

A Project Report
on
**A HYBRID APPROACH TOWARDS A
MOVIE RECOMMENDATION SYSTEM**

*Submitted in partial fulfillment of the
requirement for the award of the degree of*

Bachelor of Technology
in
Computer Science and Engineering



**Under The Supervision of
Dr. Shrddha Sagar
Professor**

Submitted By

**Prakhar Anand
20SCSE1010198**

**Sajid Azam
20SCSE1010035**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
May, 2023**



**SCHOOL OF COMPUTING SCIENCE AND
ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA**

**CANDIDATE'S
DECLARATION**

We hereby certify that the work which is being presented in the thesis/ project/ dissertation, entitled **“A HYBRID APPROACH TOWARDS A MOVIE RECOMMENDATION SYSTEM”** in partial fulfillment of the requirements for the award of the B. Tech. submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of January, 2023 to May, 2023, under the supervision of Dr. Shreddha Sagar, Professor, School of Computing Science and Engineering, Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by us for the award of any other degree of this or any other places.

Prakhar Anand (20SCSE1010198)
Sajid Azam (20SCSE1010035)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Shreddha Sagar
Professor, SCSE

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Prakhar Anand (20SCSE1010198) and Sajid Azam (20SCSE1010035), has been held on _____ and their work is recommended for the award of B. Tech.

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Program Chair

Signature of Dean

Date: May, 2023

Place: Greater Noida

ABSTRACT

Online streaming media platforms are booming these days. OTT (Over-The-Top) platforms like Netflix, Amazon Prime, Hotstar, etc., provide online services that cater specifically to their users and provide them with some much-needed entertainment, which helps them to relax in this extremely busy world and spend some quality time with their friends and family. But when it's time for a short break just to freshen up the mind, everyone prefers choosing from the options that are just a click away, what could be better than a user being recommended their favourite songs to play, movies to watch during a break. This is where the highly advanced and convenient recommendation systems come into play. Recommendation systems solve this problem by analyzing the large volume of dynamically generated data to provide personalized product suggestions and services to the users. With the day-to-day increasing popularity and demand of such services, and with increasing competition in this massive service-oriented sector, a highly functional and effective recommendation system could provide a significant marketing edge to a company over its competitors, as wide range of products, services and their substantial amount of information are available on the service provider company's website and as a result, the users struggle to find relevant information matching their preferences. Thus, we have tried to design a framework for a movie recommendation system that will recommend movies to the user based on their preferences and user content history. First, we have taken a dataset provided by TDMB movies. Then, we have used and tested various classical machine learning models like Content-Based Filtering and Collaborative Filtering to recommend the best movies based on user ratings, user watch history, etc., that are more likely to be watched by the customer. We have also combined these models to create a Hybrid model which provides better recommendations. This could help in improving the revenue earned by the media platforms, which in turn is the main purpose of any recommendation system.

Keywords—OTT, Recommendation System, Content-based Filtering, Collaborative Filtering, Hybrid Filtering, Machine Learning

List of Tables

Table No.	Table Caption	Page No.
1	Table I: Study Done in the Field by Various Authors in the Referenced Literature	7-12
2	Table II: Dataset files with their respective Data Variables	24
3	Table III: Accuracy Scores of Collaborative-Filtering Recommendation Algorithm	53

List of Figures

Figure No.	Figure Name	Page No.
1	Content-Based Filtering	14
2	Cosine Similarity	18
3	Collaborative Filtering	19
4	Steps to be followed in Proposed System	21
5	Machine Learning	26
6	Popularity-Based Recommendations	47
7	Description-based Content Filtering Recommendations	48
8	Movie metadata-based Content Filtering Recommendations	48
9	Collaborative Filtering Recommendations	49
10	Hybrid Filtering Recommendations	49
11	Distribution of movies in their respective genre	50
12	Months for Blockbuster Movies	51
13	Heatmap of Movie Releases	51
14	Directors who have directed the highest revenue generating movies	52
15	Trend in Movie Runtime	53

Table of Content

	Title	Page No.
	Abstract	I
Chapter 1	Introduction	1-2
	1.1. Introduction	1
	1.2. Problem Statement	1
	1.3. Proposed Solution	2
Chapter 2	Literature Review	3-12
Chapter 3	Proposed Methodology	13-22
	3.1. Parameters	13
	3.2. Algorithms to be Used	13-20
	3.3. Evaluation Metrics	20-21
	3.4. Steps to be followed in Proposed System	21-22
Chapter 4	Implementation	23-49
	4.1. Parameters Used	23
	4.1. Dataset	23-25
	4.3. Hardware and Software Requirements	25-26
	4.4. Technologies Used	26
	4.5. Tools Used	27-28
	4.6. Source Code	28-46
	4.7. Experimental Recommendations	47-49
Chapter 5	Results	50-53
Chapter 6	Conclusion and Future Scope	54
	References	55-56
	Publication Screenshot	57

CHAPTER 1

INTRODUCTION

1.1. Introduction

Over the years, the Internet has been the backbone of modern technology. Due to the rapid development and major breakthroughs in different fields like healthcare, science and engineering, leisure and entertainment, etc. which heavily rely on technology, different applications and uses were discovered to support and improve human life. One such revolutionizing application in the field of entertainment is the online streaming media service. This service focuses on delivering audio and video content, like songs, and movies, to the viewer (client) over the Internet for a certain fee. Due to people being busier than ever, the last few years, have seen immense growth in this industry, especially the Over-the-Top (OTT) platforms that provide movies and web shows, as everyone loves to have some entertainment, which helps them to relax and get rid of their stress, and enjoy a peaceful time with their friends and family. These platforms attract viewers by providing them with fresh content and targeting audiences of all age groups, not letting any viewer get bored. Since, these days, the general public doesn't have enough time to visit movie theatres or a music concert, and when they are free which is typically on their weekends, they tend to stay at home and do not want to rush to such events. So these platforms came up with a solution and brought great content which is easily accessible and affordable. Now, people could binge-watch their favorite or trending movies and shows at their fingertips with an affordable subscription fee for these services. Some of the most successful online OTT platforms are Amazon Prime Video by Amazon, Netflix, Hotstar by Star India, Hulu by Walt Disney, etc. These platforms have successfully grasped the ideas and strategies and implemented them with the required technology, all in order to develop a highly practical and profitable product.

1.2. Problem Statement

The challenge for our group is to create a real-time recommendation system that is capable of recommending movies that are currently relevant to the user or in which the user has shown an interest in their past. The recommendation system should also aim to enhance user experience by providing personalized recommendations specific to an individual user, as it has a better chance to be viewed by the customer in comparison to general recommendations, thus, increasing conversion. In order to meet this challenge, we first have to figure out the different patterns and relations in the input dataset and draw various useful conclusions through analysis and visualization, which is very useful in developing the required system. Then, we will have to figure out the most suitable machine learning algorithms to use for creating a highly effective movie

recommendation model. We will also need to figure out which input features are the best to use for this task. Furthermore, we will like our recommendation system to be diverse i.e., it should recommend different kinds of movies to the user. This would require a complex model to successfully implement the whole application.

1.3. Proposed Solution

Many reliable and supporting tools and frameworks related to machine learning have been developed and implemented by nearly all online streaming platforms, and scientists are still researching and developing more useful tools. One of the most important tools and frameworks is the recommendation system. A recommendation system is a real-world software application that is used by most online streaming service platforms in order to suggest various kinds of shows and movies available on their platform to their customers and also provide their consumer base with the necessary information to help them decide which movies and shows are to their liking and meet their preferences, so that they may purchase the movie or show generating revenue for the business. The movies can be recommended on the basis of trends i.e., the most popular and the most watched movies on a site or on the viewer demographics, or it can also be based on an analysis of the viewer's past viewership with the various movies available on the platform, which acts as a prediction for their future behaviour. So online streaming media companies use various filtering algorithms that help in filtering out users or items in recommendation systems. These are generally based on user reviews and ratings or past user-item interactions. Some of the major filtering approaches are Collaborative Filtering, Content-based Filtering, and Hybrid Filtering.

CHAPTER 2

LITERATURE REVIEW

A considerable amount of research has been done on recommendation systems. Some of the research done by various researchers is as follows:

T. Keerthana [1] proposed an approach that uses the content-based (domain-dependent algorithm), collaborative, and hybrid filtering techniques to achieve accurate recommendations.

JH (Janghyun) Baek [2] proposed an approach in which he created an apparel-specific multi-step recommender system for Amazon users that first recommends items based on user experience and feedback. This is a deep learning-based model which tries to predict user ratings on products and based on this prediction suggests the one that has a higher rating. Using this output, the system further looks for similar products using two very distinct techniques: image-based processing and Natural Language Processing.

Mohammad R. Rezaei [3] tried to develop a recommendation model for digital music tracks available on Amazon. They analyzed, tested and integrated their proposed deep neural network (DNN) architecture with various traditional models to predict the rating scores that customers give to a music track.

Rohit Dwivedi [4] designed a model using matrix factorization using a user-based nearest-neighbor collaborative filtering approach. Initially, using the pivot function a pivot table was made based on the user. The models were evaluated based on different metrics like RMSE, mean square error, mean absolute error, and the Average actual ratings and Average predicted ratings were calculated.

Huang [5] proposed a Graph-based recommender system which is a unique approach to recommender systems as it neither uses collaborative filtering nor the content-based approach. Instead, it combines these two approaches and produces a hybrid model without the need to use a top-level classifier or regression model.

Zhang [6] tested several Deep Learning approaches for recommendation systems, such as Multilayer Perceptron (MLP), Autoencoder (AE), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN). Their argument is that Deep neural networks are highly capable in modeling the nonlinearity in data with a non-linear activation functions like the sigmoid function. This enables it to capture multiple patterns in complex user-item interactions. Deep neural networks also enable automatic feature learning from raw data, reducing the requirement to perform intensive feature engineering.

Linden [7] proposed a system that uses item-based collaborative filtering which gives recommendations by utilizing user-generated signals, like explicit item ratings, that were gathered from other users and based on items that the user has purchased or has rated, which are then paired with similar items.

Md Zaid Ahmed [8] decided to use the Collaborative Filtering Algorithm to improve the accuracy of product recommendations. They used products from Amazon e-commerce website to design a practical model for some specifically designed scenarios. Their system uses cosine similarity as metric to find similar products on the basis of multiple user ratings and generate a matrix that helps in recommending new products to other users.

Ashrita Kashyap [9] introduced Movie REC, a movie recommendation system that allows a user to select his/her choices from a given list of attributes and then recommends the user a list of movies based on the cumulative weight of the different selected attributes and using the K-means algorithm.

Yeole Madhavi [10] used Content-based Recommendation using CountVectorizer and Cosine Similarity, Content-based Recommendation using TfidfVectorizer and Cosine Similarity and their Hybrid Algorithm for their recommendation system. His implementation resulted in getting better results for the hybrid algorithm when compared to the other two algorithms. Rahul

Pradhan [11] successfully implemented a movie recommendation system using the Collaborative-Filtering Algorithm and found out that these systems have certain limitations due to which they do not recommend efficiently to its users. He found that even though Collaborative Filtering is the most successful and powerful algorithm, but has some high runtime and faces some major issues like data Sparsity which he concluded could be removed by using a Hybrid movie recommendation system.

Y. Koren [12] suggested approaches utilizing matrix factorization to enhance recommender systems, particularly for the Netflix Prize dataset. They evaluated and compared several matrix factorization techniques, such as singular value decomposition (SVD), non-negative matrix factorization (NMF), and regularized matrix factorization (RMF), and introduce an upgraded version of RMF, known as SVD++, which involves implicit feedback. Their study indicates that matrix factorization techniques are more efficient in collaborative filtering and outperform conventional neighborhood-based approaches. Furthermore, they also introduced a new and unique approach for predicting rating errors, which can assist in selecting the most suitable algorithm and determining its parameters.

Sarwar [13] presented a new approach to collaborative filtering technique that

focuses on the similarity between items rather than users. They proposed an algorithm that computes the similarity between items based on the ratings given to them by users and then uses this similarity to make recommendations. The authors also compared their item-based approach to traditional user-based collaborative filtering and demonstrated that it was more efficient and accurate, especially in cases where there are a large number of users and items. They also presented a variation of their algorithm that takes into account the popularity of items and revealed that it further enhances the performance of the system. The results of their experiments suggest that their approach can lead to better recommendations, especially in large-scale systems.

Burke [14] suggested a novel approach to improve recommendation quality by incorporating social networks. This approach was based on the idea that people are more likely to trust recommendations from their social circle rather than from unknown sources. The authors presented reasonable evidence from previous studies that have demonstrated the effectiveness of social networks in improving the quality of recommendations. Their proposed approach involved creating a social network for each user and utilizing it to identify potential recommenders. Their approach was also able to overcome the cold-start problem that arises when recommendations cannot be made for new users with limited data about their preferences. Their study revealed that the integration of social networks can increase recommendation quality by up to 20%.

Salakhutdinov [15] proposed a new probabilistic matrix factorization model that incorporates Gaussian priors on the latent factors, allowing for uncertainty in the factorization process and uses Bayesian inference to learn the model parameters. They also demonstrated the results of experiments on several benchmark datasets in predicting user ratings in recommender systems. Their system also had the ability to handle missing data and the ability to incorporate additional information such as user and item features. They concluded that their probabilistic matrix factorization method outperforms traditional matrix factorization methods and non-probabilistic matrix factorization methods such as matrix completion.

Cheng [16] described a co-clustering based collaborative filtering framework that is designed to handle large datasets by decomposing the user-item matrix into smaller sub-matrices that can be processed in parallel. They also introduced a novel regularization term that encourages sparsity in the co-clustering solution, improving prediction accuracy.

