "ocean"}, {"id": 726, "na..."} Pirates of the Caribbean: At World's End Captain Barbossa, long believed to be dead, ha... 139.082615 [{"name": "Walt Disney Pictures", "id": 2}, {"iso_3166- "name": "Unite..."}]
```

```
movies.shape
```

```
[4] ... (4883, 20)
```

```
credits.head()
```

```
[5] ...
```

movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "..."}, {"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...", "cr..."}, {"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr..."}, {"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...", "cr..."}, {"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c..."}, {"credit_id": "52fe479ac3a36847f813eaa3", "de...

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

```
[6]
```

```
movies.head()
```

Figure 10: movie_metadata.csv after selecting required features

After this, we are going to use the movies_metadata.csv(The Movies Dataset) to get movies data of the year 2017.

```

meta = pd.read_csv('datasets/movies_metadata.csv',low_memory=False)meta['year'] =
meta['release_date'].dt.year

ew_meta= meta.loc[meta.year == 2017,['genres','id','title','year']]new_meta
meta['release_date'].dt.eta= meta.loc[meta.year == 2012,['genres','id','title','year']]new_meta
meta['release_date'].dt.year

```

movie_id	title	overview	genres
19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]
285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]
206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]
49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]
49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]

Figure 11: Data of movies of year 2017 collected from movies_metadata

Since, the features like 'genres', 'cast' and 'crew' contains data as JSON objects, so we need to evaluate them using the "ast" module.

```
import ast
```

```

data['genres']=data['genres'].map(lambdax:ast.literal_eval(x))data['cast']
data['cast'].map(lambda x: ast.literal_eval(x)) data['crew']
=data['crew'].map(lambda x: ast.literal_eval(x))

```

Now for creating a 'genre_list', we define a function as follows: def make_genreList(x):gen = []

```
st = " " for i in x:
```

```
if i.get('name')=='Science Fiction':scifi = 'Sci-Fi' gen.append(scifi)
```

```
else:
```

```
gen.append(i.get('name'))if gen == []: returnnp.NaNelse: return(st.join(gen))
```

```
data['genres']=data['genres'].map(lambdax:ast.literal_eval(x))data['cast']=data['cast'].map(
```

```
lambda x: ast.literal_eval(x)) data['crew']=data['crew'].map(lambda x: ast.literal_eval(x))
```

Now for creating a 'genre_list', we define a function as follows: `def make_genresList(x):gen = []`

`st = " " for i in x:`

`if i.get('name')=='Science Fiction':scifi = 'Sci-Fi' gen.append(scifi)`

`else:`

`gen.append(i.get('name'))if gen== []: returnnp.NaNelse: return(st.join(gen))`

`data['genres_list']=data['genres'].map(lambdax: make_genresList(x))data['genres_list']`

Finally we get the 'genre_list' as follows:

```
0      Adventure Action Fantasy Comedy
1      Action Adventure Fantasy Sci-Fi
2      Action Adventure Fantasy Sci-Fi
3      Action Adventure Comedy Sci-Fi
4      Fantasy Action Adventure
      ...
526      Romance Comedy
527      Crime Comedy Action Family
528      Family Animation Romance Comedy
529      Crime Drama Thriller
530      NaN
Name: genres_list, Length: 531, dtype: object
```

Figure 12: 'genre_list' evaluated from the JSON object

Similarly, we evaluate a list of data for 'actor_1_name', 'actor_2_name', 'actor_3_name' and 'director_name'.

Then we finally extract all these features with the same feature names as in the previous csv file and then merge them to a new file.

Code Snippet:

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

import ast
ast.literal_eval('{{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}}')
def convert3(text):
    L = []
    counter = 0
    for i in ast.literal_eval(text): if counter < 3:
        L.append(i['name'])
    counter+=1
    return L
movies['cast'] = movies['cast'].apply(convert) movies.head()
movies['cast'] = movies['cast'].apply(lambda x:x[0:3]) def
fetch_director(text):
    L = []
    for i in ast.literal_eval(text): if i['job'] == 'Director':
        L.append(i['name'])
    return L

movies['crew'] = movies['crew'].apply(fetch_director) #movies['overview'] =
movies['overview'].apply(lambda x:x.split()) movies.sample(5)

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

Figure 13 : Code Snippet_1

