University at Buffalo
Project Phase #2
REPORT

Pre-Processing for Training

Since we are dealing with a mix of both numerical and non numerical data to create our model while including all our data, we have a complex situation.

Thus we need to make it that all our data is combined into one stream for training and other purposes. Thus we have separated the columns into two categories Namely the Numeric Features (YEARS, VOTES, RunTime) and the categorical features(ONE-LINE,STARS,GENRE).

We pass the numeric data through the pipeline to fill any missing values and also scale data.

For the non numeric data since there are a lot of recurring values like genres and stars , we have used one hot encoding to convert them into a numeric measure.

Other methods like TF IDF are also available but looking at the content of our data this approach is more suited.

After the individual processing we have combined both the numeric and the converted numeric values into the same pipeline using the column transformer.

```
# Identify numeric and categorical features
numeric_features = ['VOTES', 'RunTime','YEAR']
categorical_features = ['GENRE','ONE-LINE','STARS']

# Define transformations
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

# Combining preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])

# Now, 'preprocessor' can be used within a Pipeline along with an estimator of your choice
```

Here in our dataset we have both Qualitative data(ONE-LINE,STARS,GENRE) and Quantitative data(Numerical), So to process the model if we have text data it is difficult that model trains properly for that purpose, To overcome this we used **ColumnTransformer** in scikit-learn it is a powerful tool for applying different transformations to different columns of a dataset, allowing for flexible preprocessing of this kind of data.

This is the main technique we have used to overcome the problem of processing the model on text data.Above is the code snippet used in our code.

Numeric Transformer:

- The '**numeric_transformer**' pipeline is applied to numeric features ('VOTES', 'RunTime', 'YEAR').
- **'SimpleImputer'** is used to handle missing values by replacing them with the median of each feature. This strategy is chosen to mitigate the impact of outliers.
- **'StandardScaler'** is applied to standardize the numeric features by removing the mean and scaling them to unit variance. This step ensures that all numeric features have a similar scale, which is crucial for many machine learning algorithms, particularly those sensitive to feature scaling like SVMs and KNN.

Categorical Transformer:

- The **'categorical_transformer'** pipeline is applied to categorical features ('GENRE', 'ONE-LINE', 'STARS').
- **'SimpleImputer'** is used with a constant strategy to fill missing values with the string 'missing'. This strategy ensures that missing categorical values are handled gracefully without biasing the model.
- **'OneHotEncoder'** is applied to encode categorical variables as binary vectors. It creates binary columns for each category in the categorical feature, representing the presence or absence of that category. This encoding is essential for algorithms that cannot directly handle categorical data.

Combining Preprocessing Steps:

- The **'ColumnTransformer'** named **'preprocessor'** combines the numeric and categorical transformers.
- It specifies which transformer to apply to each type of feature (numeric or categorical) using the '**transformers'** parameter. Each transformer is specified with a tuple containing a name, the transformer itself, and a list of column names to apply the transformation to.

Explanation and Analysis

1. Random Forest Algorithm

- Justification for why we have chosen Random Forest

  For Handling Mixed Data Types our dataset includes numerical (e.g., RATING, VOTES, RunTime), categorical (e.g., GENRE, YEAR), and text data (e.g., ONE-LINE, STARS). Random Forest can handle this variety of data types effectively, especially with appropriate pre-processing.
  Random Forest is less prone to overfitting compared to individual decision trees, making it a good choice for datasets with diverse and complex relationships.