Liu [17] described a multi-view clustering framework for movie recommendation that can leverage multiple sources of information to improve prediction accuracy. The authors defined a novel objective function that incorporates multiple views of the data and a regularization term that encourages sparsity in the solution,

which they tested on several benchmark datasets. Their framework has the ability to handle missing data and also to incorporate additional sources of information. X. Huang [18] proposed a new collaborative filtering method based on weighted non-negative matrix factorization (WNMF) for recommendation systems as according to the authors, this method is better in capturing the inherent user-item interaction patterns than the traditional matrix factorization methods. In their system, they introduced a novel weight term in the WNMF framework that takes into account the relevance of each user-item interaction. This weighting scheme allows the model to assign more importance to highly relevant interactions and less importance to irrelevant ones, thus improving the quality of recommendations.

Dutta [19] developed a movie recommendation system that integrated collaborative filtering that relies on the similarity of users' preferences to make recommendations and fuzzy logic which is used to handle uncertainty and imprecision in the ratings given by users, to provide personalized recommendations to users. They evaluated its performance and concluded that this approach was superior to other traditional techniques.

By analyzing the numerous views and approaches towards research and development of recommendation systems as described above, it is evident that most researchers took a linear approach by using the collaborative-filtering algorithm which only focuses on user ratings and past user interactions and discards any kind of association with other parameters. Also, most of the above-mentioned research does not deal with the cold-start problem which is clearly visible as very less amount of work has been done involving content-based filtering which also considers other item attributes and their interactions, though there have been recent advancements over the past few years. For recommending new movies to the user, we can use movie attributes like genre, cast, crew, revenue, etc. which is possible in content-based filtering. Some of the researchers have proposed the idea of using a hybrid system but they have only proposed a theoretical overview of this approach. We can also compare and evaluate these algorithms using multiple metrics.

So, the main aim of this research is to solve the problem of developing a real-time practical recommendation system that is capable of recommending movies customized to the user, by using the best possible approach and considering different relevant factors while doing so. The recommendation system should also aim to enhance the user experience by providing personalized recommendations specific to an individual user, as it has a better chance to be viewed by the customer in comparison to general recommendations, thus, increasing conversion. Furthermore, we will like our recommendation system to be diverse i.e. it should recommend different kinds of movies to the user.

TABLE I

STUDY DONE IN THE FIELD BY VARIOUS AUTHORS IN THE REFERENCED LITERATURE

Author	Objective	Methods/ Techniques	Findings	Limitations
JH (Janghyun) Baek et. al. [2]	To design an apparel specific Amazon Product recommendation system based on user feedback and ratings.	Deep Learning, Image-based Processing and NLP.	This multilayer model first suggests products with a higher rating and using this output, the system further looks for similar products using image-based processing and Natural Language Processing, which is very practical in a real-world situation.	Measurement of the system's performance due to the lack of available data and resources for online evaluation, issues with scalability and design, overfitting, and the difficulty in finding the optimal model configuration, not always providing ideal recommendations, especially when based on image similarity.
Mohammad R. Rezaei [3]	To design a model to predict highly accurate reviews' rating score.	Multinomial Naïve Bayes, Logistic regression, Deep Neural Networks (DNN).	DNN model (MSE score of 0.51-0.53) outperforms other methods significantly Even though the RL (MSE score of 1.03) and MNB (MSE score of 1.29) models are not as accurate as the DNN is, but they are relatively simple, and they need no specific	The author faced challenges in text pre-processing and encoding due to limited knowledge in NLP, which could affect recommendation quality. Limited dataset size resulted in using a small-sized model to prevent overfitting, leading to a trade-off between model

			<p>effort to setup them but in the end, they are not accurate enough. So, they can be considered in problems with no need for high accuracy.</p>	<p>complexity and prediction performance.</p>
<p>Zan Huang et. al. [5]</p>	<p>To develop a recommendation system for an online Chinese bookstore setting.</p>	<p>graph-based algorithm that integrates content-based and collaborative recommendation algorithms, Hopfield Net Algorithm.</p>	<p>Based on hold-out-test results, hybrid system performed better but the improvement was not significant. Based on plot subject test, content-based system outperformed the other algorithms.</p>	<p>Exploiting high-degree association did not show significant benefit, the data does not fully represent customer interests. Future work includes refining weighing schemes, adjusting parameters, and exploring other recommendation methods.</p>
<p>Shuai Zhang et. al. [6]</p>	<p>To review the taxonomy of deep learning-based recommendation models, a thorough summary of the state-of-the-art in the field, explore emerging</p>	<p>Deep Learning</p>	<p>Gives a comprehensive review of the application of deep learning in the field of recommendation systems, exploring trends and also highlighting the issues and direction of the</p>	<p>More future work is required for improving the Scalability and time complexity of the system lacks unified evaluation standards, making it challenging to compare models. It also faces difficulty of test samples and</p>

	trends and provide fresh perspectives on the development.		development in the field.	control over inference
Greg Linden et. al. [7]	To create an Amazon Product recommendation system using item-to-item collaborative filtering and comparing it with traditional collaborative filtering algorithms	Item-to-Item Collaborative filtering	item-to-item collaborative filtering algorithm is fast even for extremely large data sets. Its recommendation quality is excellent. Unlike traditional collaborative filtering, the algorithm also performs well with limited user data.	The algorithm relies on the precomputation of the expensive similar-items table offline. Any inaccuracies or deficiencies in this table could affect the quality of recommendations. It also faces challenges when dealing with sparse or incomplete user profiles resulting in non-accurate or personalized recommendations.
Md. Zaid Ahmed et. al. [8]	To provide an in-depth explanation of a system that classifies and recommend products on Amazon to the user.	Collaborative Filtering, Sentiment Analysis, Cosine Similarity, RMSE	Item-based collaborative filtering is an effective method for obtaining high-quality recommendations with a RMSE accuracy of 0.52, which is better than average and can be used for large dataset as well.	Reliance on historical data, difficulty in handling new users and products, limited diversity in recommendations, and challenges with sparse data. Scalability is not explicitly addressed in the paper.

Ashrita Kashyap et al. [9]	To introduce “MOVREC” a movie recommendation system to recommend movies based on the user’s choices.	Cumulative Weight and K-Means Algorithm with Collaborative Filtering	A successful response from a small set of users was obtained and the system was tested successfully.	Challenge of evaluating system performance since movie recommendations are subjective and based on individual opinions, a larger dataset is needed for more meaningful results. Need to incorporate different machine learning and clustering algorithms to study and compare the outcomes further.
Rahul Pradhan et al. [10]	To provide an overview of various techniques create a recommendation system that recommends movies to the user.	Item based Collaborative Filtering	Collaborative filtering is most successful and powerful algorithm. The proposed approach can handle quite a big amount of data effectively.	Only uses collaborative filtering and faces challenges such as high runtime and data sparsity. It also needs to address weaknesses and improve the user interface.
Y. Koren et al. [11]	To suggest approaches utilizing matrix factorization to enhance recommender systems, particularly for the Netflix Prize dataset	Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), and Regularization	Their study indicates that matrix factorization techniques are more efficient in collaborative filtering and outperform conventional neighborhood-based approaches. Furthermore, they	Published in 2009, so it fails to incorporate major developments in the field over the last decade. Also, it focuses only on matrix factorization techniques and do not provide explicit explanations or insights into why certain

		zed Matrix Factorization (RMF), SVD++,	also introduced a new and unique approach for predicting rating errors, which can assist in selecting the most suitable algorithm and determining its parameters. They also introduced an upgraded version of RMF and a new approach for predicting errors.	recommendations are made.
B. Sarwar et. al. [12]	To present a new approach to collaborative filtering technique that focuses on the similarity between items rather than users.	Item-based Collaborative Filtering	This item-based approach is more efficient and accurate than traditional user-based collaborative filtering, especially in cases where there are many users and items. A new variation of this algorithm further enhances the performance of the system which can lead to better recommendations, especially in large-scale systems.	Only focuses on collaborative filtering and the improvement over other techniques is not significantly large.

R. Burke et. al. [13]	To provide a theoretical study on incorporating social networks to provide recommendations.	Not mentioned	This approach was able to overcome the cold-start problem and the study also revealed that the integration of social networks can increase recommendation quality by up to 20%.	Limited to social network incorporation and does not address the challenges associated with obtaining and utilizing social network data for recommendations. Also does not incorporate recent advancements in the field.
-----------------------	---	---------------	---	--

CHAPTER 3

PROPOSED METHODOLOGY

So, the different obstacles that we need to overcome involve identifying parameters that after analysis provides deep insights and helps in drawing various useful conclusions; figuring out and deciding the appropriate machine learning algorithms to use for creating a highly effective movie recommendation system; evaluating the performance of used algorithms and through a comparative study, choosing the best one among them. This would require a complex system design to successfully implement the whole application.

3.1. Parameters

In order to successfully design the desired recommendation system, the first step is to identify the different parameters that could possibly exist and could be used to analyze various possible algorithms which are useful to build recommendation systems and find the best one. The identified parameters are as follows:

- 1) **Dataset and Data Variables:** Datasets are used by movie recommendation systems to have access to a large sample and a rich variety of relevant information and several features (variables) which may include movie ratings, movie-related information like cast and crew details, budget, revenue, viewer-movie interactions, etc. which can be analyzed to find patterns and provide recommendations to the users.
- 2) **Methodology:** This describes the approach and the implementation specifications followed to develop the functional movie recommendation system.
- 3) **Similarity Metric:** This parameter helps in finding similarities between multiple users and multiple items i.e., it helps in identifying viewers with similar movie preferences and similar types of movies. Some common mathematical similarity metrics are Jaccard's similarity, Cosine similarity, etc.
- 4) **Evaluation Metric:** This parameter is used to evaluate and quantitatively measure the performance of a movie recommendation system. Some common performance evaluation metrics are precision, recall, root mean square error, etc.

3.2. Algorithms To be Used

This section gives a detailed explanation of the methods and machine learning algorithms used to design our movie recommendation system. The designed recommendation system uses various filtering algorithms like Collaborative Filtering, Content-Based Filtering and then combines both of these algorithms into a Hybrid Filtering algorithm. The concept and working of all previously

mentioned algorithms have been defined as follows:

1. **Popularity-Based Filtering:** This is one of the first recommendation systems to ever get developed and successfully deployed. As its name suggests, it has a working principle based on popularity or the latest trend. These systems check the movies which are most popular among the users and directly recommend those without considering other factors like user history i.e. what are movies that the user has viewed or interacted with in the past, etc. This has a chance to increase user engagement when compared to no recommendation system. But such systems have a huge drawback that they do not provide personalized recommendations to users.

2. **Content-Based Filtering:** Content-Based filtering recommendation system tries to make suggestions on attributes of items that a customer previously liked or interacted with. Such a system first builds item profiles and then derives a user profile from them. This user profile contains features that are important for a particular individual. Once the system knows about a customer's preferences, it can recommend new items according to it. This technique considers a wide variety of factors which include both objective and descriptive factors before making any recommendations.

We have shown the content-based filtering approach through a visual diagram in Figure 1:

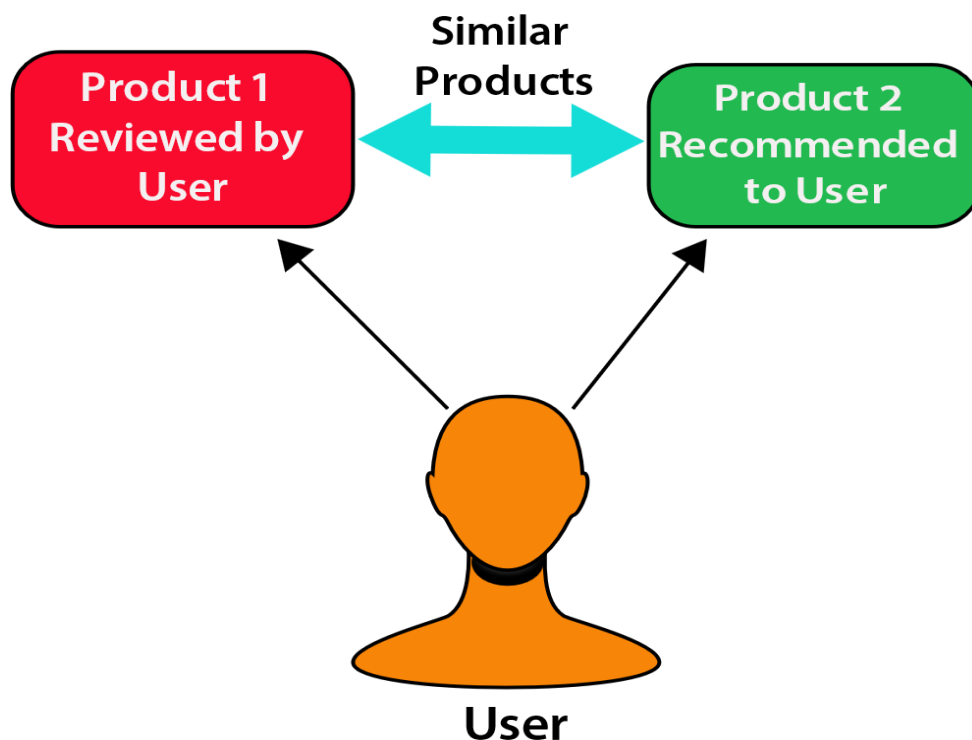


Fig.1. Content-Based Filtering

For content-based filtering, we have used the TF-IDF (Term Frequency Inverse Document Frequency) Vectorizer technique, which is used to transform text into a meaningful representation of numbers that are used to fit machine learning algorithms.

Term Frequency is defined as the numerical frequency with which a term appears in a particular document. So it is document-specific. The term frequency of a term ‘t’ in a specific document ‘d’ is calculated as:

$$tf(t, d) = \frac{\text{Number of times } t \text{ appears in } d}{\text{Total number of terms in } d} \quad (1)$$

Inverse Document Frequency is a numerical representation that measures how important a term is across the corpus of documents i.e. is it a common or rare term based on its appearance across the entire set of documents So it is consistent across all documents. The general equation of IDF value of a term ‘t’ in the document collection (corpus) ‘N’ is:

$$idf(t) = \log_e \left(\frac{N}{\text{Number of documents having term } t} \right) \quad (2)$$

When a word is common and appears in many documents, its IDF value (normalized) will tend towards 0, while a rare word that appears in only a few documents will tend towards an IDF value of 1.