```

movies['crew'] = movies['crew'].apply(fetch_director) #movies['overview'] =
movies['overview'].apply(lambda x:x.split()) movies.sample(5)
def
collapse(L):
    L1 = []
    for i in L:
        L1.append(i.replace("
",""))
    return L1
movies['cast'] = movies['cast'].apply(collapse)
movies['crew'] = movies['crew'].apply(collapse)
movies['genres'] = movies['genres'].apply(collapse)

movies['keywords']=movies['keywords'].apply(collapse) movies.head()
movies['overview'] = movies['overview'].apply(lambda x:x.split())
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] +
movies['crew']
new = movies.drop(columns=['overview','genres','keywords','cast','crew'])
#new.head()
new['tags'] = new['tags'].apply(lambda x: " ".join(x)) new.head()
from sklearn.feature_extraction.text
import CountVectorizer
cv=CountVectorizer(max_features=5000,stop_words='english')

vector=cv.fit_transform(new['tags']).toarray) vector.shape
from sklearn.metrics.pairwise
import cosine_similarity
similarity = cosine_similarity(vector)
similarity[new['title'] == 'The Lego Movie'].index[0] def recommend(movie):
    index = new[new['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])
    for i in distances[1:6]:
        print(new.iloc[i[0]].t
            title)

recommend('Batman Begins')

```

Figure 14 :Code Snippet_2

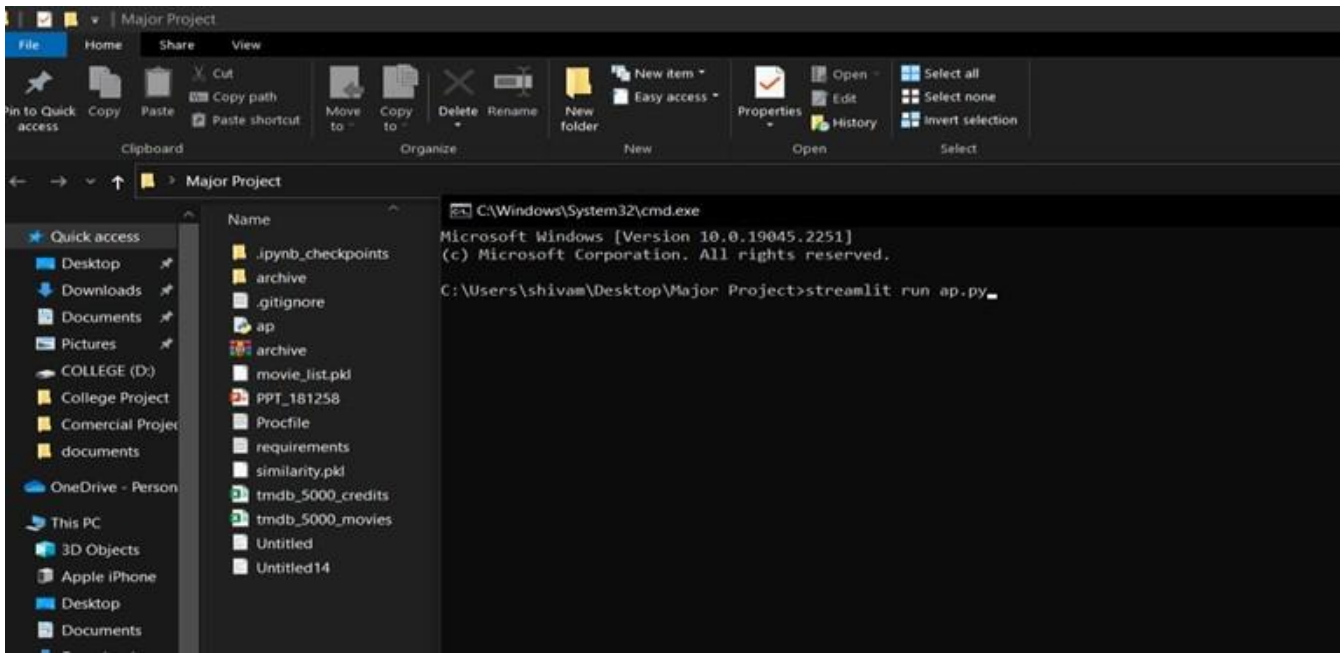


Figure 15: CMD_Project Screenshot

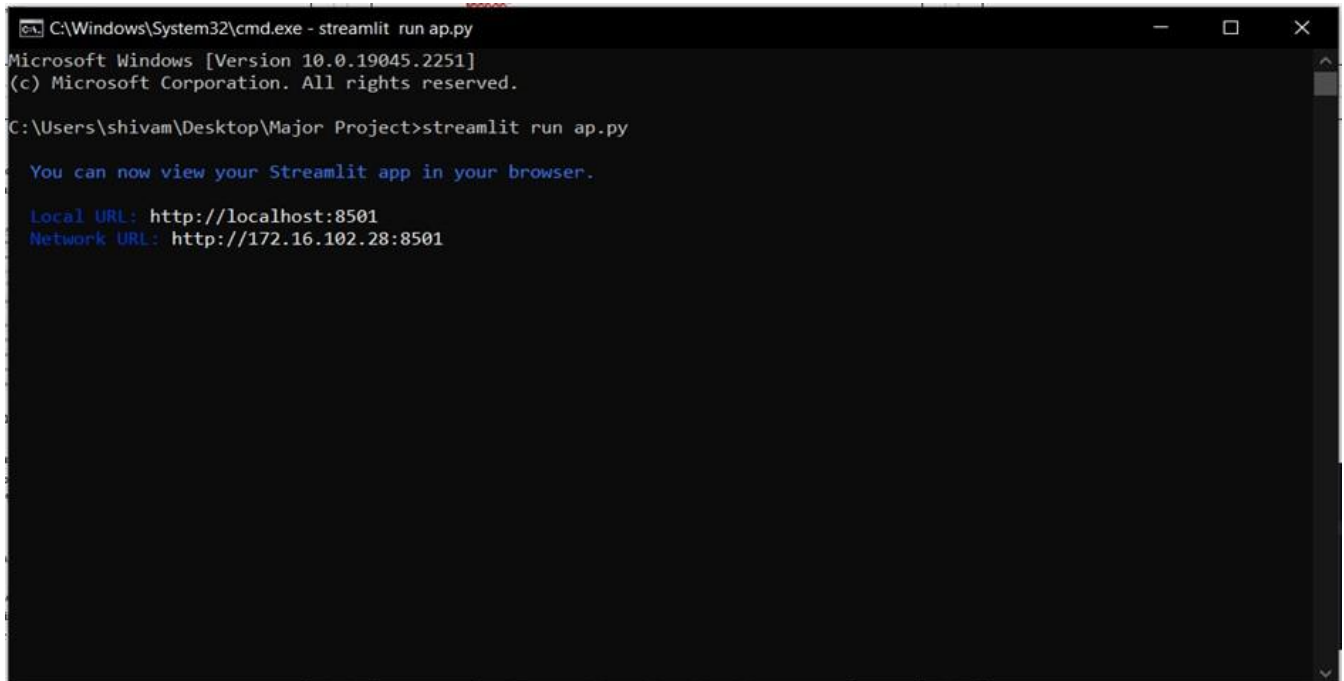


Figure 16:CMD_Local Host Link

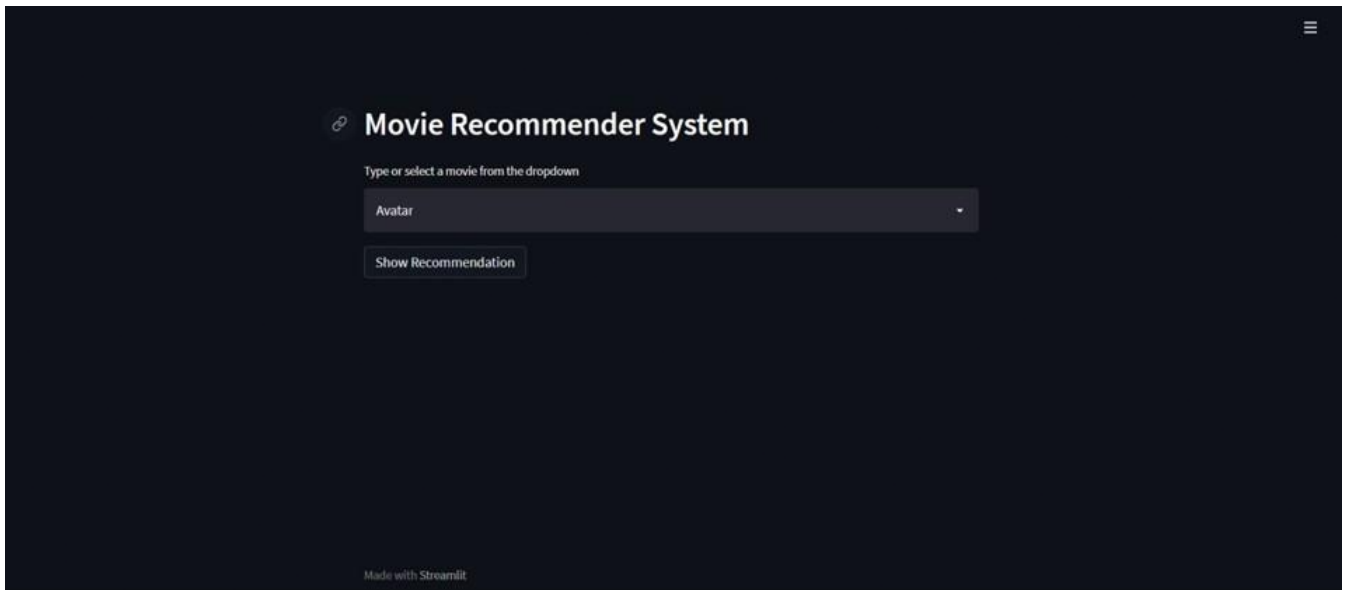


Figure 17: Home Page

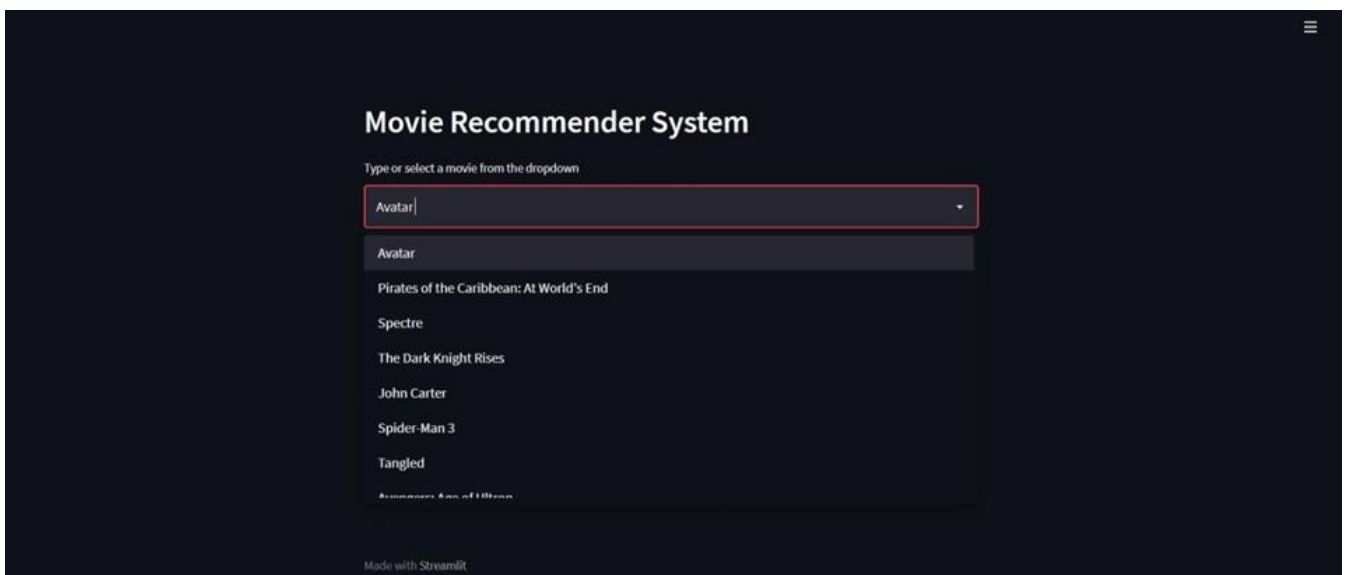


Figure 18: Auto Complete feature

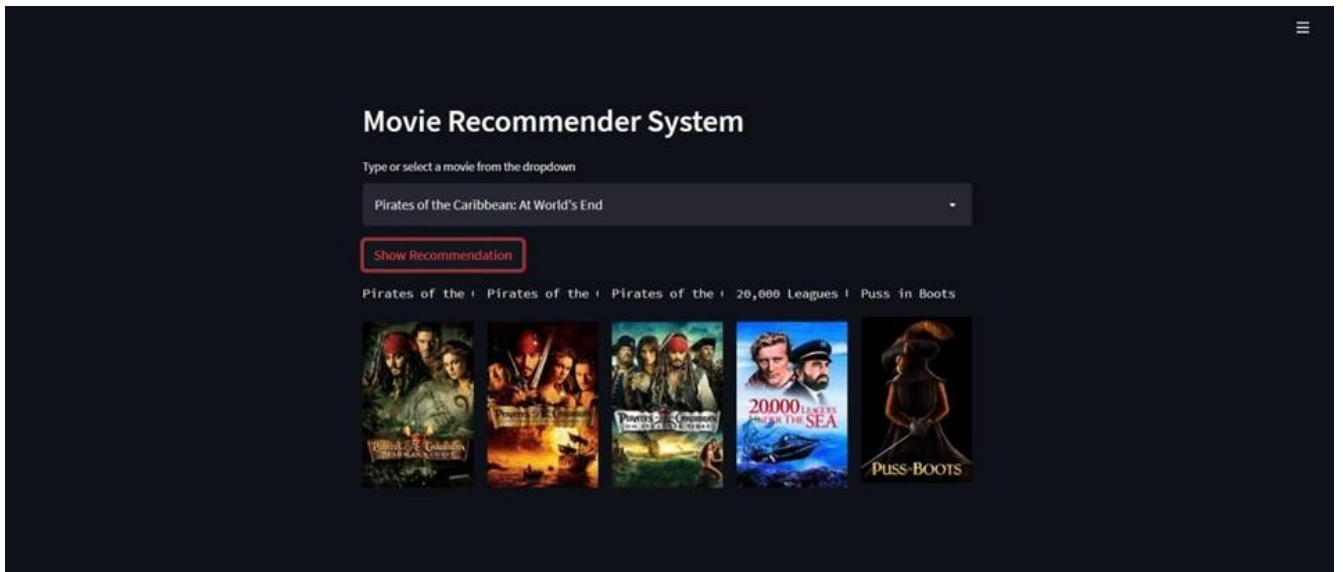


Figure 19: Movie Details

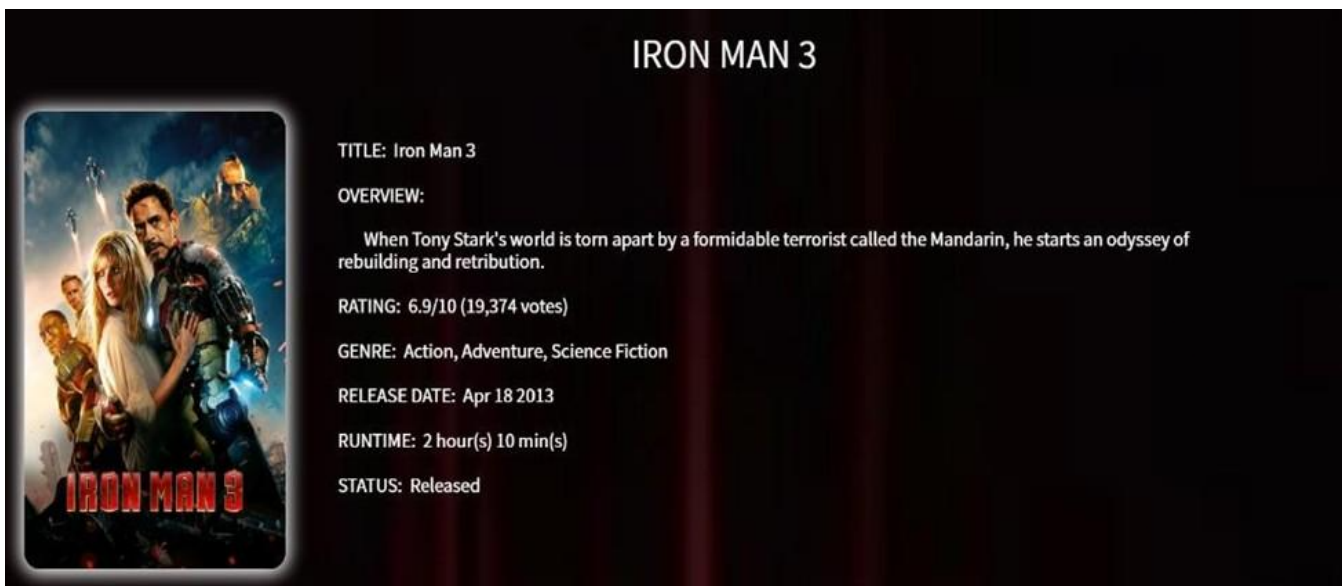


Figure 20: Particular Movie Detail

CHAPTER 04

PERFORMANCE ANALYSIS

4.1 Discussion on the Results Achieved

As per our aim and problem statement, our recommendation system is ready as a web page which is very user-friendly and easy to use. It takes a movie name as input and then it successfully matches it in our dataset to check for movie details and reviews. Movie details like cast and crew were taken from TMDB api and used IMDB website for getting reviews. Our recommendation engine successfully recommends movies on the basis of a similarity matrix generated from our dataset using cosine distance with features like genres and cast of the movies.

This system is viewed to users as a web page and the model works at the backend side. Movie name is passed in the backend and using the cosine similarity matrix, movies closest to the searched movie is recommended to the user.

4.2 Project Outcome

- Project Report: Completed
- Website: N/A
- Web-App-Yes
- Patent: N/A

CHAPTER 05

CONCLUSION

5.1 Conclusion

The primary objective of this project is to enhance the precision, quality, and scalability of movie recommendation systems by implementing a hybrid approach that combines content-based filtering and collaborative filtering. To achieve this, Singular Value Decomposition (SVD) is used as a classifier, and Cosine Similarity is the proposed methodology. The approach is tested on three distinct Movie datasets, and existing pure approaches and suggested hybrid approaches are compared. The results reveal that the suggested approach outperforms pure approaches in terms of accuracy, quality, and scalability of the movie recommendation system.

