CSE-422 Artificial Intelligence

Lab Project

Title- Movie ratings classification & analysis.

Group-4, members:

Ushama Rashid	23341079
Protaya Roy	24241160

Table of contents:

1. 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.

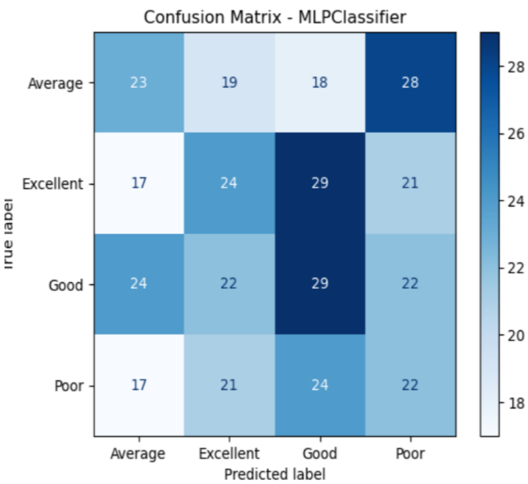
2. Dataset Description

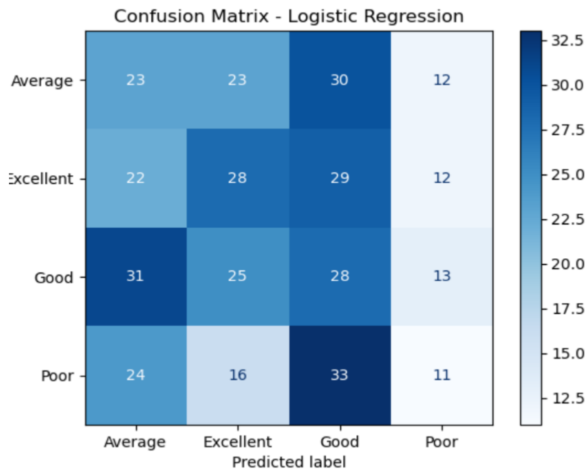
Dataset Overview

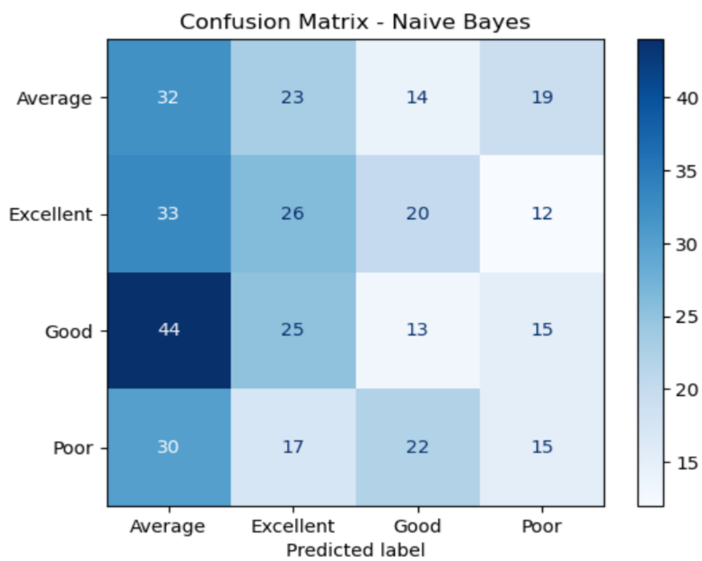
- **Total Features:** 11 input features
- **Target Column:** Rating_Category (4 classes)
- **Problem Type:** Classification
  - The target variable is categorical with values such as "Excellent," "Good," etc.
- **Total Records:** 1200 movies
- **Feature Types:**
  - **Quantitative:** Budget_MillionUSD, 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

Correlation Analysis

We have used a confusion matrix as our heatmap for each of the models.







