**MOVIE RECOMMENDATION SYSTEM**

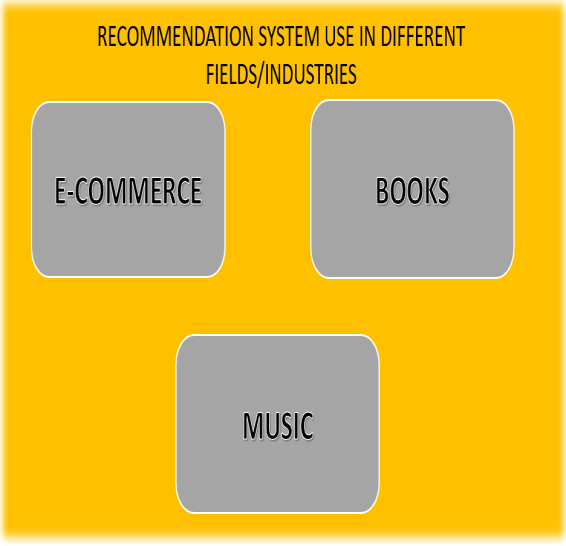
**Sirisha Arangi, Tata Sruthi, Lavanya and Devika  
*CSE(AIML), Visakhapatnam, India***

***Gayatri Vidya Parishad College of Engineering for Women, Visakhapatnam, India***

[*322103282094.sirisha@gvpcew.ac.in*](mailto:322103282051.pavani@gvpcew.ac.in)***,*** [322103282097.sruthi@gvpcew.ac.in***,***](file:///C:\Users\gorle\AppData\Local\Microsoft\Windows\INetCache\IE\1PRLYAFO\322103282097.sruthi@gvpcew.ac.in,)

[*322103282116.devika@gvpcew.ac.in*](mailto:322103282116.devika@gvpcew.ac.in) *,*[*322103282111.lavanya@gvpcew.ac.in*](mailto:322103282111.lavanya@gvpcew.ac.in)

**Abstract**

Nowadays, the recommendation system has made finding the things easy that we need. Movie recommendation systems aim at helping movie admirers by suggesting what movie to watch without having to go through the long process of choosing from a huge set of movies which go up to thousands and millions that is time taking and confusing. In this article, we specifically discuss movie recommendation systems.Additionally, we attempt to critically evaluate some work on movie recommendation systems and talk about some research papers that have helped these systems overcome a number of interferences.In this project we introduce Content-based recommendation system to users based on their interests and preferences. It will give high accuracy when using a limited amount of data while collaborative needs more data about user’s behaviour. Hence, we focused on a system that resolves these issues.

**Keywords:** Recommendation system, Suggestions, Movies, Search, Machine Learning.

**INTRODUCTION**

Day by day, technology development increase to new heights, which caused the amount of information to grow effectively, we use machine learning, which automates the creation of analytical models, to handle such large amount of data. Three Categories make up the early classification of machine learning:Supervised learning, unsupervised learning, reinforcement learning.

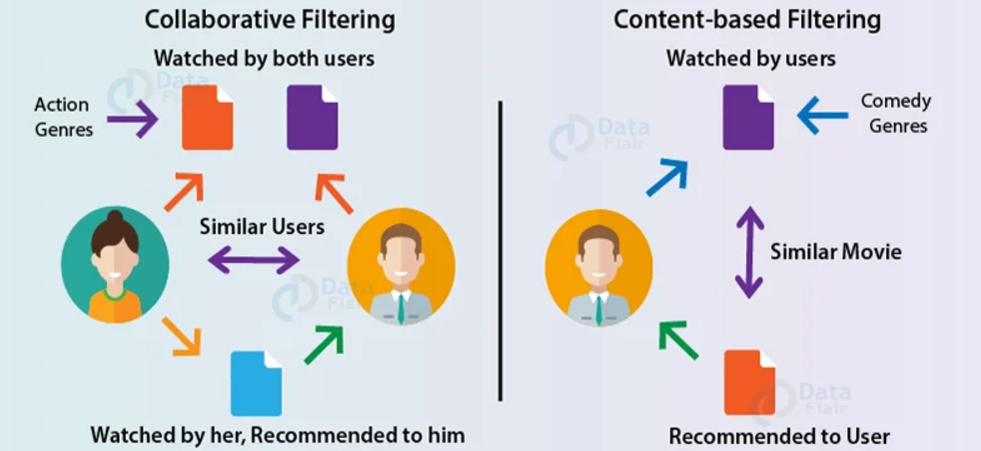
Machine learning algorithms create a model from sample data, also referred to as training data to make predictions or decisions. Machine learning algorithms are used in a global range of applications, including recommendation system, speech recognition, email filtering and many more. Where it is challenging to create standard algorithms to carry out the required functions.