The proposed approach in this project not only outperforms the existing pure approaches in terms of accuracy, quality, and scalability of the movie recommendation system but also has a shorter computing time. To provide a comprehensive review of the movie recommender systems, 77 articles that strictly focus on this topic and 32 related articles on metaheuristics and recommender systems are included in the study. The limitations of the study are that the articles were not directly selected from Scopus and Web of Science databases. The study also identifies and discusses the challenges and issues related to movie recommender systems. As we dive into the vast realm of literature search, we resorted to utilizing a plethora of academic databases such as EBSCO Academic Search Premier, ScienceDirect, IEEE Library, ResearchGate, SpringerLink, and the ACM Portal. However, despite the extensive research performed, it was unearthed that a staggering 80% of the scrutinized papers were indexed in Scopus, and an additional 60% were readily accessible via the Web of Science database. It is worth noting that the contemporary information overload has unequivocally amplified the pertinence of recommendation systems. In this regard, we are unrelentingly seeking novel techniques to augment the accuracy representation of movies for the content-based recommender system. To begin with, we will adopt a content-based recommender algorithm that eliminates any instances of a cold start issue, which we have previously alluded to. In

Section 4.1 of our innovative recommender system, we have an extensive list of features that outshine the rest in terms of diversity and precision. One factor contributing to this is the involvement of various research teams within our organization, each with their unique insights. Furthermore, we presented the industry-standard cosine similarity measure as a tool for enhancing the quality of our recommendations. To further improve the representation of the movies, we devised TF-IDF-DC, a cutting-edge technique. This thesis proposes a content-based recommendation system for VionLabs' movie platform, utilizing the multifaceted and unique aspects of the films to construct the features that make up the system.

Our research has yielded a critical discovery- TF-IDF-DC provides a more comprehensive depiction of movies. To further enhance the accuracy of our content-based recommender system, we devised a novel approach for assigning weights to the unique features extracted from the films, which were collected from a diverse set of research teams within the organization. In this master's thesis, we present our findings on a content-based recommender system developed for VionLabs' movie website. The cosine similarity, a widely adopted measure in the industry, is also discussed.

