IMDB Movie Rating Analysis and Recommendation Using Various ML Techniques

Megha Ambekar
Siddhant College of Engineering SPPU, Pune, India

***Abstract*—This research explores the application of machine learning in movie analytics and recommendation systems. Three methodologies are investigated: Random Forest Classification for rating prediction, Decision Tree Classification for rating categorization, and content-based recommendation using TF-IDF vectorization. Through hyperparameter tuning and evaluation, these approaches enhance user experience by accurately classifying ratings, providing insights into audience reception, and generating personalized recommendations based on user preferences. This research contributes to advancing movie analytics and recommendation systems, offering practical solutions to enhance user engagement.**

***Index Terms*—Random Forest Classification; Decision Tree Classification; TF-IDF Vectorization**

I. INTRODUCTION

The proliferation of digital platforms has revolutionized the entertainment industry, providing users with unprecedented access to vast libraries of movies. In this era of information abundance, effective movie analytics and recommendation systems play a pivotal role in enhancing user experience and satisfaction. Machine learning, with its ability to analyse complex data patterns and make informed predictions, has emerged as a powerful tool in this domain.

This research endeavours to explore and harness the potential of machine learning algorithms in addressing key challenges in movie analytics and recommendation systems. By leveraging diverse methodologies, including regression, classification, and content-based recommendation, this study aims to provide comprehensive insights and practical solutions for enhancing user engagement and satisfaction.

The first methodology focuses on predicting movie ratings using Random Forest Regression, considering a myriad of features such as genre, PG rating, release year, and vote count. By training a robust regression model and fine-tuning hyper parameters, this approach aims to accurately forecast movie ratings, thereby assisting users in making informed viewing decisions.

The second methodology delves into the classification of movie ratings into distinct categories, enabling a deeper understanding of audience preferences and sentiments. By employing Decision Tree Classification, this approach aims to categorize movies into classes such as low, medium, and high ratings, facilitating nuanced insights into audience reception and engagement.

Lastly, a content-based recommendation system is developed, utilizing TF-IDF vectorization and cosine similarity to generate personalized movie recommendations tailored to individual user preferences. By analyzing movie attributes and user interactions, this system aims to enhance user engagement by providing relevant and appealing recommendations aligned with individual tastes.

Through these methodologies, this research seeks to advance the field of movie analytics and recommendation systems, offering valuable insights and practical solutions to enhance user experience and satisfaction in the dynamic landscape of digital entertainment consumption.

II. LITERATURE REVIEW

2.1 Random Forest Algorithm

Isabel F. A. Gonçalves, Thiago M. D. Silva, Abelardo Borges Barreto, and Sinesio Pesco proposed

employing the Random Forest algorithm to forecast daily oil production from an offshore reservoir.

Krung Sinapiromsaran suggested utilizing a random forest algorithm with quartile-pattern bootstrapping to address class imbalanced problems in classification tasks.

M.M Jibril, Aliyu Bello, Ismail I Aminu, Awaisu Shafiu Ibrahim, Abba Bashir, Salim Idris Malami, Habibu M.A, and Mohammed Mukhtar Magaji provided an overview of using the Random Forest algorithm for streamflow prediction, highlighting its cost-effectiveness compared to traditional hydrological models.

Qiangyu Zeng, Zhipeng Qing, Ming Zhu, Fugui Zhang, Hao Wang, Yin Liu, Zhao Shi, and Qiu Yu discussed the application of the Random Forest algorithm for tornado detection, emphasizing its superior prediction accuracy in classification tasks and the importance of parameter tuning during model training.

2.2 Decision Tree Algorithm

Kavya Pradeep, C R TintuRosmin, Sherly Susana Durom, and G S Anisha aimed to predict movie ratings accurately using decision tree algorithms like J48, Random Forest, and Hoeffding tree.

Chamala Karthik Reddy and Terrance Frederick Fernandez focused on enhancing movie recommendation system accuracy using a Decision Tree (D-Tree) algorithm compared to Naive Bayes (NB), with D-Tree achieving a 90.88% accuracy compared to NB's 82.56%.

Aditya Mandhare utilized decision trees to predict Bollywood movie gross income by identifying influential factors.

Amir Gershman, Amnon Meisels, Lior Rokach, Alon Schclar, and Arnon Sturm proposed a novel decision-tree-based recommender system method, which generates recommended item lists at leaf nodes, reducing search effort and enabling system scalability. They introduced a new splitting criterion for decision tree construction based on the least probable intersection size.