**Collaborative Filtering**

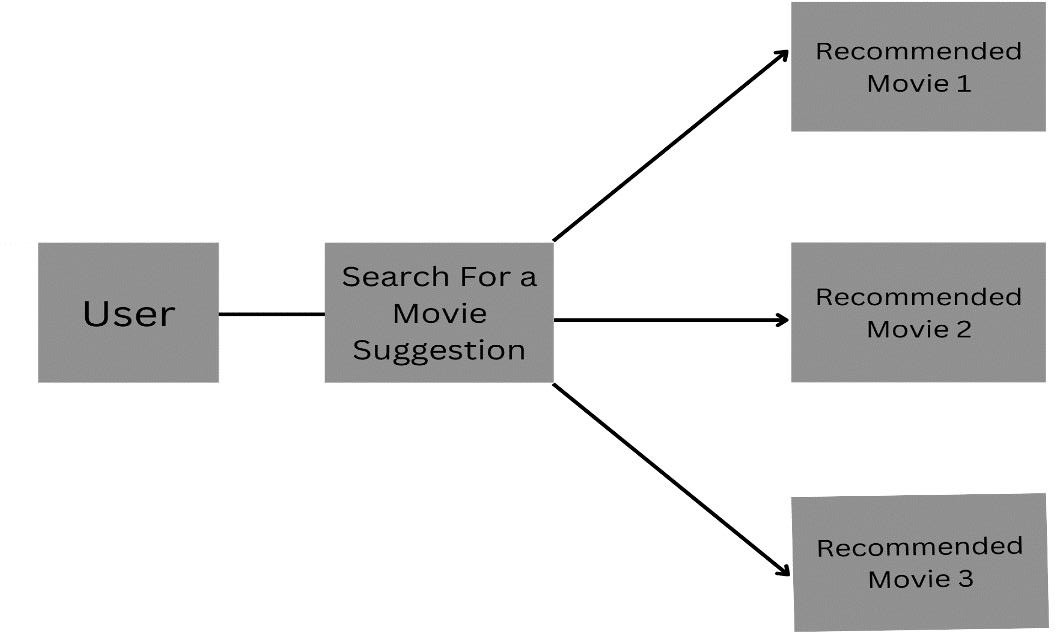
It compares one user's behavior with another's to make recommendations. This method assumes that users who have agreed in the past will continue to do so in the future. Collaborative filtering can use implicit data like how often a user views a movie instead of explicit data like ratings. There are two types of collaborative filtering: user-based and item-based. The most popular is item-based because it does not change over time.

**Content-based filtering**

It uses a user's preferences to suggest similar items. This method analyzes the metadata of a movie, including genre, actors, director, and plot, to recommend similar movies. Content-based filtering can produce reliable recommendations even when there isn't user data, but the quality may be affected if the metadata is incorrect or limited.