The assessment of our new strategy using k-NN and several metrics was conducted at the end of the project. The results demonstrate that our approach provides significant improvements, reinforcing the importance of incorporating diverse and distinctive movie features and implementing a robust weighting methodology.

5.1 Application of the Minor Project

Recommendation systems are the wide areas which are widely used in almost all the sectors. The main purpose of these recommendation systems is to save the time of people. So this is the reason that it plays an important role in different areas. Basically, the recommendation systems are used in real life programs such as social media, E-commerce, entertainment and so on. In the area of entertainment these recommendation systems are used in listening to music or watching films.

Basically, movie recommendation systems suggest movies to watch based on the user's interest and saves the user's time as he/she doesn't need to go through a long process of searching a movie to watch from a large list or set of movies. For example if the person

decides to indulge in a movie or series from an OTT platform then he/she ends up by watching more than what he/she has in their mind. So have you ever wondered how they could do such a thing?

The reason is that almost all the OTT platforms rely on their movie recommendation systems. These OTT platforms identify the choice and priority of a user and show the best recommendation which has a higher chance to get selected by them.

5.2 Limitation of the Movie Recommendation System

- a. The study at hand has some limitations which are outlined below:
- b. The study's findings are limited by the fact that the search conducted in digital libraries such as IEEE, Springer, and ProQuest may not be comprehensive.
- c. A consistent and standardized methodology for conducting searches across all digital libraries is lacking.
- d. This study is limited to findings that are based solely on English language publications as non-English publications were not included in the analysis.
- e. The studies were chosen by considering their title, abstract, and keywords. However, in some cases, the articles were also identified based on their full text.
- f. The study may face the challenge of author bias for paper selections, which means that the personal preferences, beliefs, or values of the researchers may influence their choice of papers to include in the study.

Cold Start Problem: The main problem in the recommendation system is the cold start problem. Basically, the cold start problem occurs in the recommendation system due to the lack of information on users or other attributes. It is a well known issue in recommendation systems as there is little information about the user, which results in impotence to draw correct recommendation to users. The cold start problem is further divided into three types: new item problem, new user problem and new system problem. If this problem occurs, then it is very difficult to provide recommendations as in the case of a new user, the available information is very less