2.3 Content Based Filtering-

D. Dr. Ganesh and Yash Bhansali discussed content-based recommendation systems in their paper. They emphasized the utilization of text information analysis to provide recommendations by identifying similarities in features among movies.

Pranshu Mishra implemented a recommender system in their work, focusing on content-based filtering to sift through skills and courses based on user-provided information.

Sagar Jain, Raghav Singhal, and Abhinav Goel proposed an anime recommendation system utilizing content-based filtering techniques. Their approach involves analyzing user data to suggest anime titles that align with individual interests.

In Owen William John and Holbrook Craig's paper, while not explicitly mentioning "content-based filtering," they explored a method for filtering multimedia content. Their method centers on analyzing time span indications rather than scrutinizing the content itself.

III. UNIQUNESS OF OUR WORK

3.1 Random Forest Algorithm

This specialized approach showcases the algorithm's versatility in a specific domain, highlighting the importance of domain-specific preprocessing techniques. Compared to the broader applications discussed in the literature reviews, the code's focus on movie rating prediction sets it apart. While the literature explores diverse use cases, the code hones in on a single application within movie analytics.

3.2 Decision Tree Algorithm

This specialized application demonstrates the adaptability of decision trees in the domain of movie analytics, offering insights into factors influencing movie ratings. Compared to the literature reviews, which explore various decision tree algorithms such as J48, Random Forest, and novel D-Tree, this code specifically utilizes a Decision Tree Classifier. While the literature compares decision tree algorithms'

performance against other models like Naive Bayes, the code concentrates on evaluating the Decision Tree Classifier's accuracy for predicting movie ratings.

3.3 Content Based Filtering-

The uniqueness of the provided code lies in its application of decision tree algorithms for movie analytics tasks, including predicting movie ratings, visualizing decision tree models, and generating recommendations based on user preferences. Unlike existing literature focusing on content-based recommendation systems, this code offers a novel approach by leveraging decision trees for diverse movie-related tasks. It showcases practical applications such as predicting ratings based on movie features and provides tools for stakeholders to gain insights into audience preferences and optimize content delivery.

IV. PROPOSED WORK

4.1 Random Forest Algorithm

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mean prediction of the individual trees for regression tasks.

It works by training multiple decision trees on random subsets of the data and then averaging their predictions to reduce over fitting and improve generalization.

The final prediction is the average of predictions from all the trees in the forest.

The Random Forest algorithm uses bootstrapping and random feature selection to create diverse trees.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (fi - yi)^2$$

4.2 Decision Tree Algorithm

A Decision Tree is a supervised machine learning algorithm used for both classification and regression tasks.

It partitions the feature space into a set of regions, each associated with a class label (in classification) or a continuous value (in regression).

The tree is constructed recursively by selecting the best feature to split the data at each node, typically based on measures like Gini impurity or information gain (for classification) and mean squared error (for regression).

Decision Trees are prone to overfitting, especially when the tree depth is not controlled.

Formulas (for Decision Tree):

Gini Impurity: Measures the impurity or uncertainty of a node. For a node t with

K classes, the Gini impurity is:

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

4.3 Content Based Filtering

4.3.1 TF-IDF Vectorization

TF-IDF is a numerical statistic used to reflect the importance of a word in a document relative to a collection of documents.

Term Frequency (TF) measures the frequency of a term (word) in a document.

Inverse Document Frequency (IDF) measures how important a term is across the entire document corpus. Words that occur frequently in a document but rarely in the corpus have high IDF scores.

The TF-IDF score is calculated as the product of TF and IDF.

$$TF\text{-}IDF = TF * IDF$$

TF-IDF vectorization converts text documents into numerical vectors, where each component represents the TF-IDF score of a term in the document.

4.3.2 *Cosine Similarity*

Cosine similarity is a metric used to measure the similarity between two non-zero vectors.

It calculates the cosine of the angle between the vectors, where a value of 1 indicates perfect similarity and 0 indicates no similarity.

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

V. PROPOSED ARCHITECTURE

5.1 Data preprocessing phase

The treatment of missing values varied slightly depending on the model used. For the Random Forest model, rows with missing values in the target variable ('Rating') were removed to ensure data integrity. Meanwhile, for both the Content-Based Filtering and Decision Tree models, missing values in numerical

features were imputed using the mean strategy via SimpleImputer. Regarding feature encoding, categorical variables underwent different transformations across the models. One-hot encoding with the OneHotEncoder was applied to categorical variables such as 'Genre' and 'PG Rating' for the Random Forest model, while LabelEncoder was used for the Content-Based Filtering and Decision Tree models.