Imbalanced Dataset Check

- The distribution of classes is fairly balanced:
  - Excellent: 304
  - Good: 324
  - Average: 294
  - Poor: 278
- A percentage chart confirmed that the dataset is not significantly imbalanced.

```
print("Target variable distribution:")
print(y.value_counts(normalize=True))
```

```
Target variable distribution:
Rating_Category
Good         0.270000
Excellent    0.253333
Average      0.245000
Poor         0.231667
Name: proportion, dtype: float64
```

EDA Highlights

- High-budget movies tended to receive better ratings.
- Sequels slightly leaned toward poorer ratings.
- Movies with famous producers had a higher chance of being rated "Excellent."

```
Numerical features statistics:
       Director_Popularity  Budget_MillionUSD  Runtime_Minutes  Release_Year  \
count          1067.000000        1096.000000      1074.000000   1088.000000
mean              5.452624         152.768723       129.401304   2002.330882
std               2.600397          85.998943        28.744659     13.068670
min               1.010000           1.040000        80.000000   1980.000000
25%               3.310000          78.857500       103.000000   1991.000000
50%               5.400000         156.060000       130.000000   2002.000000
75%               7.690000         226.017500       154.000000   2014.000000
max              10.000000         299.730000       179.000000   2024.000000

       Num_Main_Actors  Avg_Actor_Popularity  Num_Awards_Won  \
count      1075.000000           1066.000000     1061.000000
mean          2.531163              5.495159       24.113101
std           1.143236              2.620555       14.464153
min           1.000000              1.000000        0.000000
25%           1.000000              3.230000       12.000000
50%           3.000000              5.470000       24.000000
75%           4.000000              7.820000       37.000000
max           4.000000             10.000000       49.000000

       Marketing_Spend_MillionUSD
count                 1089.000000
mean                    24.351947
std                     14.648245
min                      0.000000
25%                     11.130000
50%                     24.430000
75%                     37.160000
max                     49.980000
```

3. Dataset Preprocessing

Problem: Null/Missing Values

- **Detected:** Missing values in Genre, Budget, Director_Popularity, etc.
- **Solution:**
  - Numerical columns: Imputed using **most frequent** strategy to handle NaN values.
  - Categorical columns: Imputed using **most frequent** strategy to handle NaN values.

Problem: Categorical Variables

- **Detected:** Genre, Is_Sequel, Has_Famous_Producer
- **Solution:** One-hot encoding was applied to ensure compatibility with sklearn models.

Problem: Feature Scaling

- **Detected:** Input features like Budget, Runtime, etc., vary in magnitude.
- **Solution:** Used StandardScaler to normalize numerical features (mean=0, std=1).