and also for a new item, ratings are not available sometimes and therefore collaborative filtering cannot make useful recommendations in case of new user as well as in case of new item. However, by using content based recommendation methods, we can get recommendations as they do not depend on any information of other users to recommend the item.

Data Sparsity: The sparsity of the user or rating matrix poses a significant challenge in developing an effective movie recommendation system. Given that most users do not rate movies, it can be quite challenging to identify users who have rated the same movie with similar ratings. Consequently, the system struggles to identify users with similar movie preferences, particularly when there is limited user data available. This challenge is further compounded by the fact that making accurate movie recommendations becomes increasingly difficult as the amount of user information decreases.

Scalability: As the number of objects increases, the scalability of the system becomes a major concern. If the system has to deal with hundreds of millions of users and millions of items, it becomes impractical to respond to user requests in a timely manner due to the large amount of data. As a result, it may not be feasible for the system to make movie recommendations in such a scenario.

Unit testing, the initial level of software testing, is a critical process where developers ensure that individual components of the software are working as expected at the code level. In a test-driven environment, tests are written and executed by developers prior to the software or feature being passed on to the testing team. While unit testing can be done manually, automating the process can speed up delivery cycles and increase test coverage. Debugging also becomes easier as it is easier to find issues earlier than later in the testing process. With Test Left, advanced testers and developers can shift left and utilize the fastest test automation tool embedded in any IDE.

Once each unit is thoroughly tested, it is integrated with other units to create modules or components designed to perform specific tasks or activities. Integration testing ensures that segments of an application work seamlessly together by testing these groups of components as a whole. Typically, these tests are framed by user scenarios, such as logging into an application or opening files. These tests are usually a combination of

automated functional and manual tests that can be done by developers or independent testers.

System testing, on the other hand, is a black box testing method that evaluates the entire system as a whole to ensure it meets the specified requirements. It tests the functionality of the software end-to-end and is usually conducted by a separate testing team than the development team before the product is released into production.

5.3 Contributions:

- **Shivam verma:** Research and Development for the Database, web-app Development.
- **Gautmi Singh:** Research and development of approach taken for Prediction, Calculating Accuracy, Data Analysis, and visualization of data.

