Movie Ratings Classification & Analysis

Ushama Rashid (23341079), Faria Naz Tabassum (21201815)

Zaima Mashiat Nabi (23241120), Md. Zaber Mahmud (22299273)

CSE-437 Data Science Group-10
Department of Computer Science and Engineering
BRAC University

***Abstract*—This project has helped us determine the rating of a movie into four classifications: "Excellent", "Good", "Average" or "Poor". With the help of categorical and quantitative figures like budget, genre, runtime, popularity and many more, models like KNN, random forest, logistic regression and decision tree are used to predict the ratings of a given movie.**

***Index Terms*—Movie Ratings, Classification, Machine Learning, KNN, Logistic Regression, Naive Bayes, Decision Tree, MLP Classifier.**

I. Introduction

This project aims to classify movies into one of four rating categories: "Excellent," "Good," "Average," or "Poor." Using features such as budget, genre, runtime, and popularity metrics, we use multiple machine learning models to predict the movie rating category. The goal is to develop a system that can automate movie rating prediction based on pre-release metadata. The motivation lies in the increasing reliance on data-driven decision-making in the film industry and the need to assess film performance before market release.

II. Dataset Description

The dataset comprises 1200 movies and 11 features, including both quantitative (e.g., Budget, Runtime, Marketing Spend) and categorical features (e.g., Genre, Sequel). The goal is to predict a target variable, Rating_Category, with four possible classes: "Excellent," "Good," "Average," and "Poor."

A. Dataset Overview

- Total Features: 11 input features
- Target Column: Rating_Category (4 classes)
- Problem Type: Classification
- The target variable is categorical with values such as "Excellent," "Good," etc.
- Total Records: 1200 movies
- Feature Types:
  - Quantitative: Budget_MillionUSD, Runtime_Minutes, Release_Year, Director_Popularity, Num_Main_Actors, Avg_Actor_Popularity, Num_Awards_Won, Marketing_Spend_MillionUSD
  - Categorical: Genre, Has_Famous_Producer, Is_Sequel

B. Imbalanced Dataset Check

The distribution of classes is fairly balanced:

- Excellent: 304
- Good: 324
- Average: 294
- Poor: 278

III. Dataset Preprocessing

Data preprocessing steps include handling missing values, encoding categorical variables, and scaling numerical features. The preprocessing pipeline ensures the dataset is ready for machine learning models.

A. Null/Missing Values

Missing values were detected in several columns, including Genre, Budget, Director_Popularity, etc. Imputation was performed as follows:

- Numerical columns: Imputed using the most frequent strategy
- Categorical columns: Imputed using the most frequent strategy

B. Categorical Variables

The following categorical variables were one-hot encoded:

- Genre
- Is_Sequel
- Has_Famous_Producer

C. Feature Scaling

Numerical features like Budget, Runtime, etc., were scaled using the StandardScaler to ensure all features have the same magnitude, which helps improve model performance.

IV. Dataset Splitting

Split Type: Stratified to preserve class distribution
Training Set: 70% (840 samples)
Testing Set: 30% (360 samples)

V. Model Training & Testing

A. K-Nearest Neighbors (KNN)

- Library: Scikit-learn
- Parameters: n_neighbors=5
- Evaluation: 27% accuracy

KNN is a distance-based, instance-based learning algorithm. It classifies a new data point by looking at the most common class among its k nearest neighbors. The model is simple, does not require training, and is highly sensitive to feature scaling.

B. Logistic Regression

- Library: Scikit-learn
- Solver: liblinear
- Evaluation: Comparable to MLP

C. Random Forest

- Type: Random Forest
- Evaluation: Surprisingly competitive

D. Decision Tree

- Library: Scikit-learn
- Evaluation: Approximately 19% accuracy

VI. Model Selection & Comparison

A. Accuracy Comparison

A bar chart was plotted to compare model accuracies. The bar chart image can be included here:

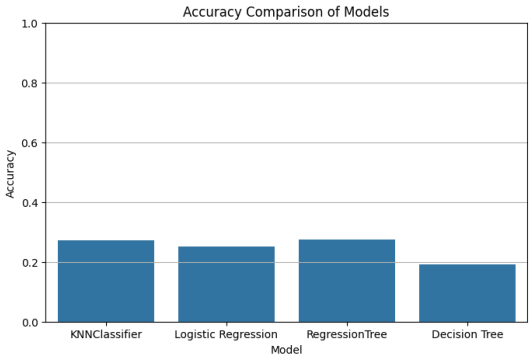

Fig. 1. Accuracy Comparison of Models

B. Precision & Recall

Precision/recall values were compared, with MLPClassifier leading in terms of performance. Here is the precision and recall chart:

Fig. 2. Precision and Recall by Model

C. Confusion Matrices

Confusion matrices for all models were plotted using ConfusionMatrixDisplay. Here is the confusion matrix for Logistic Regression:

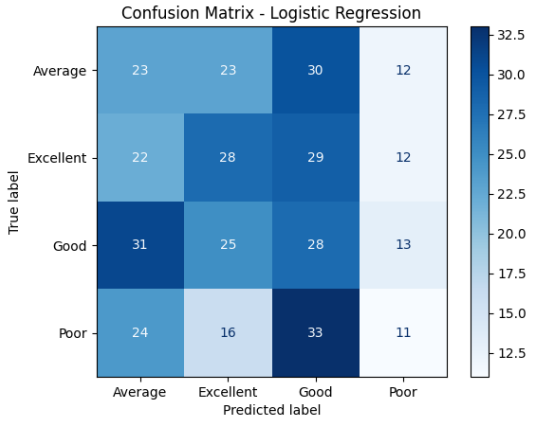

Fig. 3. Confusion Matrix - Logistic Regression

For KNN:

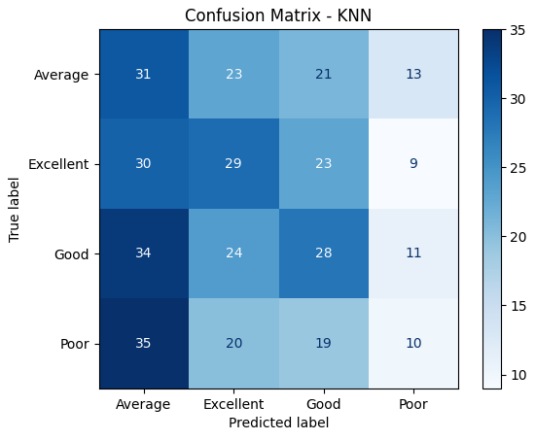

Fig. 4. Confusion Matrix - KNN

And for Random Forest:

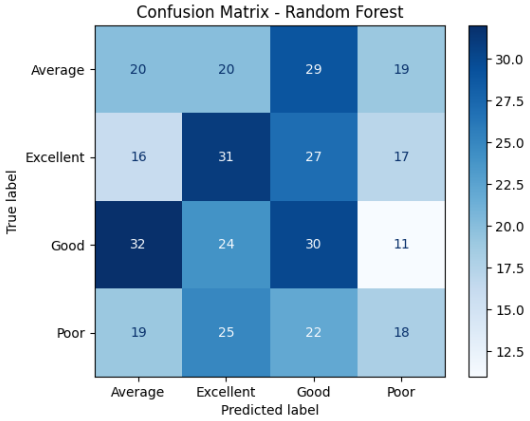

Fig. 5. Confusion Matrix - Random Forest

For Decision Tree:

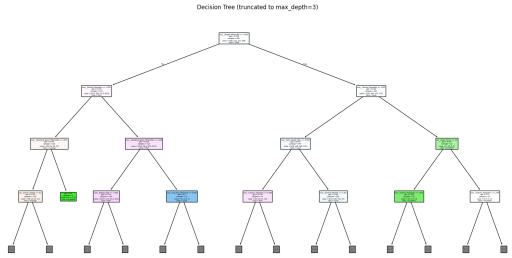

Fig. 6. Confusion Matrix - Decision Tree

D. ROC Curves & AUC Scores

ROC curves for all four classes were plotted for each model. AUC scores per class were shown in the legends. Below is the ROC curve for KNN:

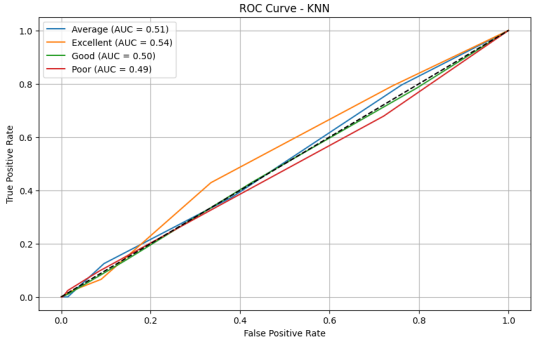

Fig. 7. ROC Curve - KNN

For Logistic Regression:

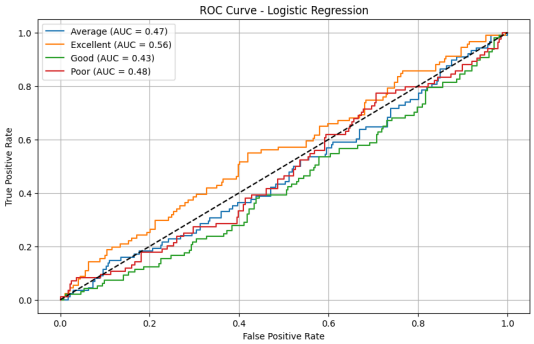

Fig. 8. ROC Curve - Logistic Regression

And for Random Forest:

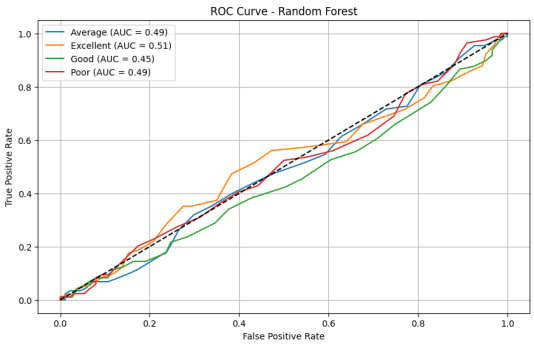

Fig. 9. ROC Curve - Random Forest

VII. Conclusion

From the evaluation, Logistic Regression and Naive Bayes provided competitive results compared to the MLPClassifier. While the neural network might improve with tuning and more data, the simpler models performed adequately. The biggest challenge was the near-baseline performance of the neural network, indicating either insufficient data complexity or the need for model tuning. Imputation and encoding choices played a big role in data usability. The project was successful in building a full classification pipeline with clean comparisons across algorithms.