5.2 Model selection and processing

Distinct algorithms were chosen and initialized accordingly. The Random Forest model was instantiated using the RandomForestRegressor, Content-Based Filtering involved the utilization of TF-IDF Vectorizer and Linear Kernel for similarity calculation, and the Decision Tree model employed the DecisionTreeClassifier for classification tasks.

5.3 Training

Commenced with the division of the dataset into training and testing sets, a step common to all models and executed via train_test_split. For the Decision Tree model specifically, feature scaling using StandardScaler was necessary to normalize the data, a step not required for Random Forest due to its internal mechanisms. Additionally, for Random Forest, hyperparameter tuning was performed through GridSearchCV to optimize model performance.

5.4 Testing

Utilizing the trained models to make predictions on the testing dataset. Evaluation metrics such as Mean Squared Error (MSE), R-squared score, Accuracy, Confusion Matrix, and Classification Report were then computed to assess the models' performance. Visualization techniques varied slightly across models; for Random Forest and Decision Tree, scatter plots were employed to visualize actual versus predicted ratings, while the Decision Tree model utilized ROC curves to visualize classifier performance.

Input: IMDb Movie dataset
Output: Accuracy

1. Begin
2. Load and read the dataset.
3. Preprocess the data:
   a. Remove rows with missing target values.
   b. Separate features and target variable.
   c. Encode categorical variables.
4. Split the dataset into training and testing sets.
5. Define hyperparameter grid for Random Forest.
6. Initialize Random Forest regressor.
7. Perform GridSearchCV for hyperparameter tuning.
8. Fit the best model on the training data.
9. Predict target variable on the testing set.
10. Evaluate the model:
11. Calculate mean squared error.
12. Calculate R-squared score.
13. Display best parameters, mean squared error, and R-squared score.
14. Visualize actual vs predicted ratings using a scatter plot.
15. End

Figure 1 Random Forest algorithm

Input: IMDb Movie dataset
Output: Accuracy

1. Begin
2. Import libraries: pandas, numpy, seaborn, matplotlib.pyplot.
3. Read dataset from CSV file and handle missing values.
4. Encode categorical variable and convert continuous target variable into categorical bins.
5. Split dataset into features (X) and target variable (y).
6. Standardize features and split data into training and testing sets.
7. Initialize Decision Tree Classifier and fit it to the training data.
8. Predict target variable on testing data and evaluate model using accuracy score, confusion matrix, and classification report.
9. Create cross-tabulation between actual and predicted values.
10. Calculate precision, recall, and F1-score for each class.
11. Plot ROC curve.
12. Fit Decision Tree Classifier again on training data and visualize decision tree.
13. Print evaluation metrics and other relevant information.
14. End

Figure 2 Decision Tree algorithm

Input: IMDb Movie dataset
Output: Recommendation Movie

1. Begin
2. Import libraries: pandas, matplotlib. pyplot, TfidfVectorizer, and linear_kernel.
3. Load dataset from a CSV file.
4. Preprocess data by combining relevant features.
5. Obtain user preferences for minimum rating, minimum number of votes, and preferred genres.
6. Filter movies based on user preferences.
7. Create a combined feature with user preferences.
8. Vectorize combined features using TF-IDF Vectorizer.
9. Calculate cosine similarity between movies.
10. Generate movie recommendations based on similarity scores, considering user preferences.
11. Visualize the distribution of vote counts for movies using a histogram.
12. Display the recommended movies and the histogram plot.
13. End