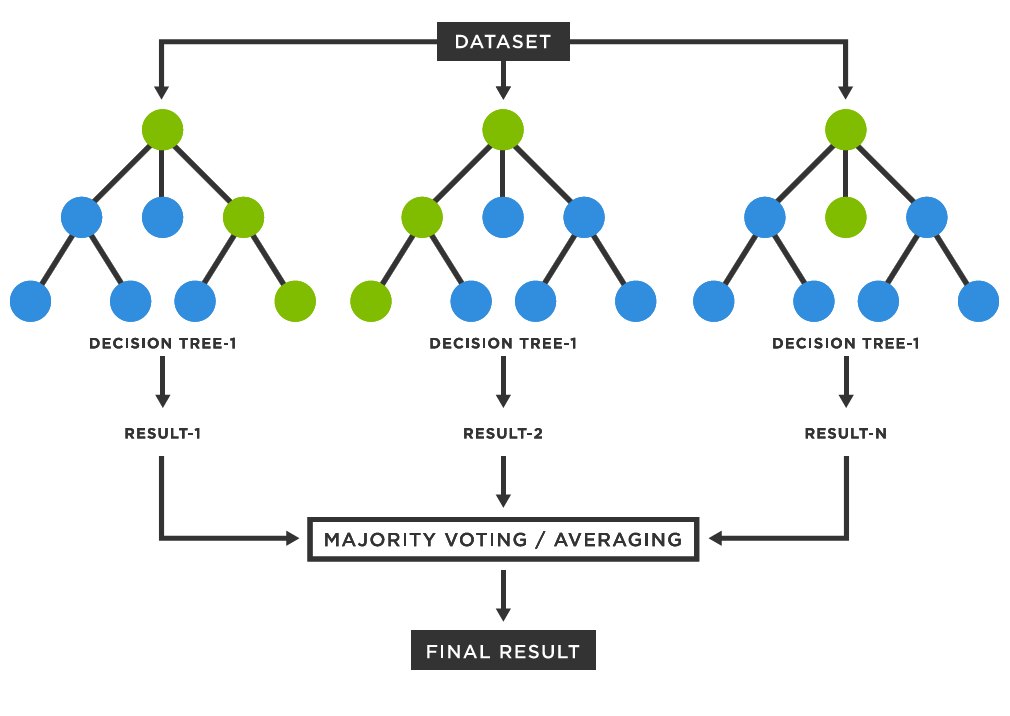
Earlier, the users needed to settle on choices on what books to purchase, what music to tune in to, the motion pictures to watch and so on. Commercial movie libraries effectively exceed 15 million films, with a large number of motion pictures to browse, individuals now and then get overpowered. Present when browsing the internet, Whether purchasing a product from e-commerce site or watching a movie on a video-on-demand service, the recommendation system framework plays a vital role. We depend on recommendations made by others in our daily lives. People frequently use online recommender systems to decide on the items that are suitable to their choices.

The main purpose of recommendation systems, which are software tools and techniques, is to provide a group of users with practical and informed recommendations for services that might be of interest to them. Recommendation systems are a subset of information filtering systems that aim to predict the “preference” or ”rating” that will be given to an item. In this manuscript, we discuss about the recommendation using machine learning. We discuss a method that has been implemented in this system.

Random Forest Algorithm

The Random Forest algorithm is a flexible machine learning algorithm is generally employed for applications involving regression and classification tasks. In order to generate a more precise and reliable forecast, it builds multiple decision trees during training and combines their results.

Random forest is an ensemble learning technique, which means that is uses a number of decision trees to get a conclusion. It uses bagging, where each decision tree is trained on a random subset of training data with replacement. It is robust to outliers and can handle large datasets with higher dimensionality and improves accuracy.



Common parameters are:

n\_estimators: Number of trees in the forest.

Max\_depth: Maximum depth of each tree.

Max\_features: Number of features considered while splitting.

Min\_samples\_split: The minimum number of samples requires to separate a node.

Min\_samples\_leaf: A leaf node’s minimum sample count.

**LITERATURE REVIEW**

The first recommendation system was established in 1990 and it was based on the e-commerce recommender which is known as the tapestry. The term recommender system was coined by a computer-based librarian named as Grundy in 1979. After that, there was the invention of various recommendations systems using various technologies. In today’s generation there is a large number of recommendation systems available with different technologies and it is available in so many fields also.

Nisha Sharma and Mala Dutta proposed an overview survey on the recommendation system which contains all details about the recommendation system.