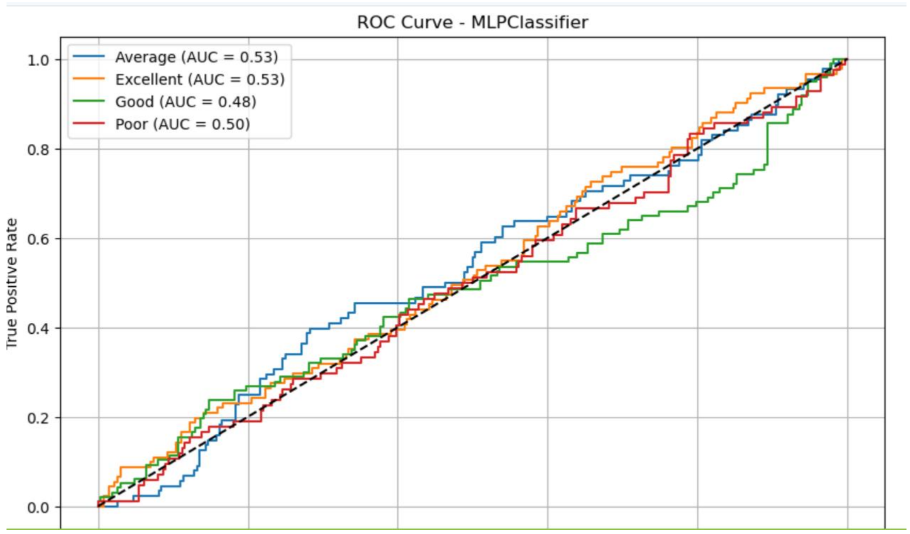
4. Dataset Splitting

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

5. Model Training & Testing:

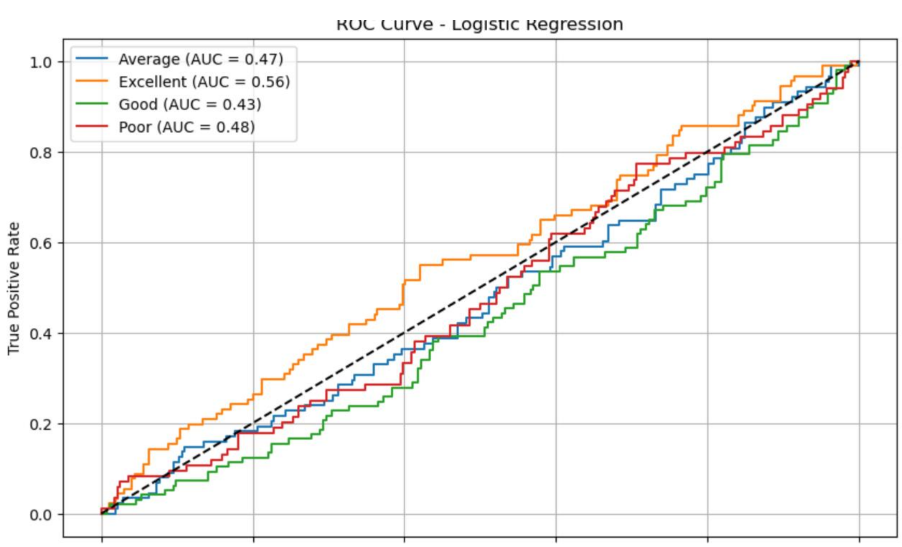
5.1. MLPClassifier

- **Library:** Scikit-learn
- **Layers:** 1 hidden layer with 100 neurons
- **Evaluation:** 24-26% accuracy
- **Note:** Basic neural network without hyperparameter tuning



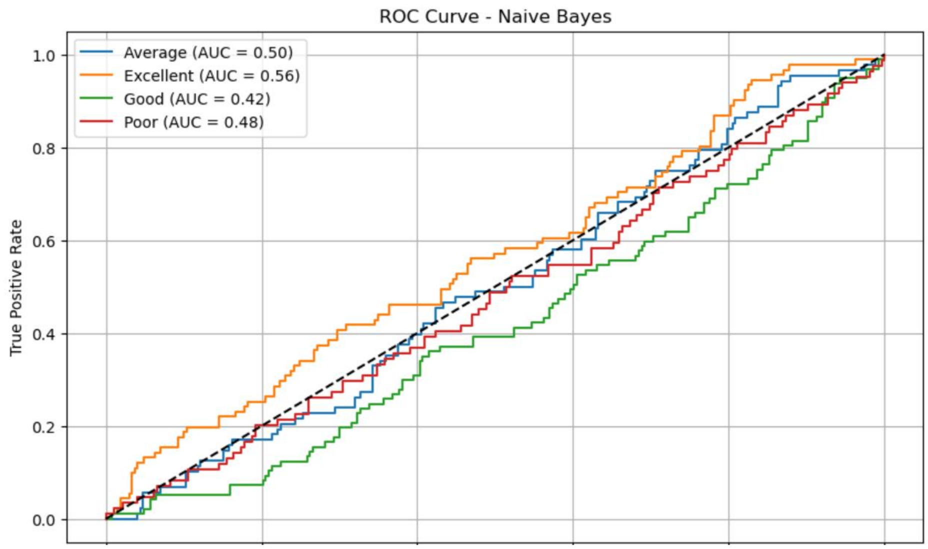
5.2. Logistic Regression

- **Library:** Scikit-learn
- **Solver:** Default (liblinear)
- **Evaluation:** Comparable performance with MLP.