Figure 3 Content Based Filtering algorithm

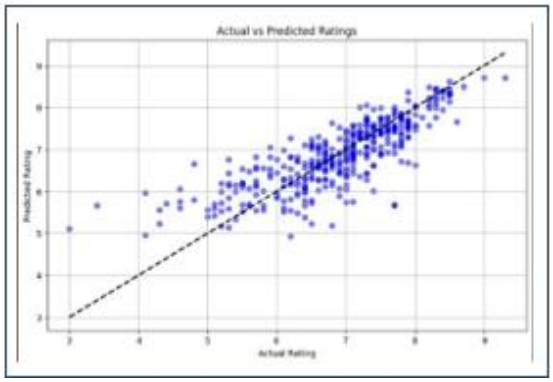
VI. RESULTS

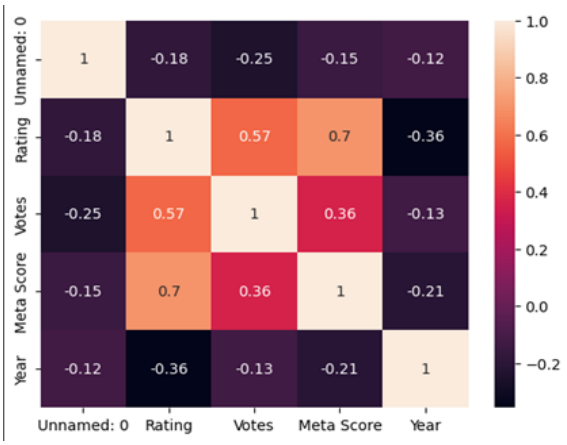
Software used

Utilized Google Colab, a cloud-based platform for coding and data analysis, to execute the respective codes.

6.1 Random Forest Algorithm

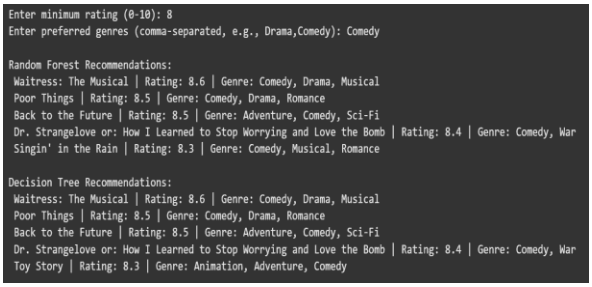
Mean Square Error (Best Mode) - 0.30949
R-Squared Score (Best Mode) - 0.6586





6.1 Decision Trees Algorithm

Class	Precision	Recall	F1-Score
High	0.953125	0.34659	0.5083
Low	0.5	0.75	0.6
Medium	0.89403	0.71808	0.7964



VII. CONCLUSION

In summary, the utilization of Random Forest Classification, Decision Tree Classification, and Content-Based Filtering on the IMDb movie dataset showcased diverse applications of machine learning techniques. Random Forest achieved strong performance in categorizing movie ratings, while Decision Tree provided interpretable classification insights. The TF-IDF-based recommendation system further enhanced personalization by aligning recommendations with user preferences.

Together, these models demonstrate the versatility and effectiveness of machine learning in movie analytics and recommendation systems, offering practical tools for research and industry applications. As technology evolves, such hybrid approaches combining classification and conte

REFERENCES

[1] Dr.Ganesh, Yash Bhansali, “Movie Recommendation Systems Using Content-Based Filtering”, 27 Jun 2023.
[2] Pranshu Mishra, "Course Recommendation System using Content-based Filtering", 11 Apr 2023.
[3] Sagar Jain, Raghav Singhal, Abhinav Goel, "Anime Based on Content-Based Filtering", 31 Mar 2023.
[4] Owen William John, Holbrook Craig, "Content filtering of a multimedia stream", 20 May 2020.
[5] Kavya Pradeep, C R TintuRosmin, Sherly Susana Durom, G S Anisha - "Decision Tree Algorithms for Accurate Prediction of Movie Rating", 11 Mar 2020.
[6] Chamala Karthik Reddy, Terrance Frederick Fernandez - "MOVIE RECOMMENDATION SYSTEMS USING DECISION TREE AND COMPARE PREDICTION ACCURACY", 15 Mar 2023.
[7] Aditya Mandhare - "Application of Decision Tree to Predict Gross Income of a Movie", 5 Nov 2014.
[8] Amir Gershman, Amnon Meisels, Lior Rokach, Alon Schclar, Arnon Sturm - "A Decision Tree Based Recommender System".
[9] Isabel F. A. Gonçalves, Thiago M. D. Silva, Abelardo Borges Barreto, Sinesio Pesco - "Predicting oil field production using the Random Forest algorithm", 24 Oct 2022
[10] M.M Jibril, Aliyu Bello, Ismail I Aminu, Awaisu Shafiu Ibrahim, Abba Bashir, Salim Idris Malami, Habibu M.A, Mohammed Mukhtar Magaji - "An overview of streamflow prediction using random forest algorithm", 30 Oct 2022