5.4 Future Work

As we look to the future, advancements in Machine Learning and high-performance computing continue to drive the development of new techniques in the field. There are several aspects that we will consider for our future work. Firstly, we aim to introduce more precise features of the film into our content-based filtering approach. Instead of relying solely on ratings, we plan to extract object features such as subtitles and color to obtain more accurate data about the movie.

While the proposed approach already takes into account the genres of movies, in the future we also aim to consider the age of users as it can significantly impact movie preferences. For instance, during childhood, animated movies are often preferred over other genres. Additionally, we will work on improving the memory requirements of our approach to make it more efficient.

So far, our proposed approach has been implemented on various movie datasets, but we plan to expand our research by implementing it on larger datasets such as Film Affinity and Netflix to evaluate its performance. Overall, we will continue to explore new ways to enhance our content-based filtering approach to provide even more precise movie recommendations.

As we delve into the realm of recommendation systems, we must recognize the importance of user data. In order to elevate the accuracy of our recommender system, we must gather more information from the users regarding the movies they dislike. Incorporating this data into our system and generating new scores to be added to our previous results can enhance the overall performance of our system.

However, the scope of our approach must not be limited solely to movie genre. The age of the user is a critical factor to be considered. As preferences for movies are known to shift with age, it is necessary to factor in this element. For instance, children are more inclined towards animated movies compared to other genres. It is imperative that we focus on improving the memory requirements of our proposed approach. While we have only implemented our approach using movie datasets, it is feasible to expand our work to incorporate datasets from Netflix and the Film Affinity, thereby providing more comprehensive results.

- Collaborative filtering recommendations will be the way forward after gathering sufficient user data. As we previously discussed in Section 2.2, collaborative filtering operates by analyzing the social information of users, and this will be explored in more detail in future research. This technique leverages the preferences and habits of like-minded users to generate recommendations for a particular user. By analyzing the shared patterns of behaviors and choices among a group of users, the system can predict the interests and preferences of a user based on the behavior of others with similar tastes. This approach has been successfully employed in many online platforms to provide personalized recommendations to users. Therefore, it is essential to integrate this method into our recommender system to enhance its accuracy and efficiency.