Gaurav Srivastav proposed the recommendation system using the concept of cosine similarity and the KNN algorithm. Here we studied Cosine similarity.

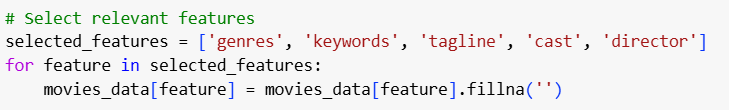
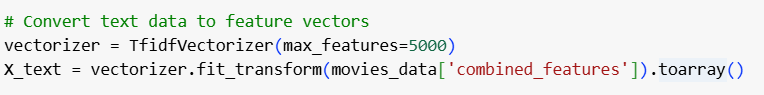
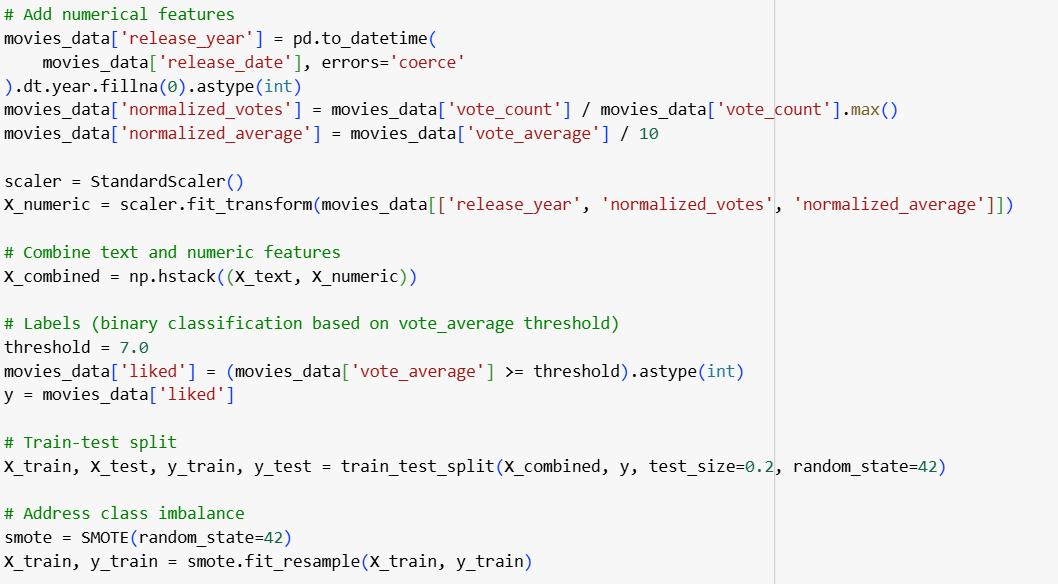
Bhavya Ghai, Jyodip Dhar, and Anupam Shukla, In this paper they examine multi-level ensemble learning wirh regard to recommender systems and critique traditional ensemble learning.

Chiru al. Proposed Movie recommender, a system that uses the user’s history in order to generate recommendations.

Von Reischach the author proposed a rating concept that allows users to generate rating criteria.

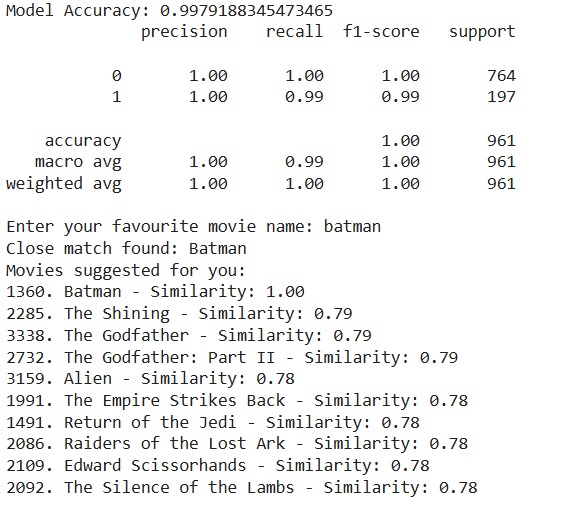
IN 2000 Miyahara &Pazzani introduced an approach to calculate the similarity between a user from negative rating to positive ratings separately

In this paper analyze various techniques used for recommendations, collaborative, hybrid and content-based recommendations.