5.3. Naive Bayes

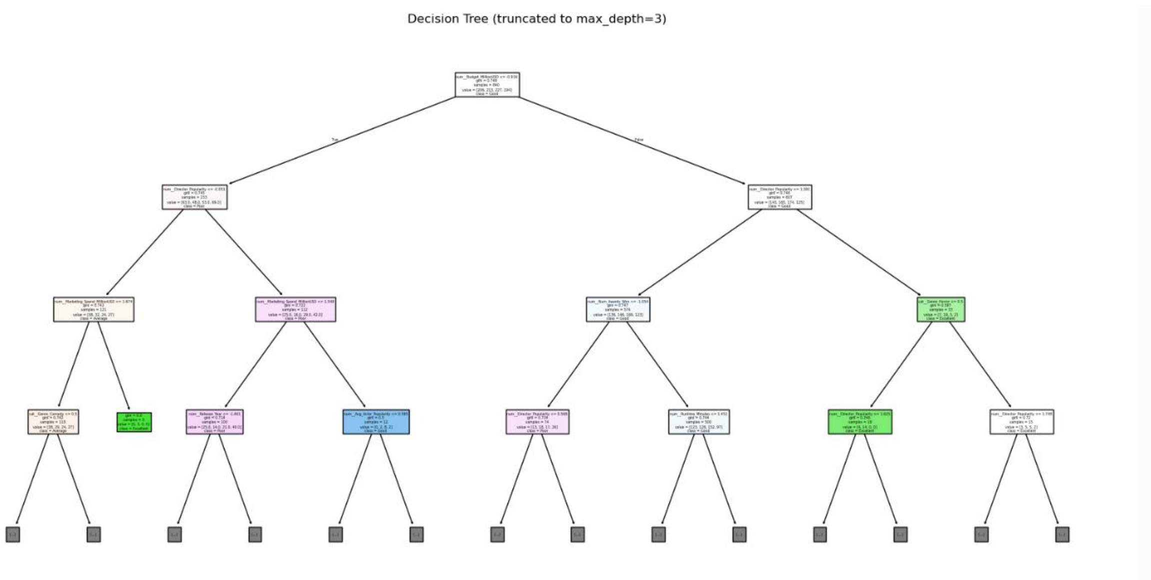
- **Type:** GaussianNB
- **Evaluation:** Simpler but surprisingly competitive for some classes



5.4. Decision Tree

- **Library-** Scikit-learn
- **Evaluation-** Approximately 19% accuracy.

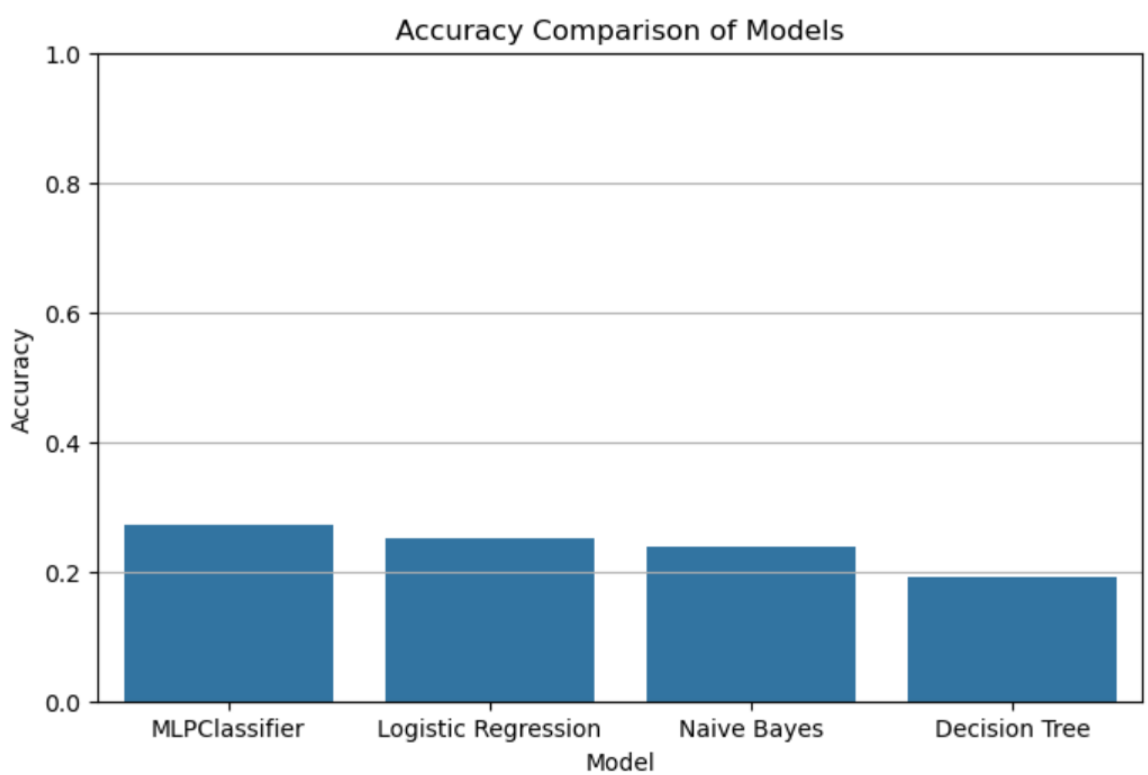
```
Decision Tree accuracy: 0.19166666666666668
Classification report:
              precision    recall  f1-score   support

     Average       0.17      0.16      0.17        88
   Excellent       0.20      0.21      0.20        91
        Good       0.18      0.16      0.17        97
        Poor       0.21      0.24      0.22        84

    accuracy                           0.19       360
   macro avg       0.19      0.19      0.19       360
weighted avg       0.19      0.19      0.19       360

CV scores: [0.25833333 0.2375     0.25       0.24166667 0.24166667]
Mean CV accuracy: 0.24583333333333335
```