- As we delve into the future of recommendation systems, it is essential to introduce more precise and proper features of the movies. Currently, the typical collaborative filtering recommendations rely on the rating data, but we can enhance the accuracy by incorporating object features of the movies. For instance, we can extract more specific features such as the subtitles of the movie and the colors used in the film. By doing so, we can obtain more

precise and detailed information about the movie, which will help in generating more accurate recommendations.

- For recommender systems, user data plays a crucial role in improving the accuracy of recommendations. To further enhance the performance of our recommender system, we plan to gather more comprehensive user data in the future, including information on users' movie preferences and their dislike movie lists. By adding the user dislike movie list to our recommender system, we can generate more precise and personalized recommendations. Specifically, we will use this information to adjust our recommendation algorithm and generate scores that reflect both the user's likes and dislikes. This way, we can improve the accuracy and effectiveness of our recommender system and provide users with more relevant movie recommendations.
- To improve the performance of our recommender system, we plan to incorporate machine learning techniques in our future research. One approach is to introduce dynamic parameters that can adjust the weight of each feature automatically based on the feedback from the users. This will allow us to find the most suitable weights for each feature and improve the accuracy of our recommendations. With the help of machine learning, we can also identify new patterns and trends in user behavior that can be used to further optimize our system. By constantly refining our algorithms, we can provide more personalized and relevant recommendations to our users.
- In order to increase the accessibility and efficiency of the recommender system, we plan to make it an internal service that can be easily accessed by developers through APIs. This will allow for easier integration of the recommender system into other applications or platforms. Additionally, we plan to implement a sorting feature for movie lists on the website based on the recommendations generated by the system. This will make it easier for users to discover new movies that are likely to be of interest to them.

REFERENCE:

1. H. Shivhare, A. Gupta, and S. Sharma, "Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure," in IEEE International Conference on Computer, Communication and Control, 2015.
2. M. Kumar, D. K. Yadav, A. Singh, and V. Kr. Gupta, "A Movie Recommender System: MOVREC," International Journal of Computer Applications (0975 – 8887), vol. 124, no. 3, 2015.
3. R. Kim, Y. J. Kwak, H. Mo, M. Kim, S. Rho, K. L. Man, and W. K. Chong, "Trustworthy Movie Recommender System with Correct Assessment and Emotion Evaluation," in Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. II, 2015.