So for a term ‘t’ that is rare across the entire ‘N’ number of documents of the corpus and only df(t) number of documents contain this term, IDF is calculated as:

$$idf(t) = \log_e \left(\frac{1+n}{1+df(t)} \right) + 1 \quad (3)$$

An important point to note in the above equation is that 1 is added to the denominator because as the term ‘t’ is rare very less documents contain it i.e., df(t) approaches 0. This may result in a division by 0 while calculating the IDF value, which is not practical. So we add 1 to the denominator to avoid this situation.

Similarly, for a term ‘t’ that is common across the entire ‘N’ number of

documents of the corpus and the number of documents containing this term is $df(t)$, IDF is calculated as:

$$idf(t) = \log_e \left(\frac{N}{df(t)} \right) \quad (4)$$

Then we multiply the TF and IDF values to get the required TF-IDF value. So the TF-IDF weight of a term 't' in a document 'd' is given as:

$$tfidf(t, d) = tf(i, d) \times idf(t) \quad (5)$$

When a content-based movie recommendation system uses the TF-IDF vectorizer, it first tokenizes (converting into a stream of characters) the documents containing information on the movie profiles and for each movie, calculates the TF-IDF scores for each term on the basis of their frequency in the corpus and determines their importance. The vectorizer then converts them into vectors and uses them to find similar movies. Based on a user's viewing history and preferences, the content-based system will recommend new movies with high TF-IDF scores for terms that are relevant to the user.

3. Collaborative filtering: Collaborative filtering recommendation systems first look at interactions between users and items try to find either similar types of users or similar types of products and based on this similarity, calculate prediction scores and then recommend new products to target users. Every user and item is described by a feature vector in the same space.

Collaborative Filtering is mainly of two types:

a) Item-Item based Collaborative Filtering: This technique starts by searching the items that the user has had prior interactions with and finds similar items to it with the help of similarity metrics, and using which a prediction function is defined which then suggests and recommends items to the user.

The steps involved in Item-Item based Collaborative Filtering are as follows:

Step-1: First, convert the given data in the form of a user-item matrix.

Step-2: We then start building the model by finding similarity between all the item pairs. We create vectors for each individual item using the user ratings that are known and then we find the similarity between them which can be found in multiple ways, but we have used cosine similarity in our framework.

Cosine Similarity: Cosine similarity is a mathematical tool used to assess and quantify the similarity between two given non-zero vectors, which is measured in terms of the cosine of the angle between the two vectors ($\cos \theta$). Specifically, it measures the similarity in the direction or orientation of the vectors irrespective of their magnitude or scale. Both vectors need to lie on the same plane in the same inner product space, meaning they must produce a scalar through dot product multiplication.

Cosine similarity ranges from -1 to 1, with -1 representing diametrical vectors in opposite directions, 0 representing two mutually perpendicular vectors, and 1 representing vectors in the same direction.

Mathematically, the cosine similarity of vectors A and B is determined by finding the dot product of the two vectors and dividing it by the product of their lengths (magnitude) and is given as follows:

$$Sim(A, B) = \frac{A \cdot B}{|A| |B|} = \frac{\sum_{i=1}^n A_i * B_i}{\sqrt{\sum_{i=1}^n A_i^2} * \sqrt{\sum_{i=1}^n B_i^2}} \quad (6)$$

As cosine similarity depends on orientation i.e., the angle between two vectors and not their lengths, it has very high computational efficiency and is also pretty useful in high-dimensional spaces, as in spaces with a high number of dimensions, it is very difficult to calculate lengths of vectors due to the high complexity of space variables.

Similarly, in the case of data sparsity where most of the dimensions of a vector are very small (approach 0) or are exactly equal to 0, cosine similarity determines the similarity between two vectors only based on non-zero dimensions thus, allowing cosine similarity to also be able to operate successfully for sparse data. So, cosine similarity comes with a lot of useful advantages and is thus, one of the most important mathematical tools used in machine learning. Figure 2 depicts the vectors and their orientation which helps us to easily understand the concept of Cosine Similarity.

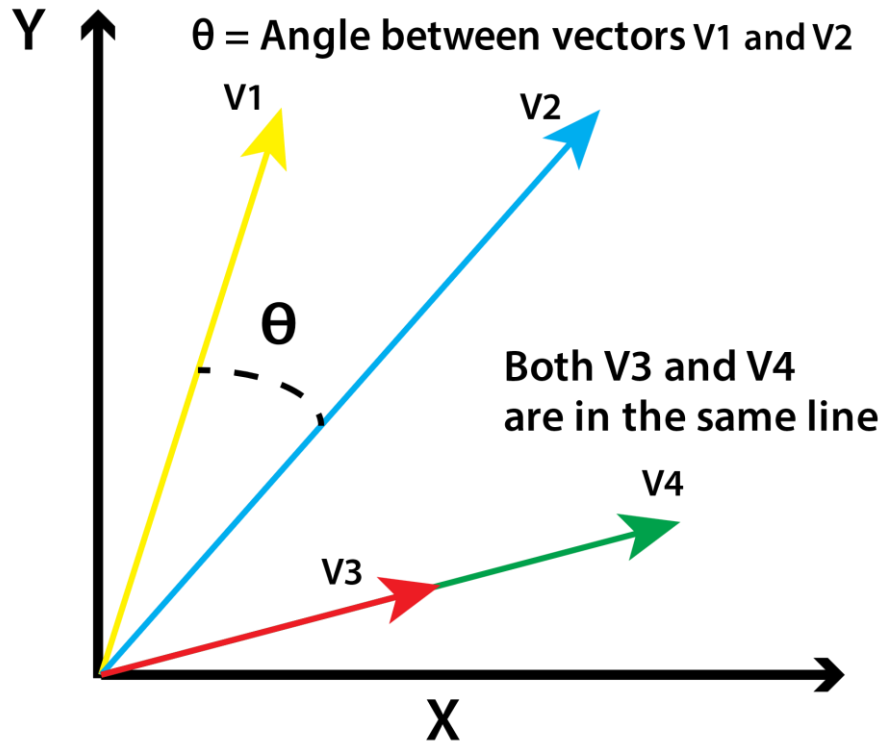


Fig.2. Cosine Similarity

Step-3: Calculate the recommendation or rating score as:

$$rating(U, I_i) = \frac{\sum_j rating(U, I_j) * Sim(I_i, I_j)}{\sum_j Sim(I_i, I_j)} \quad (7)$$

b) User-User based Collaborative Filtering: This technique looks for similar users i.e., users with similar interests, based on the items the users have already rated or interacted with, and recommends other items that might be appealing to other target users/audiences.

The steps involved in User-User based Collaborative Filtering are as follows:

Step-1: Convert the given data in the form of a user-item matrix based on the interest of a user in a certain item.

Step-2: We then start building the model by finding the set of similar users to the target user for which we have again applied cosine similarity for vectors created using the user-item matrix.

Step-3: Calculate the recommendation score as:

$$rating(U, I_i) = \frac{\sum_j rating(U, I_j) * Sim(U_i, U_j)}{\sum_j Sim(U_i, U_j)} \quad (8)$$

In order for both of these collaborative filtering methods to work, we have used the Singular Value Decomposition (SVD) technique, which is basically a matrix factorization technique that decomposes any given matrix into 3 generic and familiar matrices. The SVD of a $m \times n$ matrix A is given by the formula:

$$A_{m \times n} = U_{m \times m} S_{m \times n} V_{n \times n}^T \quad (9)$$

where U and V are orthogonal matrices with orthonormal eigenvectors chosen from AA^T and AA^T respectively. S is a diagonal matrix with r elements equal to the root of the positive eigenvalues of both matrices U and V , which have the same positive eigenvalues. The diagonal elements are composed of singular values.

Figure 3 shows a visual diagram of the collaborative filtering approach.

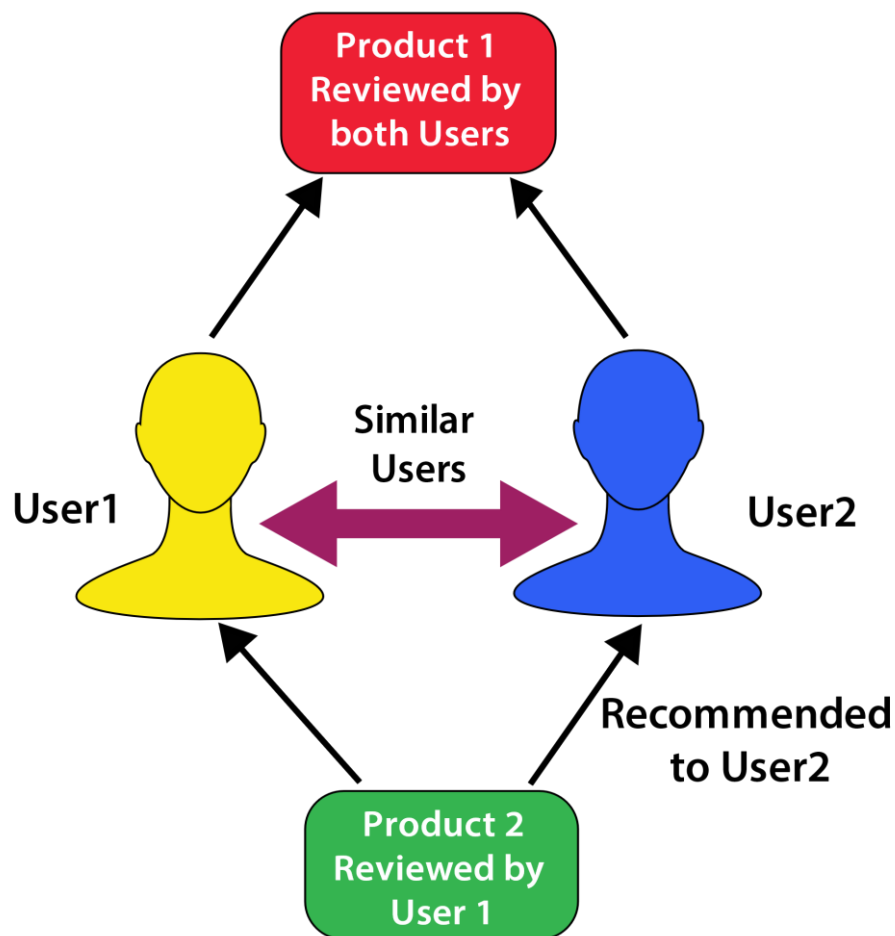


Fig.3. Collaborative Filtering

4) Hybrid Filtering: This approach combines the previously mentioned algorithms and then offers more accurate and diverse recommendations to the user. This algorithm is generally preferred and used by the major leaders of online streaming services since it has the advantages of all the other techniques and overcomes the limitation barrier of each individual approach. Its effectiveness depends on the quality of all the combined algorithms, the implementation details as well as the accuracy and consistency of input data.

3.3. Evaluation Metrics

The evaluation metric is used to evaluate and quantify the performance of a machine learning model. The choice of evaluation metric depends on the specific problem being solved and the goals of the model. The following metrics will be used to evaluate the different algorithms applied in the proposed system:

1) Root Mean Square Error (RMSE): RMSE measures the average distance between the predicted values and the actual values. It is calculated by taking the square root of the average of the squared differences between the predicted values and the actual values. Its equation is given as:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (10)$$

where n is the number of observations, y_i is the actual value of the target variable for the i th observation, and \hat{y}_i is the predicted value of the target variable for the i th observation.

2) Mean Absolute Error (MAE): MAE measures the average absolute distance between the predicted values and the actual values. It is calculated by taking the average of the absolute differences between the predicted values and the actual values. It is mathematically calculated using the equation:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (11)$$

where n is the number of observations, y_i is the actual value of the target variable for the i th observation, and \hat{y}_i is the predicted value of the target variable for the i th observation.

Here are some key differences between the two:

a) RMSE is more sensitive to outliers than MAE, since it squares the differences between predicted and actual values. This means that large errors have a greater impact on the RMSE score than on the MAE score.

b) Because of its squaring operation, RMSE gives more weight to large errors compared to MAE. So, RMSE is generally used when large errors are particularly undesirable.

c) Both RMSE and MAE are scale-dependent, which means that their values are affected by the units of the target variable. In other words, changing the scale of the target variable (e.g., from dollars to euros) will change the value of the evaluation metric.

3.4. Steps to be followed in the Proposed System Design

Figure 4 shows the steps to be followed for our proposed framework to solve the problem.