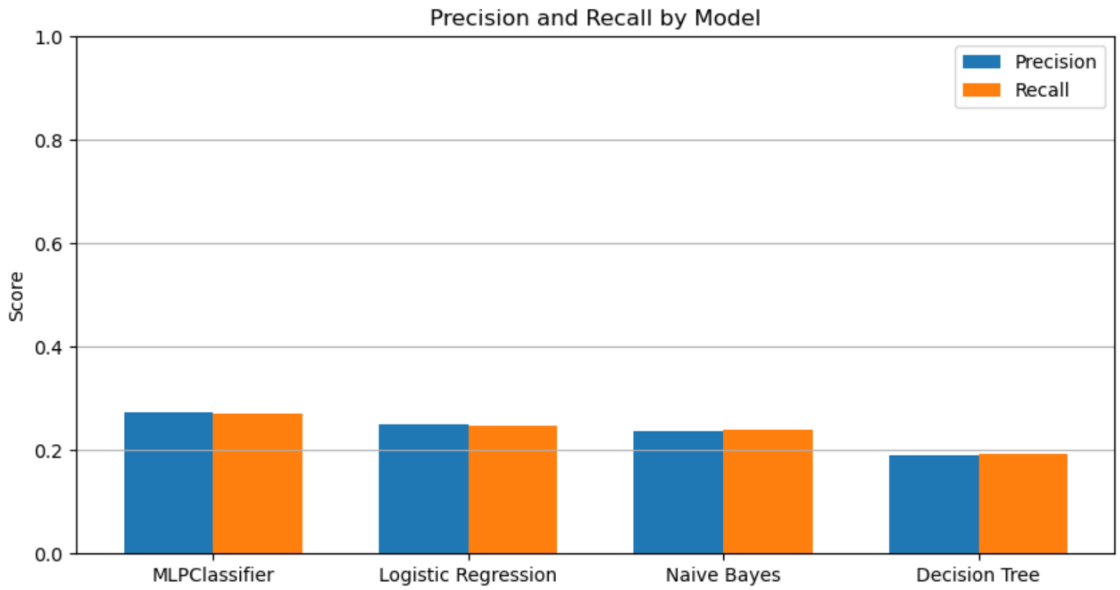


6. Model Selection & Comparison Analysis

Accuracy Comparison (Bar Chart)

A bar chart was plotted to show prediction accuracies across all models.





Precision & Recall

- All models were compared using **macro average precision** and **recall**.
- Precision/recall values are comparable for Logistic Regression and Naive Bayes with MLPClassifier leading.

False Positive Rate

```
Neural Network (MLP) accuracy: 0.2722222222222222
Classification report:
              precision    recall  f1-score   support

     Average       0.28      0.26      0.27        88
   Excellent       0.28      0.26      0.27        91
        Good       0.29      0.30      0.29        97
        Poor       0.24      0.26      0.25        84

    accuracy                           0.27       360
   macro avg       0.27      0.27      0.27       360
weighted avg       0.27      0.27      0.27       360
```

MLPclassifier having the highest accuracy.

Confusion Matrices

- Confusion matrices for all three models were plotted using ConfusionMatrixDisplay.

ROC Curves & AUC Scores

- ROC curves for all 4 classes were plotted for each model.
- AUC scores per class were shown in legends.

7. 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.