4. Z. Wang, X. Yu, N. Feng, and Z. Wang, "An Improved Collaborative Movie Recommendation System using Computational Intelligence," Journal of Visual Languages & Computing, vol. 25, no. 6, pp. 793-799, 2014.
5. D. Roy and A. Kundu, "Design of Movie Recommendation System by Means of Collaborative Filtering," International Journal of Emerging Technology and Advanced Engineering, vol. 3, no. 4, 2013.
6. B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Improving the Effectiveness of Collaborative Filtering on Anonymous Web Usage Data," in Workshop on Intelligent Techniques for Web Personalization (ITWP01), pp. 53-61, 2001.
7. S. Nadi, M. H. Saraee, and A. Bagheri, "A Hybrid Recommender System for Dynamic Web Users," International Journal Multimedia and Image Processing, vol. 1, no. 1, pp. 3-8, 2011.
8. S. R. S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, and B. Venkatesh, "Content-based movie recommendation system using genre correlation," in Smart Intelligent Computing and Applications, pp. 391-397, 2018.

9. A. Kashyap, B. Sunita, S. Srivastava, P. H. Aishwarya, and A. J. Shah, "A movie recommender system: MOVREC using machine learning techniques," *International Journal of Engineering Science and Computing*, vol. 10, no. 6, 2020.
10. S. Agrawal and P. Jain, "An improved approach for movie recommendation system," in *International Conference on I-SMAC*, pp. 336-342, 2017.
11. M. M. Reddy, R. S. Kanmani, and B. Surendiran, "Analysis of Movie Recommendation Systems; with and without considering the low rated movies," in *International Conference on Emerging Trends in Information Technology and Engineering*, pp. 1-4, 2020.
12. A. Håkansson, "Portal of research methods and methodologies for research projects and degree projects," in *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013*, Las Vegas, Nevada, USA, 2013, pp. 67–73.
13. T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *IJCAI*, 1999, pp.
14. Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, Christchurch, New Zealand, pages 49–56, 2008.
15. Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.
16. Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in folksonomies. In *Knowledge Discovery in Databases: PKDD 2007*, pages 506–514. Springer, 2007.
17. Joseph A Konstan. Introduction to recommender systems: Algorithms and evaluation. *ACM Transactions on Information Systems (TOIS)*, 22(1):1–4, 2004.
18. Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages

426–434. ACM, 2008.

19. Daniel Z Lieberman, Suen H Massey, Vilmaris Quiñones Cardona, and Kenneth P Williams. Predicting content preference: Applying lessons learned from the commercial web to therapeutic software. *Cyberpsychology*, 2(2), 2008.
20. Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.
21. Xu Hailing, Wu xiao, Li Xiaodong, and Yan Baoping. Comparison study of internet recommendation systems. *Journal of Software*, 20(2):350–362, 2009.

APPENDICES

1. Mathematical equation for Cosine similarity
2. Mathematical equations for Singular Value Decomposition
3. Model
4. Code snippet for implementation of the machine learning model
5. Code snippet for evaluation parameters