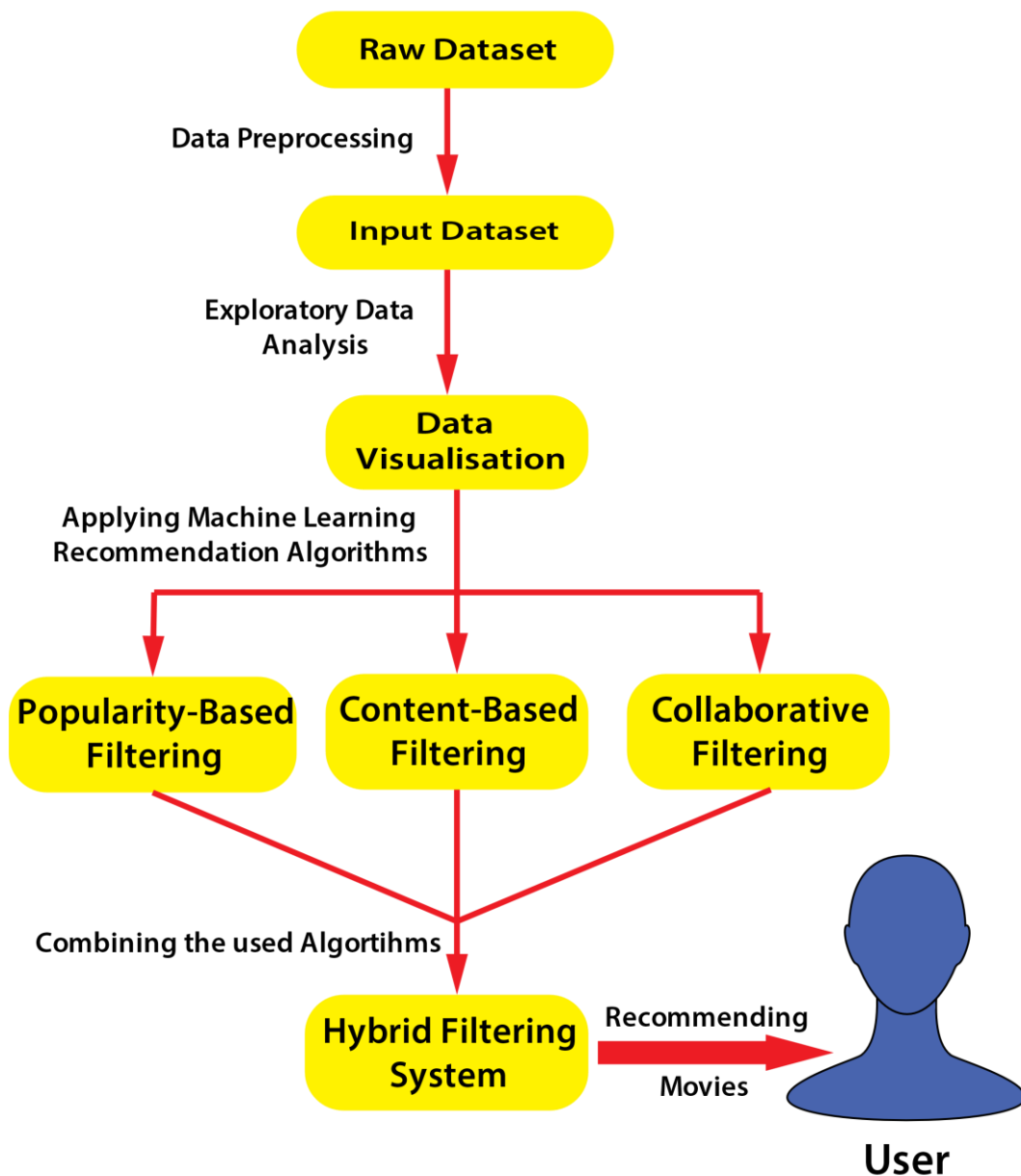


Fig.4. Flow of Proposed System

1) Acquisition of training data set: The accuracy of a machine learning algorithm mainly depends on the extent of the correctness of the input data set and the number of parameters considered. Our input data set is originally from the online movie critique giant “Movie Lens (TMDB)”. The objective of our data set is to provide data that is helpful in analyzing and recommending movies to the user that the user may be interested in and has a higher chance of viewing, generating revenue for any online streaming platform.

2) Data pre-processing: Data pre-processing is the most important process. Sometimes, movie-related data contains a lot of incomplete and incorrect values. Such data can diminish and undermine the accuracy and consistency thus, compromising the system. This makes it necessary to implement strategies to improve the quality of data to be properly utilized for the Machine Learning Techniques that will be applied later effectively and also to be used for other tasks, data pre-processing is performed, which is essential to obtain accurate results and successfully recommend the precise content to the user. We generally perform this entire process in two steps:

a) Data Cleaning by Removing Missing Values- We should remove all the instances that have zero (0) as their value or contain any missing values as they are irrelevant and bring inconsistency to the input data. Therefore, these instances are eliminated. It also helps us to reduce the dimensionality of data which helps us to decrease the required computational time improving efficiency.

b) Splitting of data- After cleaning irrelevant parts of the input data, we generally normalize the data, i.e., we transform the columns of the input data set to the same scale to prepare the data for training and testing the machine learning model. The training process helps to correlate the processed output against the actual output. This helps us to modify the training model, which is the most suitable to achieve our goal.

3) Exploratory Data Analysis and Visualization: In this step, we create the necessary graphs and plots to display different data distributions and observe relationships between data which will allow us to understand the data in more depth and draw inferences. It is a very critical part since it determines how the model will be built.

4) Apply Machine Learning to Create a Model: When data is finally ready, we will apply the different Machine Learning algorithms discussed earlier, then analyse the performance of each technique using certain metrics as it will play a major role in the final model which will recommend movies to the customers.

CHAPTER 4 IMPLEMENTATION

4.1. Parameters Used

The proposed system is designed and based on the following parameters:

- 1) User preferences: The system considers the user's past viewing history, ratings, and likes/dislikes to determine their preferences.
- 2) Movie attributes: The system takes into account various attributes of the movies such as genre, release year, language, director, actors, and plot to recommend movies to the user.
- 3) Similarity: The system may recommend movies that are similar to the user's previously viewed movies or movies they have liked.
- 4) Methodology: Popularity-based, Collaborative filtering, Content-based filtering and Hybrid filtering.
- 5) Diversity: The system may recommend movies that are different from the user's usual preferences to introduce them to new genres or styles of movies.

4.2. Dataset

The performance of a machine learning algorithm is primarily influenced by the accuracy and correctness of the input data set and the number of parameters considered. Our input data set is an open-source dataset from the online movie critique giant "Movie Lens (TMDB)". The objective of our data set is to provide data that is helpful in analysing and recommending movies to the user that the user may be interested in and has a higher chance of viewing, generating revenue for any online streaming platform. This dataset contains the following CSV files:

- 1) movies metadata.csv: The Movies Metadata file is the primary file which gives a comprehensive overview of the movies included in the MovieLens dataset. With its extensive collection of information on 45,000 movies, the Movies Metadata file is a cornerstone of the MovieLens dataset.
- 2) keywords.csv: This file includes the key plot elements represented by keywords for each movie.
- 3) credits.csv: This file is dedicated to provide a comprehensive overview of the talented individuals who were involved in behind-the-scene works in the MovieLens movies, its cast and crew which are critical components of any movie.

4) links.csv: This file provides an easy, straightforward and user-friendly way to cross-reference the movies in the dataset with their external identifiers i.e., with their corresponding IDs on TMDB and IMDB.

5) links_small.csv: While the full MovieLens dataset includes a vast collection of movies, this file provides information on a smaller subset with 9,000 movies with their IMDB and TMDB IDs just in case if a small number of movies are required for cross-referencing with other databases.

6) ratings_small.csv: The ratings file includes a subset of 100,000 rating data on how 700 users rated 9,000 movies in the links_small.csv file, offering insight into the preferences and behaviour of our user community.

TABLE II
Dataset files with their respective Data Variables

<i>Dataset</i>	<i>Data Variables</i>
movies metadata.csv	10 variables including budget, language, genre, etc.
keywords.csv	2 variables including id and keyword.
credits.csv	3 variables including cast, crew and id.
links.csv	3 variables including movieId, imdbId and tmdbId.
links_small.csv	variables including movieId, imdbId and tmdbId.
ratings_small.csv	4 variables including userId, movieId, rating and timestamp.

Table I shows the dataset files and their respective variables (data columns) used to experiment on the designed system.

Content-Based filtering does not require a train-test split because it is using a similarity metric to generate recommendations based on the similarity of the metadata of the input movie to the metadata of all movies in the dataset. Collaborative filtering uses K-Fold from the surprise library to split the dataset into k folds for cross-validation which means that the data is split into k parts (or “folds”) of roughly equal size, and in each iteration of the cross-validation, one of the folds is used as the test set and the other four are used as the training set. This process is repeated k times, with each fold used once as the test set. So, there is no explicit train-test split required for collaborative filtering. Hybrid filtering

does not require a train-test split because the model was trained on the entire dataset.

While our proposed system has been designed with the specific attributes of the MovieLens dataset in mind, it can be adapted for use with other datasets as well. With some modifications to the input data and model parameters, our system can be applied to a wide range of datasets with varying attributes and characteristics.

4.3. Hardware and Software Requirements

When installing or running any application, including machine learning algorithms, it is essential to consider both hardware and software requirements. Hardware requirements specify the minimum or recommended specifications for the physical components of a computer system, such as the CPU, RAM, storage, GPU, and other components, that are necessary to run a particular software or application. Software requirements refer to the minimum or recommended specifications for the software or application that is being installed or run on a computer system. Software requirements typically include the operating system, software dependencies (such as libraries or frameworks), and any other components or configuration settings required by the software to function properly. In the case of machine learning, the size and complexity of the dataset, as well as the specific algorithms and techniques being used, will often determine the necessary hardware and software requirements for optimal performance. This section specifies the hardware and software requirements for the proposed system which are as follows:

1) Recommended Operating Systems:

- Windows: Windows 8 or higher
- Mac: macOS Sierra or higher
- Linux: Ubuntu

2) Hardware Requirements

- a) Processor: A multi-core processor of minimum 1 GHz; Recommended 2 GHz or more; like Intel Core i5 or higher is recommended for faster data processing.
- b) Memory: A minimum of 4 GB RAM; Recommended 8 GB or more.
- c) Storage: A minimum of 32 GB; Recommended 64 GB or more of HDD (Hard Disk Drive) or SSD (Solid State Drive).
- d) Graphics Processing Unit (GPU): A minimum of consumer-grade GPUs such as NVIDIA GTX 16 series or AMD Radeon RX GPUs or higher-end GPUs.
- e) Internet Connectivity: Ethernet Connection (LAN) or Wireless adapter (Wi-Fi)

3) Software Requirements

- Programming Language: Python v.3.0 or higher.
- Programming Libraries: Pandas, Numpy, Matplotlib, Seaborn, Scikit Learn, Surprise.
- IDE (Integrated Development Environment): Google Colab, Jupyter Notebook.

4) Supported Browsers:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge

4.4. Technologies Used

1. Machine Learning: In Machine learning, a computer program learns some tasks T using knowledge E (experience i.e. past data) and its performance P improves at doing those tasks. Thus, machine learning is the concept in which a machine learns on its own without it being needed to be programmed explicitly. In machine learning, we generally build (program) a model using an algorithm, which can be used for the input data to produce the desired results. Figure 5 shows this basic concept of machine learning i.e. in machine learning, all n inputs (Input 1, Input 2, Input 3, ... Input n) which are to be given to the machine learning model are known and the output that the model should produce is also known, we just have to find the best and most efficient unknown model that can be used to get the required output, based on these known parameters.

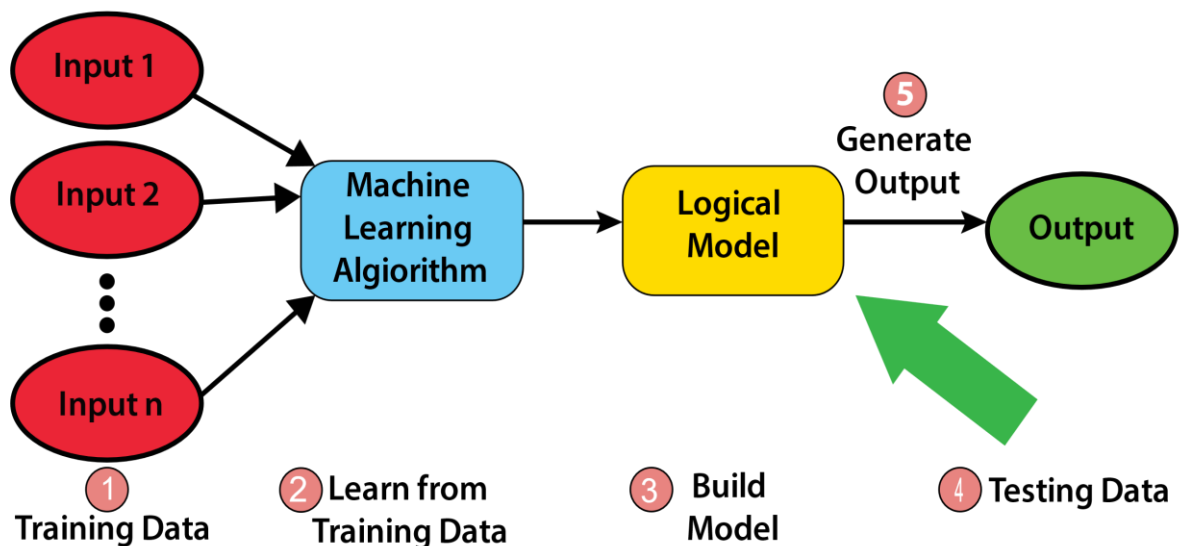


Fig.5. Machine Learning

2. **Filtering Algorithms:** These are algorithms that help in filtering out users or items in recommendation systems. These are generally based on user reviews and ratings or past user-item interactions. Some of the major filtering approaches are Collaborative Filtering, Content-based Filtering, and Hybrid Filtering.

4.5. Tools Used

1. Python: Python is a dynamic, object-oriented, high-level programming language known for its interpreted nature. Its dynamic semantics and a wide range of built-in data structures along with it employing dynamic typing and binding, makes it highly appealing and a preferred choice for Rapid Application Development and also as a scripting or glue language for connecting different components. With its emphasis on readability, Python's simple, straightforward and easy-to-learn syntax contributes in the reduction of the burden and cost of program maintenance. The language promotes modular programming and code reuse through support for modules and packages. Python's versatility extends to its availability on major platforms, providing the Python interpreter and a comprehensive standard library at no cost in both source and binary forms, enabling widespread distribution.

The Python libraries which have been used in our project are:

- A. Pandas:** pandas is a software library present in the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- B. NumPy:** NumPy stands for Numerical Python and was created in 2005 by Travis Oliphant. NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
- C. Matplotlib:** Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy.
- D. Seaborn:** Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with data frames and the Pandas library. The graphs created can also be customized easily.