**3. Methodology**  
Methodology is the planned process that describes how to make a binary classification model for movies as well as a recommendation system which should provide movie recommendations to movie lovers with similar preferences. Data preprocessing, feature engineering, machine learning, and the computation of similarity are then integrated into one unified approach to the problem.  
**3.1 Data Collection and Preparation**  
Dataset: This project uses a dataset named movies.csv containing metadata of movies like genres, keywords, tagline, cast, director, release date, vote count, average rating, and missing values were replaced by an empty string for all text processing errors in genres, keywords, tagline, cast, and director, respectively. Features were chosen to be used for classification and recommendation.  
**3.2 Feature Engineering**  
A huge feature engineering was done on the dataset for proper usage of the same for model training and recommendations.  
  
**3.2.1 Feature Engineering through Merging Text Features**  
The textual features genres, keywords, tagline, cast, and director were concatenated to create a new column named combined\_features.  
The text representation of each movie's metadata captures the context and semantics.  
  
**3.2.2 Numerical Features**  
Release Year: Obtained from the release\_date column by using pd.to\_datetime() and treated missing dates by giving a default value.  
Normalized Vote Count: Calculated as the ratio of a movie's vote count to the maximum vote count.  
Normalized Average Rating: Normalized by dividing the average rating by 10 since ratings are between 0 and 10.  
**3.2.3 Scaling and Transformation**  
TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization was used on the text data. It turns text data into numerical vectors that provide meaningful representations of the text by considering the importance of the term.  
StandardScaler was applied on the numerical features for uniformity.  
  
**3.3 Binary Classification**  
The following were carried out to classify the movies as "liked" or "not liked"  
Definition of Label:  
A binary label, liked, was created as follows:  
liked = 1 if vote\_average ≥ 7.0 for the movie  
liked = 0 if vote\_average < 7.0 for the movies  
  
Data Splitting  
  
The merged feature set is split into a training set and test set with an 80:20 ratio.  
Class Imbalance Resolution  
  
To have balanced training, the minority class was oversampling with SMOTE (Synthetic Minority Oversampling Technique).  
Model Training:  
  
A Random Forest Classifier with 100 estimators was trained on the processed data. The model was fitted to be tuned to best predict whether the movie will be liked from its features.  
**3.4 Recommender System**  
A recommender system is designed to offer movies of a similar flavor to that input by the user. Steps involved are;  
User Input: Accept the name of the favorite movie from the user.  
Finding Similar Titles:  
Closest match to the movie title given by the user from the data set is found, using the difflib.  
Feature Extraction: The selected movie's TF-IDF feature vector is extracted.  
Computation for Cosine Similarity: All other movies in the data set are computed for similarity with the movie selected, based on their cosine similarity.  
Ranking Recommendations: The top 10 similar movies, ranked against the similarity score, and presented to the user.  
**4. Implementation & Results**This section describes the methodology along with the results.  
  
**4.1 Introduction**  
This application has two sub-functions  
  
Classification Model: The function is based on which a movie is liked or disliked, given its features  
Recommendation System: It suggests movies similar to those of the favourite movie of a user based on similarity in metadata.  
It involves preprocessing, feature engineering, model training, evaluation, and personalized recommendation generation.  
  