E. Scikit-learn (Sklearn): Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

2. Google Colab: Colaboratory, often referred to as "Colab," is a Google Research product that acts as an online Integrated Development Environment (IDE), which enables users to write and execute Python code directly in their web browsers. It is exceptionally valuable for a wide range of applications, including machine learning, data analysis, and educational purposes. In technical terms, Colab functions as a hosted Jupyter notebook service, streamlining the setup process for users and providing them with cost-free access to computing resources, including GPUs., without any charges.

4.6. Source Code

Importing the Libraries

```
!pip install scikit-surprise
```

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer, CountVec-
torizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarit-
y
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
```

```
pd.options.display.max_columns=None

import warnings; warnings.simplefilter('ignore')

from google.colab import drive
drive.mount('/content/drive')
```

Reading the Larger Dataset

```
meta = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/movies_m
etadata.csv')
meta.head()
```

Let's identify the total number of null values in the data:

```
meta.isnull().sum()
```

Cleaning the Dataset

```
meta['genres'] = meta['genres'].fillna('').apply(literal_eval).app
ply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])
```

```
meta['production_companies'] = meta['production_companies'].fillna('
').apply(literal_eval).apply(lambda x: [i['name'] for i in x] if is
instance(x, list) else [])
```

```
meta['production_countries'] = meta['production_countries'].fillna('
').apply(literal_eval).apply(lambda x: [i['name'] for i in x] if i
sinstance(x, list) else [])
```

```
meta['spoken_languages'] = meta['spoken_languages'].fillna('').app
ply(literal_eval).apply(lambda x: [i['name'] for i in x] if isinstanc
e(x, list) else [])
```

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

Let's take a final look at the data before moving to our next steps

```
meta.head()
```

```
meta['original_language'].drop_duplicates().shape[0]
```

```

lang_dframe = pd.DataFrame(meta['original_language'].value_counts())
lang_dframe['language'] = lang_df.index
lang_dframe.columns = ['number', 'language']
lang_dframe.head()

```

```

plt.figure(figsize=(12,5))
sns.barplot(x='language', y='number', data=lang_dframe.iloc[1:11])
plt.show()

```

```

def numeric_clean (x):
    try:
        return float(x)
    except:
        return np.nan

```

```

meta['popularity'] = meta['popularity'].apply(clean_numeric).astype(
'float')
meta['vote_count'] = meta['vote_count'].apply(clean_numeric).astype(
'float')
meta['vote_average'] = meta['vote_average'].apply(clean_numeric).ast
ype('float')

```

```

meta['popularity'].describe()

```

```

order_of_months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
order_of_days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

```

```

def get_monthOfYear(x):
    try:
        return order_of_months[int(str(x).split('-')[1]) - 1]
    except:
        return np.nan

```

```

def get_dayOfWeek(x):
    try:
        year, month, day = (int(i) for i in x.split('-'))
        ans = datetime.date(year, month, day).weekday()
        return order_of_days[ans]
    except:
        return np.nan

```

```

meta['day'] = meta['release_date'].apply(get_day)
meta['month'] = meta['release_date'].apply(get_month)

```

```
plt.figure(figsize=(12,6))
plt.title("Number of Movies released in a particular month.")
sns.countplot(x='month', data=meta, order=month_order)
```

```
mean_month = pd.DataFrame(meta[meta['revenue'] > 1e8].groupby('month')
['revenue'].mean())
mean_month['month'] = mean_month.index
plt.figure(figsize=(12,6))
plt.title("Average Gross by the Month for Blockbuster Movies")
sns.barplot(x='month', y='revenue', data=mean_month, order= order_of_
_month)
```

```
year_count = meta.groupby('year')['title'].count()
plt.figure(figsize=(18,5))
year_count.plot()
```

```
months = {'Jan': 1, 'Feb': 2, 'Mar': 3, 'Apr': 4, 'May': 5, 'Jun': 6
, 'Jul': 7, 'Aug': 8, 'Sep': 9, 'Oct': 10, 'Nov': 11, 'Dec': 12}
```

```
dframe_21 = meta.copy()
dframe_21['year'] = dframe_21[df_21['year'] != 'NaT']['year'].astype
(int)
dframe_21 = dframe_21[dframe_21['year'] >=2000]
hmap_21thYear = pd.pivot_table(data=dframe_21, index='month', column
s='year', aggfunc='count', values='title')
hmap_21thYear = hmap_21thYear.fillna(0)
sns.set(font_scale=1)
f, ax = plt.subplots(figsize=(16, 8))
sns.heatmap(hmap_21thYear, annot=True, linewidths=.5, ax=ax, fmt='n'
, yticklabels=order_of_months)
```

```
meta['runtime'] = meta['runtime'].astype('float')
```

```
plt.figure(figsize=(12,6))
sns.distplot(meta[(meta['runtime'] < 300) & (meta['runtime'] > 0)]['
runtime'])
```

The Simple Popularity Based Recommender

$$\text{Weighted Rank (WR)} = (v \div (v+m)) \times R + (m \div (v+m)) \times C$$

where,

R = average for the movie (mean) = (Rating)
v = number of votes for the movie = (votes)
m = minimum votes required to be listed in the Top 250
C = the mean vote across the whole report

```
count_votes = meta[meta['vote_count'].notnull()]['vote_count'].astype('int')
average_votes = meta[meta['vote_average'].notnull()]['vote_average'].astype('int')

C = averages_votes.mean()
print('The Mean value of the voting averages= ',C)
m = count_votes.quantile(0.96)
print('The minimum vote count for a movie to consider= ',m)
```

Creating the qualified database- upon whom we shall perform the next estimations

```
qualified = meta[(meta['vote_count'] >= m) & (meta['vote_count'].notnull()) & (meta['vote_average'].notnull())][['title', 'year', 'vote_count', 'vote_average', 'popularity', 'genres']]
qualified['vote_count'] = qualified['vote_count'].astype('int')
qualified['vote_average'] = qualified['vote_average'].astype('int')
print('The structure of the qualified database is= ',qualified.shape)
```

```
def rating_weighted(x):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
```

```
qualified['wr'] = qualified.apply(rating_weighted, axis=1)
qualified = qualified.sort_values('wr', ascending=False).head(250)
```

Top Movies

```
qualified.head(15)
```

	vote_average	popularity	\
15480	8	29.108149	
12481	8	123.167259	
22879	8	32.213481	
2843	8	63.869599	
4863	8	32.070725	
292	8	140.950236	
314	8	51.645403	

```

7000      8    29.324358
351       8    48.307194
5814      8    29.423537
256       8    42.149697
1225      8    25.778509
834       8    41.109264
1154      8    19.470959
46        8    18.457430

```

Pivoting down the entire dataset based on genres

```

s = meta.apply(lambda x: pd.Series(x['genres']),axis=1).stack().reset_index(level=1, drop=True)
s.name = 'genre'
g_md = meta.drop('genres', axis=1).join(s)
g_md.head(8)

```

```

   vote_average  vote_count  year  day month  genre
0             7.7      5415.0  1995  NaN  Oct  Animation
0             7.7      5415.0  1995  NaN  Oct   Comedy
0             7.7      5415.0  1995  NaN  Oct   Family
1             6.9      2413.0  1995  NaN  Dec  Adventure
1             6.9      2413.0  1995  NaN  Dec   Fantasy
1             6.9      2413.0  1995  NaN  Dec   Family
2             6.5         92.0  1995  NaN  Dec   Romance
2             6.5         92.0  1995  NaN  Dec   Comedy

```

```

def chart_builder(genre, percentile=0.90):
    dfr = g_md [g_md ['genre'] == genre]
    count_votes = dfr[dfr['vote_count'].notnull()]['vote_count'].astype('int')
    average_votes = df[df['vote_average'].notnull()]['vote_average'].astype('int')

```

```

    C = average_votes.mean()
    m = count_votes.quantile(percentile)

```

```

    qual = dfr[(dfr['vote_count'] >= m) & (dfr['vote_count'].notnull()) & (dfr['vote_average'].notnull())][['title', 'year', 'vote_count', 'vote_average', 'popularity']]
    qual['vote_count'] = qual['vote_count'].astype('int')
    qual['vote_average'] = qual['vote_average'].astype('int')

```

```

    qual['wr'] = qual.apply(lambda x: (x['vote_count']/(x['vote_count']+m) * x['vote_average']) + (m/(m+x['vote_count']) * C), axis=1)

```



```

qualif = qual.sort_values('wr', ascending=False).head(250)
return qualif

```

Top Horror Movies

```
chart_builder('Horror').head(15)
```

unt \	title	year	vote_co
1213	The Shining	1980	3
890			
1176	Psycho	1960	2
405			
1171	Alien	1979	4
564			
41492	Split	2016	4
461			
14236	Zombieland	2009	3
655			
1158	Aliens	1986	3
282			
21276	The Conjuring	2013	3
169			
42169	Get Out	2017	2
978			
1338	Jaws	1975	2
628			
8147	Shaun of the Dead	2004	2
479			
8230	Saw	2004	2
255			
1888	The Exorcist	1973	2
046			
39097	The Conjuring 2	2016	2
018			
6353	28 Days Later	2002	1
816			
12277	Sweeney Todd: The Demon Barber of Fleet Street	2007	1
745			

Top Romantic Movies

```
Chart_builder('Romance').head(15)
```

\	title	year	vote_count	vote_average
10309	Dilwale Dulhania Le Jayenge	1995	661	9
351	Forrest Gump	1994	8147	8

876	Vertigo	1958	1162	8
40251	Your Name.	2016	1030	8
883	Some Like It Hot	1959	835	8
1132	Cinema Paradiso	1988	834	8
19901	Paperman	2012	734	8
37863	Sing Street	2016	669	8
882	The Apartment	1960	498	8
38718	The Handmaiden	2016	453	8
3189	City Lights	1931	444	8
24886	The Way He Looks	2014	262	8
1639	Titanic	1997	7770	7
19731	Silver Linings Playbook	2012	4840	7
40882	La La Land	2016	4745	7

Content Based Recommender

```
lk_small = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/links_small.csv')
lk_small = lk_small[lk_small['tmdbId'].notnull()]['tmdbId'].astype('int')
lk_small.head()
```

```
meta = meta.drop([19730, 29503, 35587])
meta['id'] = meta['id'].astype('int')
smdes = meta[meta['id'].isin(lk_small)]
smdes.shape
```

Movie Description Based Recommender

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

We shall be using Tfidfvectorizer for this step.

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

```

tfidf_matrix.shape

cosine_simil = linear_kernel(tfidf_matrix, tfidf_matrix)
cosine_simil[0]

smdes = smdes.reset_index()
titles1 = smdes['title']
indices1 = pd.Series(smdes.index, index=smdes['title'])

def get_rec(title):
    idx = indices1[title]
    similar_scores = list(enumerate(cosine_simil[idx]))
    similar_scores = sorted(similar_scores, key=lambda x: x[1], reverse=True)
    similar_scores = similar_scores[1:21]
    movie_hasIndices = [i[0] for i in similar_scores]
    return titles.iloc[movie_hasIndices]

```

```

movie='3 Idiots'
print("Description of the Movie: ", movie)
print('-----')
print(smdes[smdes['title']==movie]['overview'])

```

```

Description of the Movie: 3 Idiots
-----
-

```

```

get_rec('3 Idiots').head(20)

```

```

2336          Ferris Bueller's Day Off
8161          Student of the Year
262           Outbreak
2658          The Next Best Thing
4378  Come Back to the 5 & Dime, Jimmy Dean, Jimmy Dean
1861          Enemy of the State
3098          Bring It On
7866          Contagion
4543          What a Girl Wants
5373          College
149           Hackers
3875          The Party
5462          Our Hospitality
5466          Overboard
3496          Suspiria
2882          Loser
308           The Baby-Sitters Club
1937          Patch Adams
3277          Extreme Prejudice

```

```
5870 The Lion King 2: Simba's Pride
Name: title, dtype: object
```

```
movie='The Dark Knight'
print("Description of the Movie: ", movie)
print('-----')
print(smdes[smdes['title']==movie]['overview'])
```

```
Description of the Movie: The Dark Knight
-----
6900 Batman raises the stakes in his war on crime. ...
Name: overview, dtype: object
```

```
get_rec('The Dark Knight').head(20)
```

Metadata Based Recommender

```
cred = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/credits.csv')
keywrd = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/keywords.csv')
```

```
keywrd.head()
```

```
cred.head()
```

```
keywrd['id'] = keywrd['id'].astype('int')
cred['id'] = cred['id'].astype('int')
meta['id'] = meta['id'].astype('int')
meta.shape
```

```
meta = meta.merge(cred, on='id')
meta = meta.merge(keywrd, on='id')
```

```
smd2 = meta[meta['id'].isin(lk_small)]
smd2.shape
```

```
smd2['cast'] = smd2['cast'].apply(literal_eval)
smd2['crew'] = smd2['crew'].apply(literal_eval)
smd2['keywords'] = smd2['keywords'].apply(literal_eval)
smd2['cast_size'] = smd2['cast'].apply(lambda x: len(x))
smd2['crew_size'] = smd2['crew'].apply(lambda x: len(x))
```

```

def get_dir(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan

smd2['director'] = smd2['crew'].apply(get_dir)

smd2['cast'] = smd2['cast'].apply(lambda x: [i['name'] for i in x] if
    isinstance(x, list) else [])
smd2['cast'] = smd2['cast'].apply(lambda x: x[:4] if len(x) >=4 else
    x)

smd2['keywords'] = smd2['keywords'].apply(lambda x: [i['name'] for i
    in x] if isinstance(x, list) else [])

smd2['cast'] = smd2['cast'].apply(lambda x: [str.lower(i.replace(" ",
    "")) for i in x])

smd2['director'] = smd2['director'].astype('str').apply(lambda x: str
    .lower(x.replace(" ", "")))
smd2['director'] = smd2['director'].apply(lambda x: [x,x,x])

```

Keywords

```

sm = smd2.apply(lambda x: pd.Series(x['keywords']),axis=1).stack().r
    eset_index(level=1, drop=True)
sm.name = 'keyword'

sm = sm.value_counts()
sm[:10]

sm = sm[s > 1]

stemmer = SnowballStemmer('english')
stemmer.stem('sportingly')

{"type":"string"}

```

```

def keyword_filter(x):
    words = []
    for i in x:
        if i in s:
            words.append(i)
    return words

```

```

smd2['keywords'] = smd2['keywords'].apply(keyword_filter)
smd2['keywords'] = smd2['keywords'].apply(lambda x: [stemmer.stem(i)
for i in x])
smd2['keywords'] = smd2['keywords'].apply(lambda x: [str.lower(i.replace(" ", "")) for i in x])

smd2['soup'] = smd2['keywords'] + smd2['cast'] + smd2['director'] +
smd2['genres']
smd2['soup'] = smd2['soup'].apply(lambda x: ' '.join(x))

```

```

cnt = CountVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0,
top_words='english')
cnt_matrix = cnt.fit_transform(smd2['soup'])

cosine_similar = cosine_similar (cnt_matrix, cnt_matrix)

smd2 = smd2.reset_index()
title2 = smd2['title']
indice2 = pd.Series(smd2.index, index=smd2['title'])

get_rec('The Dark Knight').head(15)

```

```

get_rec('Mrs. Doubtfire').head(15)

```

```

164                               Nine Months
2553                               Bicentennial Man
519                               Home Alone
2388                Home Alone 2: Lost in New York
1958                               Stepmom
7538    Percy Jackson & the Olympians: The Lightning T...
8996                               Pixels
7377                I Love You, Beth Cooper
1708                Adventures in Babysitting
3840                Harry Potter and the Philosopher's Stone
4366                Harry Potter and the Chamber of Secrets
6357                               Rent
2833                               Parenthood
4378                               Houseboat
6057                               Fat Albert

```

```

Name: title, dtype: object

```

```

def enhanced_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_simil[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
)
sim_scores = sim_scores[1:26]

```

```

movie_indices = [i[0] for i in sim_scores]

movies = smd2.iloc[movie_indices][['title', 'vote_count', 'vote_
average', 'year']]
count_votes = movies[movies['vote_count'].notnull()][ 'vote_count
'].astype('int')
average_votes = movies[movies['vote_average'].notnull()][ 'vote_a
verage'].astype('int')

C = average_votes.mean()
m = count_votes.quantile(0.50)

qualif = movies[(movies['vote_count'] >= m) & (movies['vote_count'
].notnull()) & (movies['vote_average'].notnull())]
qualif['vote_count'] = qualif['vote_count'].astype('int')
qualif['vote_average'] = qualif['vote_average'].astype('int')
qualif['wr'] = qualif.apply(rating_weighted, axis=1)
qualif = qualif.sort_values('wr', ascending=False).head(10)
return qualif

```

```

enhanced_recommendations('The Dark Knight')

```

	title	vote_count	vote_average	year	w
r					
7648	Inception	14075	8	2010	7.89156
8					
8613	Interstellar	11187	8	2014	7.86494
8					
6623	The Prestige	4510	8	2006	7.68767
1					
3381	Memento	4168	8	2000	7.66515
8					
8031	The Dark Knight Rises	9263	7	2012	6.89714
4					
6218	Batman Begins	7511	7	2005	6.87486
3					
1134	Batman Returns	1706	6	1992	5.80924
6					
4145	Insomnia	1181	6	2002	5.75226
8					
132	Batman Forever	1529	5	1995	5.06706
6					
9162	London Has Fallen	1656	5	2016	5.06325
1					

```
enhanced_recommendations('Mrs. Doubtfire')
```

	title	vote_count
\		
3840	Harry Potter and the Philosopher's Stone	7188
4366	Harry Potter and the Chamber of Secrets	5966
519	Home Alone	2487
2388	Home Alone 2: Lost in New York	2459
7538	Percy Jackson & the Olympians: The Lightning T...	2079
2553	Bicentennial Man	998
1958	Stepmom	286
837	Homeward Bound: The Incredible Journey	218
2833	Parenthood	177
1708	Adventures in Babysitting	169

Collaborative Filtering Application

```
reader = Reader()
ratings = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/rating
s_small.csv')
ratings.head()
```

```
rating_counts = pd.DataFrame(ratings['rating'].value_counts()).reset
_index()
rating_counts.columns = ['Labels', 'Ratings']
rating_counts
```

```
sns.set_style('whitegrid')
sns.set(font_scale=1.5)
%matplotlib inline
```

```
sns.distplot(ratings['rating'].fillna(ratings['rating'].median()))
<Axes: xlabel='rating', ylabel='Density'>
```

```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(15,7))
sns.countplot(rating_counts['Ratings'], ax=ax1)
ax1.set_xlabel('Rating Distribution', fontsize=10)
ax1.set_ylabel('Count', fontsize=10)
```

```
explode = (0.1, 0, 0, 0, 0, 0, 0, 0, 0.1, 0)
ax2.pie(rating_counts["Ratings"], explode=explode, labels=rating_cou
nts.Labels, autopct='%1.1f%%', shadow=True, startangle=0)
```



```
ax2.axis('equal')
plt.title("Rating Ratio")
plt.show()
```

```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']]
, reader)
```

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

```
trainset = data.build_full_trainset()
svd.fit(trainset)
```

```
user_rating=pd.merge(ratings,meta,left_on='movieId',right_on='id',how='inner')
user_ratings_final=user_rating[['userId', 'movieId', 'rating','original_title']]
user_ratings=user_ratings_final.sort_values(by='userId')
user_ratings.head()
```

	userId	movieId	rating	original_title
0	1	1371	2.5	Rocky III
93	1	2105	4.0	American Pie
140	1	2193	2.0	My Tutor
47	1	1405	1.0	Greed
182	1	2294	2.0	Jay and Silent Bob Strike Back

```
user_ratings[user_ratings['userId'] == 7]
```

	userId	movieId	rating	original_title
18645	7	671	4.0	Harry Potter and the Philosopher's Stone
4200	7	500	3.0	Reservoir Dogs
18739	7	745	5.0	The Sixth Sense
18801	7	780	3.0	La passion de Jeanne d'Arc
11487	7	1376	3.0	Sweet Sixteen
48	7	1405	5.0	Greed
8557	7	260	5.0	The 39 Steps

4770	7	539	3.0	Ps
ycho				
3242	7	377	3.0	A Nightmare on Elm St
reet				
19019	7	786	2.0	Almost Fa
mous				
2039	7	272	3.0	Batman Be
gins				
5345	7	588	4.0	Silent
Hill				
19360	7	1408	1.0	Cutthroat Is
land				
8395	7	141	4.0	Donnie D
arko				
6410	7	318	5.0	The Million Dollar H
otel				
2	7	1371	3.0	Rocky
III				
8345	7	112	4.0	Italiensk for begyn
dere				
19290	7	1394	3.0	Nostal
ghia				
4906	7	551	4.0	The Poseidon Adven
ture				
11415	7	1374	4.0	Rock
y IV				
19088	7	924	4.0	Dawn of the
Dead				
19236	7	1375	3.0	Roc
ky V				
19211	7	1373	2.0	The Discovery of He
aven				
11368	7	1372	3.0	Blood Dia
mond				
3042	7	364	3.0	Batman Ret
urns				
11169	7	1278	3.0	The Drea
mers				
18620	7	534	4.0	Terminator Salva
tion				
6722	7	595	3.0	To Kill a Mocking
bird				
18678	7	708	3.0	The Living Dayli
ghts				
3866	7	480	4.0	Monsoon Wed
ding				
14416	7	104	3.0	Lola r
ennt				
9707	7	594	4.0	The Term

inal					
18211	7	21	3.0		The Endless Su
mmer					
18306	7	198	2.0		To Be or Not t
o Be					
5558	7	590	4.0		The H
ours					
8976	7	380	4.0		Rain
Man					
18349	7	207	3.0		Dead Poets Soc
iety					
5760	7	592	3.0		The Conversa
tion					
8862	7	329	3.0		Jurassic
Park					
18377	7	316	2.0		Halbe Tr
eppe					
10358	7	1073	3.0		Arlington
Road					
10638	7	1125	3.0		Dreamg
irls					
9561	7	541	4.0		The Man with the Golden
Arm					
457	7	110	5.0		Trois couleurs : R
ouge					
18572	7	345	3.0		Eyes Wide
Shut					
18522	7	333	3.0		Bollywood/Holly
wood					

```
movie1=meta['original_title']=='The Conjuring'
meta[movie1][['original_title','id']]
```

```
svd.predict(7, 138843, 3)
```

```
Prediction(uid=7, iid=138843, r_ui=3, est=3.2986816354722617, details={'was_impossible': False})
```

A Hybrid Recommender

```
def convert_int(x):
    try:
        return int(x)
    except:
        return np.nan
```

```
id_mp = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/links_sm
all.csv')[['movieId', 'tmdbId']]
id_mp['tmdbId'] = id_mp['tmdbId'].apply(convert_int)
id_mp.columns = ['movieId', 'id']
id_mp = id_mp.merge(smd2[['title', 'id']], on='id').set_index('title
')
```

```
indices_mp = id_mp.set_index('id')
```

```
def recommend_my_movie(userId, title):
    idx = indices1[title]
    tmdbId = id_mp.loc[title]['id']
    movie_id = id_mp.loc[title]['movieId']
    similar_scores = list(enumerate(cosine_simil[int(idx)]))
    similar_scores = sorted(similar_scores, key=lambda x: x[1], reve
rse=True)
    similar_scores = similar_scores[1:26]
    movie_hasIndices = [i[0] for i in sim_scores]
    movies = smd2.iloc[movie_hasIndices][['title', 'vote_count', 'vo
te_average', 'year', 'id']]
    movies['est'] = movies['id'].apply(lambda x: svd.predict(userId,
indices_mp.loc[x]['movieId']).est)
    movies = movies.sort_values('est', ascending=False)
    return movies.head(10)
```

```
recommend_my_movie(7, "Inception")
```

id \	title	vote_count	vote_average	year
3381	Memento	4168.0	8.1	2000
77				
6981	The Dark Knight	12269.0	8.3	2008
155				
6623	The Prestige	4510.0	8.0	2006
1124				
6218	Batman Begins	7511.0	7.5	2005
272				

8613	Interstellar	11187.0	8.1	2014
157336				
8031	The Dark Knight Rises	9263.0	7.6	2012
49026				
4173	Minority Report	2663.0	7.1	2002
180				
6640	Déjà Vu	1519.0	6.6	2006
7551				
4145	Insomnia	1181.0	6.8	2002
320				
5580	The Three Lives of Thomasina	12.0	6.8	1963
15081				

	est
3381	4.083157
6981	4.036027
6623	3.932428
6218	3.810600
8613	3.640355
8031	3.514015
4173	3.477439
6640	3.421840
4145	3.397775
5580	3.372996

recommend_my_movie(25, "Inception")

	title	vote_count	vote_average	year	id
est					
6623	The Prestige	4510.0	8.0	2006	1124
3.994742					
6981	The Dark Knight	12269.0	8.3	2008	155
3.691721					
3381	Memento	4168.0	8.1	2000	77
3.659017					
8613	Interstellar	11187.0	8.1	2014	157336
3.608549					
6218	Batman Begins	7511.0	7.5	2005	272
3.499123					
4173	Minority Report	2663.0	7.1	2002	180
3.445458					
8031	The Dark Knight Rises	9263.0	7.6	2012	49026
3.302625					
7828	I Am Number Four	1606.0	5.9	2011	46529
3.264680					
2085	Following	363.0	7.2	1998	11660
3.228714					

4.7. Experimental Recommendations

This section depicts our experimental results which are the desired recommendations from a movie recommendation system. We have shared the output screenshots of the personalized recommendations provided by the different recommendation algorithms used, shown in Figures 6, 7, 8, 9 and 10 which will help us to easily understand and compare all algorithms used to design a movie recommendation system:

Top Romantic Movies

```
build_chart('Romance').head(15)
```

	title	year	vote_count	vote_average	popularity	wr
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457024	8.271289
351	Forrest Gump	1994	8147	8	48.307194	7.947552
876	Vertigo	1958	1162	8	18.208220	7.672104
40251	Your Name.	2016	1030	8	34.461252	7.635975
883	Some Like It Hot	1959	835	8	11.845107	7.565203
1132	Cinema Paradiso	1988	834	8	14.177005	7.564769
19901	Paperman	2012	734	8	7.198633	7.516517
37863	Sing Street	2016	669	8	10.672862	7.478971
882	The Apartment	1960	498	8	11.994281	7.345193
38718	The Handmaiden	2016	453	8	16.727405	7.297743
3189	City Lights	1931	444	8	10.891524	7.287416
24886	The Way He Looks	2014	262	8	5.711274	6.985810
1639	Titanic	1997	7770	7	26.889070	6.966226
19731	Silver Linings Playbook	2012	4840	7	14.488111	6.946465
40882	La La Land	2016	4745	7	19.681686	6.945430

Fig. 6. Popularity-based Recommendations

Popularity-based movie recommendations lack practicality because they overlook the unique and user specific preferences of individual users, are very limited in recommendations as it follows the general trend and faces difficulty in handling new and unknown movies.