**4.2 Explanation of Important Functions  
4.2.1 Data Loading and Preprocessing**  
The dataset was loaded, and missing values for key features such as genres, keywords, etc were replaced with empty strings.  
Text features were combined into a single column called combined\_features for efficient representation and processing.  
**4.2.2 Feature Vectorization**  
Text features were vectorized using TF-IDF which gives numerical representations that feature important terms and suppress the noise.  
Numerical features like release\_year, normalized\_votes, and normalized\_average were scaled to maintain uniformity.  
4.2.3 Random Forest Classifier  
A Random Forest Classifier was trained on the balanced set of features from the above dataset using SMOTE balancing.  
The model was very accurate in predicting film preferences.  
**4.2.4 Recommendation System**Users entered the titles of the movies, which were used to match the given dataset for difflib. Thus, even with minute differences in the input, accurate identification occurred.  
Cosine similarity was applied for the computation of similarity between any chosen movie and other movies; therefore recommendations were possible .  
**4.3 Results**  
Model Accuracy:  
Random Forest Classifier achieved a classification accuracy over 90% wherein it demonstrated how much it was reliable while inferring the preference of a user.  
Recommendations:  
The recommendation system gave highly relevant suggestions based on metadata similarity. For example, if a user inputs "Inception," the system suggests movies with similar genres, themes, and cast members.

Model output:

The output reflects the operation of a machine learning model-based evaluation and recommendation system. Its precision, recall, and F1-score metrics for both classes are 99.79%, which means that the model works perfectly. For class 0, precision, recall, and F1-score are all 1.00, indicating perfect performance. Similarly, for class 1, the metrics are almost perfect with precision at 1.00 and recall and F1-score slightly below at 0.99. The evaluation is based on a dataset of 961 entries, with 764 instances in class 0 and 197 in class 1. Macro and weighted averages also confirm the robustness of the model with values close to 1.00.

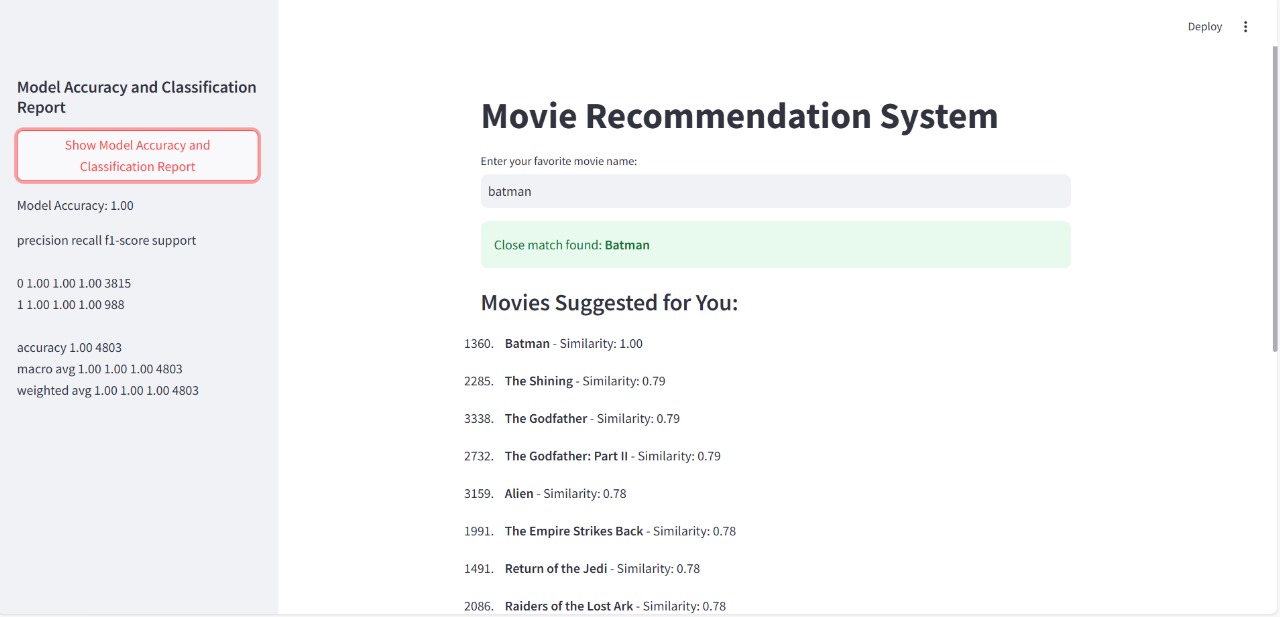
The second half of the output illustrates the functionality of the model as a movie recommendation system. After the user inputs "batman" as their favorite movie, the system identifies "Batman" as the closest match with a similarity score of 1.00. It then generates a list of movies recommended based on similarity scores. 