```

▶ get_recommendations('The Dark Knight').head(20)
7931          The Dark Knight Rises
132           Batman Forever
1113          Batman Returns
8227  Batman: The Dark Knight Returns, Part 2
7565          Batman: Under the Red Hood
524           Batman
7901          Batman: Year One
2579          Batman: Mask of the Phantasm
2696           JFK
8165  Batman: The Dark Knight Returns, Part 1
6144          Batman Begins
7933  Sherlock Holmes: A Game of Shadows
5511           To End All Wars
4489           Q & A
7344          Law Abiding Citizen
7242  The File on Thelma Jordan
3537          Criminal Law
2893          Flying Tigers
1135  Night Falls on Manhattan
8680  The Young Savages
Name: title, dtype: object

```

Fig. 7. Description-based Content Filtering Recommendations

```

✓ Os ▶ get_recommendations('Mrs. Doubtfire').head(15)
164          Nine Months
2553         Bicentennial Man
519          Home Alone
2388  Home Alone 2: Lost in New York
1958         Stepmom
7538  Percy Jackson & the Olympians: The Lightning T...
8996         Pixels
7377         I Love You, Beth Cooper
1708         Adventures in Babysitting
3840  Harry Potter and the Philosopher's Stone
4366  Harry Potter and the Chamber of Secrets
6357         Rent
2833         Parenthood
4378         Houseboat
6057         Fat Albert
Name: title, dtype: object

```

Fig. 8. Movie metadata-based Content Filtering Recommendations

Let us choose User no. 7. First lets take a look into the movies he/she has watched.

```
user_ratings[user_ratings['userId'] == 7]
```

	userId	movieId	rating	original_title
18645	7	671	4.0	Harry Potter and the Philosopher's Stone
4200	7	500	3.0	Reservoir Dogs
18739	7	745	5.0	The Sixth Sense
18801	7	780	3.0	La passion de Jeanne d'Arc
11487	7	1376	3.0	Sweet Sixteen
48	7	1405	5.0	Greed
8557	7	260	5.0	The 39 Steps
4770	7	539	3.0	Psycho
3242	7	377	3.0	A Nightmare on Elm Street
19019	7	786	2.0	Almost Famous
2039	7	272	3.0	Batman Begins
5345	7	588	4.0	Silent Hill
19360	7	1408	1.0	Cutthroat Island
8395	7	141	4.0	Donnie Darko
6410	7	318	5.0	The Million Dollar Hotel
2	7	1371	3.0	Rocky III
8345	7	112	4.0	Italiensk for begyndere
19290	7	1394	3.0	Nostalgia

Fig. 9. Collaborative Filtering Recommendations

Now, Figure 10 combines the recommendations of the previously displayed recommendation systems and gives us their hybrid recommendations.

```
recommend_my_movie(7, "Inception")
```

	title	vote_count	vote_average	year	id	est
3381	Memento	4168.0	8.1	2000	77	4.083157
6981	The Dark Knight	12269.0	8.3	2008	155	4.036027
6623	The Prestige	4510.0	8.0	2006	1124	3.932428
6218	Batman Begins	7511.0	7.5	2005	272	3.810600
8613	Interstellar	11187.0	8.1	2014	157336	3.640355
8031	The Dark Knight Rises	9263.0	7.6	2012	49026	3.514015
4173	Minority Report	2663.0	7.1	2002	180	3.477439
6640	Déjà Vu	1519.0	6.6	2006	7551	3.421840
4145	Insomnia	1181.0	6.8	2002	320	3.397775
5580	The Three Lives of Thomasina	12.0	6.8	1963	15081	3.372996

Fig. 10. Hybrid Filtering Recommendations

CHAPTER 5

RESULTS

First, we have visualized the distribution of the movies in their respective genres according to the input dataset in Figure 11.

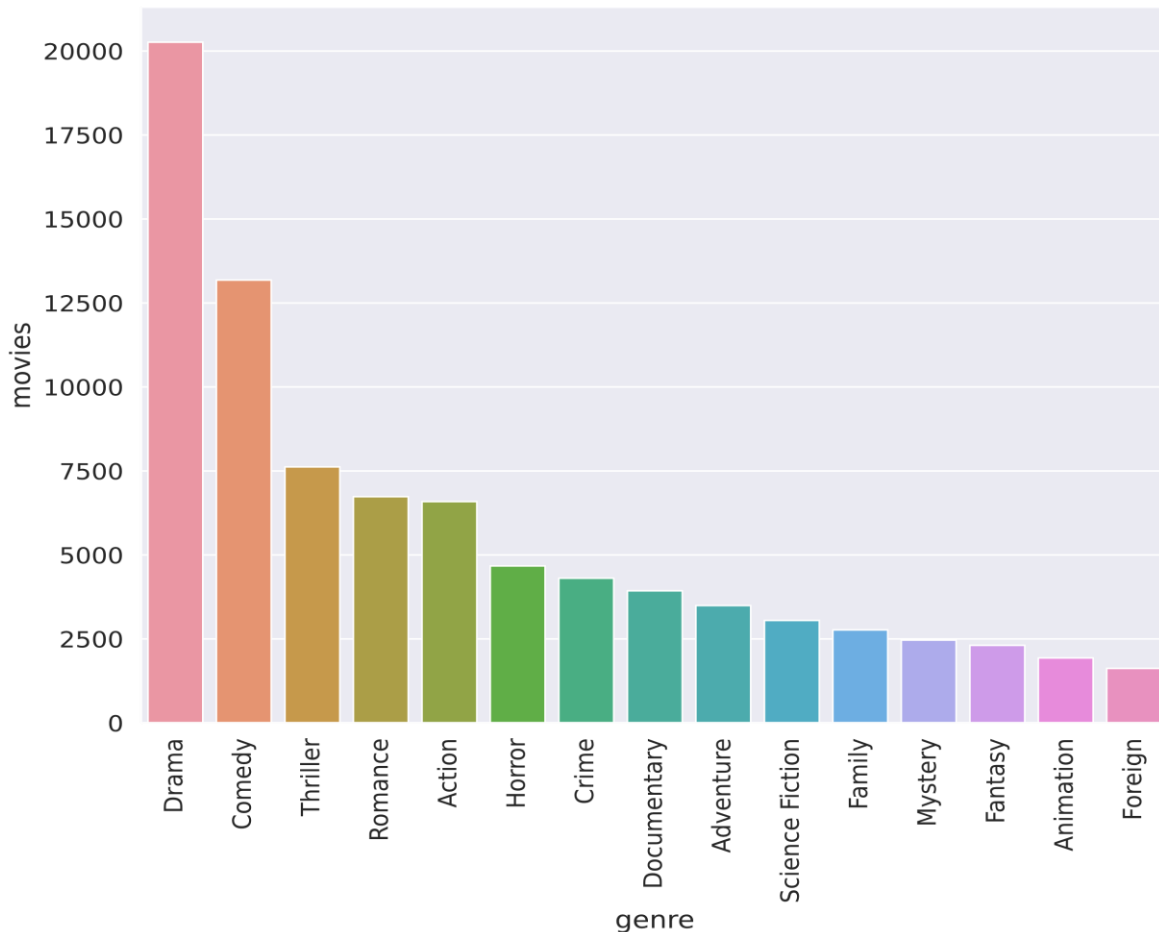


Fig.11. Distribution of movies in their respective genre

Figure 11 clearly shows that the Drama genre holds the lead, accounting for the highest number of movies. In close succession, are the Comedy, Thriller, Romance, Action, and other genres, showcasing the diverse range of films within our dataset.

Then we have visualized the distribution of movies generating the highest gross revenues and which month do such movies are more likely to release using the input dataset in Figure 12.

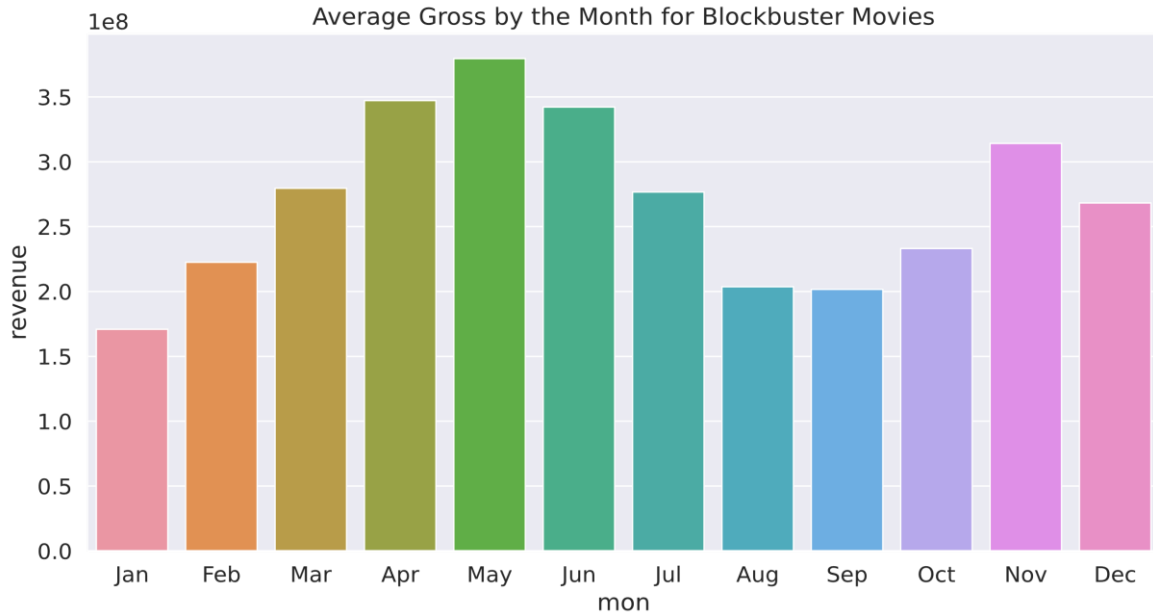


Fig.12. Months for Blockbuster movies

From Figure 12, it is clearly evident that according to our input dataset, the summer months of April, May, and June witness a surge in the average gross revenue of high-grossing movies, indicating a behavioural pattern in the audience. This sudden boost can be attributed to the well planned and strategic release of blockbuster films during this time, taking full advantage of the vacations and holiday season and capitalizing individuals inclined to have more disposable time and income to seek quality entertainment.

Then we have represented the distribution of the release of movies in a heatmap in figure 13 and tried to find out which months and years have been hot and cold for the movie releases.

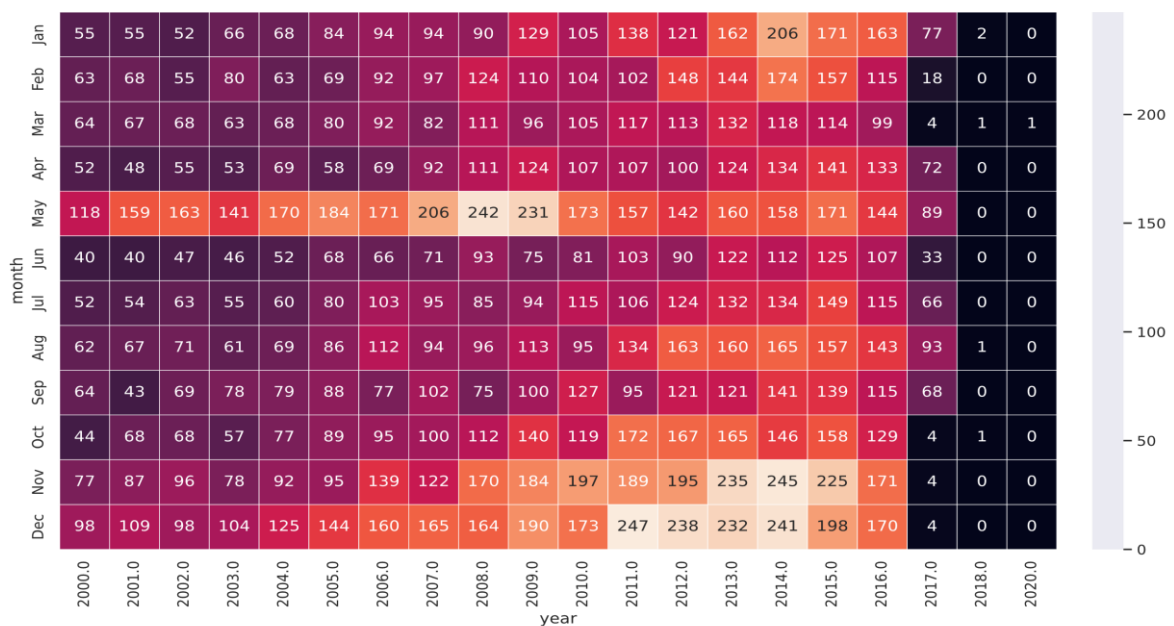


Fig. 13. Distribution of Movies based on their Genre

Figure 13 clearly shows that the month of May has been one of the most favoured times since early 2000 in which filmmakers prefer to release majority of the movies. And overtime the rest of the months have also become a good time to release movies which could be a result of the growth and development in the film industry in terms of technology and viewership. But it is also evident that there has been a sharp decline in movie releases in the year of 2020 which could be a result of COVID restrictions.

The visualization in Figure 14 clearly highlights the directors who have directed the most number of successful, highest-grossing, and revenue-generating movies.

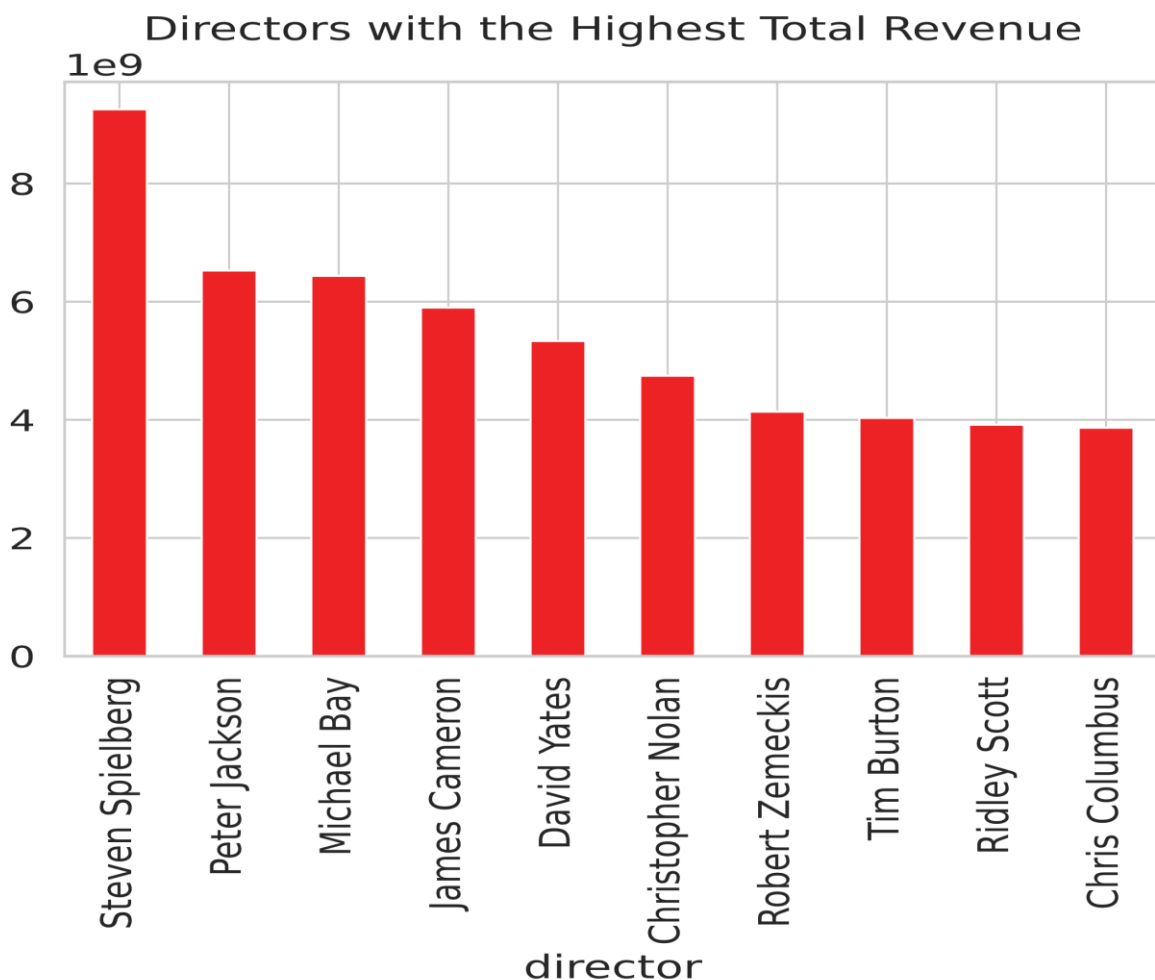


Fig. 14. Directors who have directed the highest Revenue-Generating Movies

Then we have visualized the change in the runtime of movies in Figure 15.

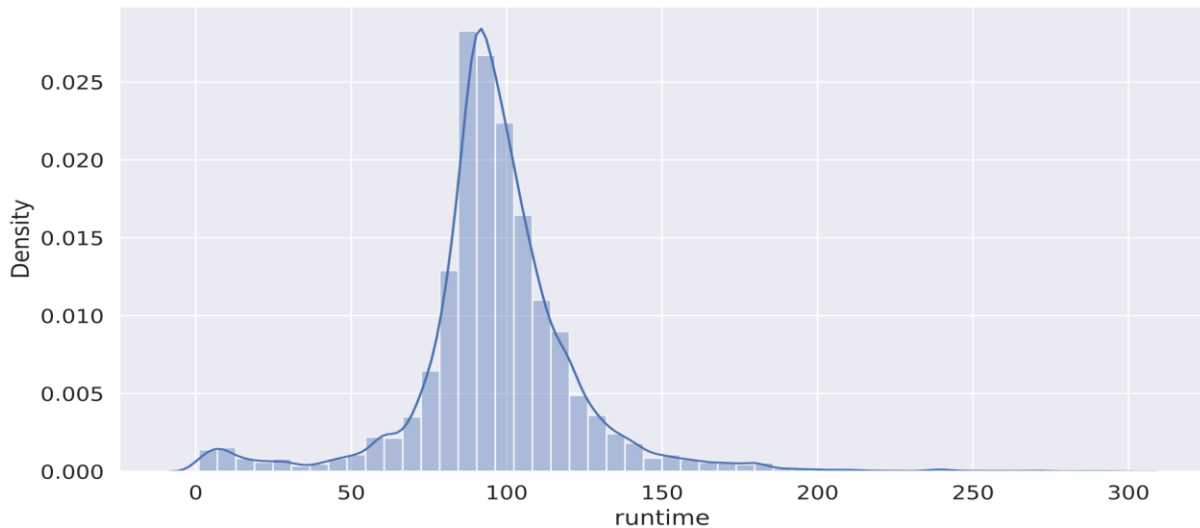


Fig.15. Trend in Movie Runtime

From Figure 15, we can note that the runtime of movies has been trending upwards throughout the years.

Our analysis involved applying a range of Machine Learning Algorithms to the input data set and training multiple models resulting in a range of models, each with their own RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) scores. The resulting RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) score can only be calculated for collaborative filtering, and it has been tabulated in Table II:

TABLE II
ACCURACY RESULTS OF COLLABORATIVE-FILTERING
RECOMMENDATION SYSTEM ALGORITHM.

Machine Learning Model	RMSE Score	MAE Score
Collaborative Filtering	0.8944	0.6903

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Through our work, we have tried to solve one of the most concerning real-world business problems which is helpful to the online media streaming community to gain more customers, scale their businesses, and thrive in this highly competitive market. In our work, we applied various recommendation system machine learning filtering algorithms, which were trained and tested for the input dataset and later evaluated and compared their recommendations. The most widely used algorithm among all our used algorithms is the hybrid algorithm, which is used by major online streaming media companies. We were successfully able to understand the differences in the recommendation capability of the various used recommendation algorithms.


Our future plans involve testing out a wider range of machine learning algorithms to find out if there exists a better option that is more accurate than the current model. We also have plans to integrate our framework with a web application and provide a user interface in order to provide a better user experience and which will make it more easily and readily accessible to more and more people. We can also diversify our recommendation model for other real-world problems in the healthcare sector, fashion sector, e-commerce sector, etc. Ultimately, our goal is to build recommendation systems that not only help online media streaming platforms succeed but also prioritize user satisfaction, trust, and ethical considerations. As we move forward, we envision a future where our work will continue to evolve and adapt to the dynamic landscape of recommendation systems by exploring advanced techniques such as sequential recommendations, multimodal recommendations, online learning, and real-time updates, which will help us to provide users with more accurate, personalized, and engaging recommendations. By combining cutting-edge research, industry collaboration, and a user-centric approach, we aim to contribute to the growth and success of the online media streaming community and enhance the overall streaming experience for users worldwide. We could also improve our current recommendation system by applying deep learning and neural networks to add features like image detection and sentiment analysis, which will definitely provide better recommendations with higher accuracy. We also aim to enhance the level of personalization in our recommendation system by incorporating features like user feedback and behavior analysis. By considering individual user preferences, viewing patterns, and feedback, we can tailor recommendations to better suit each user's unique tastes and interests. We also plan to incorporate contextual information such as time of day, location, device, and user demographics. Through these advancements, we anticipate delivering even better results to users, ensuring their needs and preferences are met more effectively.

REFERENCES

- [1] T. Keerthana, T. Bhavani, N. Suma Priya, V. Sai Prathyusha, K. Santhi Sri, "FLIPKART PRODUCT RECOMMENDATION SYSTEM". Journal of Engineering Sciences(JES), Vol 11, Issue 4, April/ 2020, ISSN NO: 0377-9254.
- [2] JH (Janghyun) Baek, John Tsai, Justin Shamoun, Muriel Marable, Ying Cui, "Amazon Recommender System". In Data Science Engineering Master of Advanced Study (DSE MAS) Capstone Projects, UC San Diego Library Digital Collections.
- [3] Mohammad R. Rezaei, "Amazon Product Recommender System". University of Toronto, Toronto, ON M5S 3G9, Canada.
- [4] Rohit Dwivedi, Abhineet Anand, Dr. Prashant Johri, Arpit Banerjee, NK Gaur, "Product Based Recommendation System On Amazon Data". Research Gate, 2021.
- [5] Zan Huang, Wingyan Chung, Thian-Huat Ong, Hscinchun Chen, "A Graph-based Recommender System for Digital Library". JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, July 2002.
- [6] Shuai Zhang, Lina Yao, Aixin Sun, Yi Tay, "Deep Learning – Based Recommender System". A Survey and New Perspectives. ACM Comput. Surv.1, 1, Article 1, July 2018.
- [7] Greg Linden, Brent Smith, and Jeremy York, "Amazon.com Recommendations Item-to-Item Collaborative Filtering". Industry Report, Amazon.com.
- [8] Md Zaid Ahmed, Abhay Singh, Abir Paul, Sayantani Ghosh, Avijit Kumar Chaudhuri, "Amazon Product Recommendation System". International Journal of Advanced Research in Computer and Communication Engineering, Vol.11, Issue 3, March 2022.
- [9] Ashrita Kashyap, Sunita. B, Sneha Srivastava, Aishwarya. PH, Anup Jung Shah, "A Movie Recommender System: MOVREC using Machine Learning Techniques", International Journal of Engineering Science and Computing (IJESC), Vol.10, Issue-6, 2020.
- [10] Yeole Madhavi B., Rokade Monika D., Khatal Sunil, "Movie Recommendation System using Content-based Filtering", International Journal of Advanced Research and Innovation Ideas in Education (IJARIIE), Vol.7, Issue-4, 2021, IJARIIE-ISSN(O)-2395-4396.
- [11] Rahul Pradhan, Ashish Chandra Swami, Akash Saxena, Vikram Rajpoot, "A Study on Movie Recommendations using Collaborative Filtering", International Conference on Advances in Materials Science, Communication and Microelectronics (ICAMCM), IOP Conf. Ser.: Mater. Sci. Eng. 2021.
- [12] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Proceedings of the ACM Conference on Recommender Systems (RecSys), vol. 2009, pp. 5-12, 2009.
- [13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," in Proceedings of the 10th International Conference on World Wide Web (WWW), vol. 2001, pp. 285-295, 2001.

- [14] R. Burke, K. Hammond, and B. Young, "Improving Recommendation Quality by Incorporating Social Networks," in Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI), vol. 2007, pp. 263-266, 2007.
- [15] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization," in Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS), vol. 2008, pp. 1257-1264, 2008.
- [16] J. Cheng, L. Ma, Y. Zhang, and Y. Zhou, "A Scalable Collaborative Filtering Framework based on Co-Clustering," in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), vol. 2012, pp. 241-249, 2012.
- [17] Y. Liu, J. Yang, and J. Zhu, "Multi-View Clustering for Movie Recommendation," in Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM), vol. 2012, pp. 2218-2222, 2012.
- [18] X. Huang, J. Li, and C. Tang, "Collaborative Filtering using Weighted Non-negative Matrix Factorization," in Proceedings of the IEEE International Conference on Data Mining (ICDM), vol. 2009, pp. 427-436, 2009.
- [19] S. Dutta and S. Sarkar, "Movie Recommender System Using Collaborative Filtering and Fuzzy Logic," in Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research (ICIC), December 2014.

PUBLICATION SCREENSHOT

Acceptance Notification 5th IEEE ICAC3N-23 & Registration: Paper ID 111 



Microsoft CMT <email@microsoft.com>

Thu, May 4, 10:47 AM (6 days ago)   

to me

Dear Prakhar Anand,
Galgotias University

Greetings from ICAC3N-23 ...!!!

Congratulations.....!!!!

On behalf of the 5th ICAC3N-23 Program Committee, we are delighted to inform you that the submission of "Paper ID- 111 " titled " A Hybrid Approach Towards a Movie Recommendation System " has been accepted for presentation and further publication with IEEE at the ICAC3N- 23 subject to incorporate the reviewers and editors comments in your final paper. All accepted papers will be submitted for inclusion into IEEE Xplore subject to meeting IEEE Xplore's scope and quality requirements.

For early registration benefit please complete your registration by clicking on the following Link: <https://forms.gle/8e6R4hho7CohnY17> on or before 10 May 2023.

Registration fee details are available @ <https://icac3n.in/register>.

You must incorporate following comments in your final paper submitted at the time of registration for consideration of publication with IEEE:

The topic chosen "A Hybrid Approach Towards a Movie Recommendation System" is interesting and relevant.
The formatting of paper is not proper. Formatting must be strictly as per template. Otherwise it will not be published.
Author list formatting is not proper. All authors information must be complete and should be in proper format and as per the sequence desired.
Citation of references within the content is not proper. Make only relevant citation. All references must be cited in content properly.
References are not in proper format. Format and assign number to the references properly.
An overview of paper is desired to eradicate typo and grammatical error.
Formatting and Quality of figures is poor. Use High quality images.
All figure and tables must be properly captioned and numbered as per IEEE conference template.
Add a comparison table in literature review section with the work already done in this filed.
Conclusion and result section needs to be improved and require better explanation.

Editor Comments/Note:

1. All figures and equations in the paper must be clear.
2. Final camera ready copy must be strictly in IEEE format available on conference website www.icac3n.in.
3. Transfer of E-copyright to IEEE and Presenting paper in conference is compulsory for publication of paper in IEEE.
4. If plagiarism is found at any stage in your accepted paper, the registration will be cancelled and paper will be rejected and the authors will be responsible for any consequences. Plagiarism must be less than 20% (checked through Turnitin).
5. Change in paper title, name of authors or affiliation of authors will not be allowed after registration of papers.
6. Violation of any of the above point may lead to rejection of your paper at any stage of publication.
7. Registration fee once paid will be non refundable.

If you have any query regarding registration process or face any problem in making online payment, you can Contact @ 8168268768 (Call) / 9467482983 (Whatsapp) or write us at icac3n23@gmail.com.

Regards:

Organizing committee
ICAC3N - 2023

Download the CMT app to access submissions and reviews on the move and receive notifications:

<https://apps.apple.com/us/app/conference-management-toolkit/id1532488001>

<https://play.google.com/store/apps/details?id=com.microsoft.research.cmt>

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our [Privacy Statement](#).

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

 Reply

 Forward