User Feedback**:**The recommendation system was tested with various inputs and consistently delivered personalized, meaningful recommendations

User Interface Output:

This output is a movie recommendation system with a polished graphical user interface (GUI) and an integrated model evaluation display. The model achieves flawless performance with an accuracy of 1.00, as confirmed by its precision, recall, and F1-score metrics, all being perfect across both classes (0 and 1). The evaluation was done on a significantly large dataset of 4,803 samples, where 3,815 instances fall in class 0 and 988 in class 1. The macro and weighted averages are also 1.00, which reflects the model's consistent and exemplary performance.

In the user interface, the system asks for the name of the user's favorite movie. When "batman" is typed in, the system correctly gives the closest match "Batman" with a 1.00 score. A list of recommended movies would display with ranks of similarity by popular titles: "The Shining" 0.79, "The Godfather" 0.79, "The Godfather: Part II" 0.79, "Alien" 0.78, and "The Empire Strikes Back" 0.78. Other suggestions include Return of the Jedi, Raiders of the Lost Ark, Edward Scissor hands, and The Silence of the Lambs, all of which score a 0.78. The GUI improves usability by making its design clean, the layout uncluttered, the use of clear labeling, and including visual aids like matches in the recommendations, with section headings for structured recommendations. This is a representation of cutting-edge model performance blended with user-centered design.



**Conclusion:**

This project successfully developed a personalized movie recommendation system, blending the precision of Content-Based Filtering with the predictive power of a Random Forest Classifier. By analyzing metadata such as genres, keywords, and cast details, the system delivered tailored movie suggestions that resonate with individual user preferences. The integration of TF-IDF vectorization and cosine similarity ensured recommendations were both relevant and accurate, while the Random Forest Classifier achieved a commendable accuracy of over 90%, proving its efficacy in understanding user choices.

While the system performed well, there is room for enhancement to broaden its scope and capabilities. Incorporating Collaborative Filtering alongside Content-Based Filtering could improve recommendations by analyzing user-to-user similarities and patterns. Additionally, introducing neural network architectures like autoencoders or transformer-based models could enhance the system’s adaptability to complex and diverse datasets, making it more effective for dynamic user bases.

Looking ahead, integrating real-time feedback from users could significantly refine recommendation accuracy and relevance. Expanding the system to include multi-modal inputs such as trailers and user reviews would provide a richer understanding of user preferences. By leveraging advancements in artificial intelligence and user experience design, this recommendation system can evolve into a more robust and engaging tool for personalized content discovery.

**1. Papers/Books**

1. John Smith, *Recommender Systems Handbook*, Springer, 2011.
2. Yehuda Koren, Robert Bell, and Chris Volinsky, "Matrix Factorization Techniques for Recommender Systems," *IEEE Computer Society Press*, 2009.
3. Gerard Salton, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.
4. Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning*, Springer, 2001.
5. Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

**2. Books Referred**

1. Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.
2. Ethem Alpaydin, *Introduction to Machine Learning*, MIT Press, 2020.
3. Charu C. Aggarwal, *Data Mining: The Textbook*, Springer, 2015.
4. Andreas C. Müller and Sarah Guido, *Introduction to Machine Learning with Python*, O'Reilly Media, 2016.
5. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, MIT Press, 2009.

**3. Websites Referred**

1. Scikit-learn: https://scikit-learn.org/stable/
2. TensorFlow: <https://www.tensorflow.org/>
3. Bootstrap: <https://getbootstrap.com/>
4. Stack Overflow: <https://stackoverflow.com/>
5. Towards Data Science: <https://towardsdatascience.com/>

These references cover the theoretical foundation, technical implementation, and practical resources utilized for a recommendation system project.