- Tuning/Training the Model

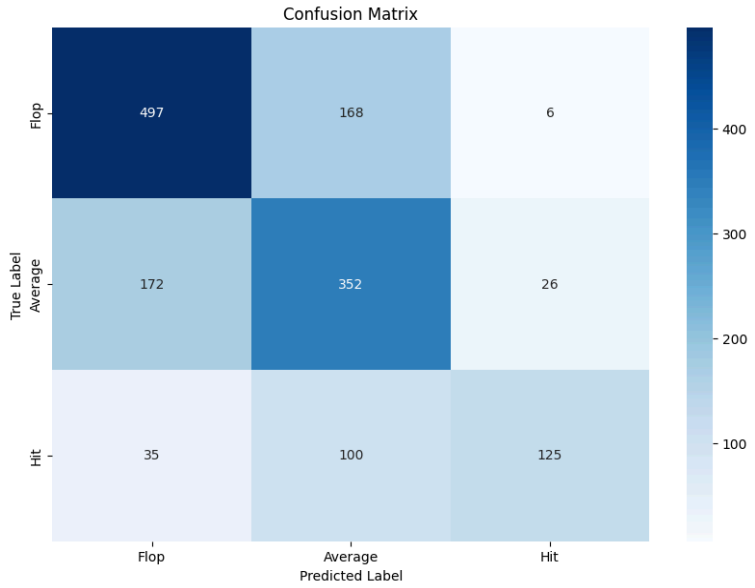
  1. **Pre-process the data:** Convert categorical variables to a format that can be used by the model (e.g., one-hot encoding), handle missing values, and normalize or scale numerical variables if necessary.
  2. **Feature Engineering**: Extract features from text data using techniques like TF-IDF for the ONE-LINE and STARS columns, and potentially engineer new features (e.g., extracting the main genre or director's name).
  3. **Splitting the dataset:** Divide the data into training and testing sets to evaluate the model's performance.
  4. **Model training and hyperparameter tuning:** Train the Random Forest model, adjusting hyperparameters (e.g., number of trees, depth of trees, min samples per leaf) to find the best performing model. This can be done using cross-validation techniques.

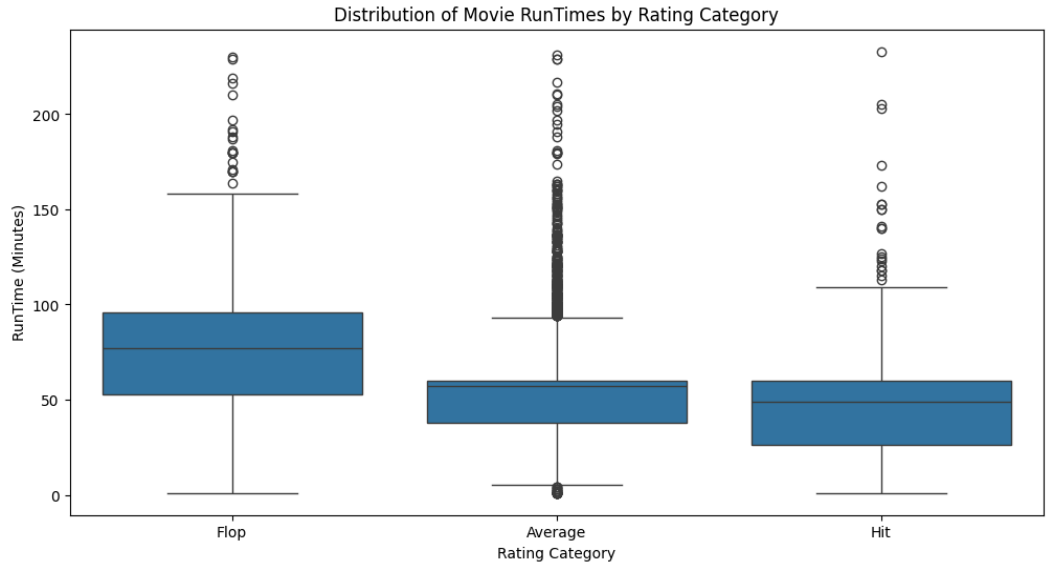
- Model Effectiveness

```
Random Forest Classifier Evaluation:
Accuracy: 0.6576637407157326
[[497 168   6]
 [172 352  26]
 [ 35 100 125]]
Classification Report:
              precision    recall  f1-score   support

        Flop       0.57      0.64      0.60       550
```

Average	0.71	0.74	0.72	671
Hit	0.80	0.48	0.60	260
accuracy			0.66	1481
macro avg	0.69	0.62	0.64	1481
weighted avg	0.67	0.66	0.66	1481

The Random Forest Classifier's performance on the dataset offers insightful information into predicting movie success with an overall accuracy of about 66%. It showcases a particular strength in precisely identifying hit movies, evidenced by a high precision score in this category. This model's ability to discern between flops, average, and hit movies, despite the complexity and variability inherent in movie data, is noteworthy. The relatively high precision in classifying hits suggests that the model has effectively captured key factors that contribute to a movie's success.





2. Support Vector Machine Classifier

- Justification

  High Dimensionality: Our dataset could become high-dimensional after preprocessing, especially when we are converting text features from the ONE-LINE and STARS columns into numerical features using techniques like TF-IDF. SVM works well in high-dimensional spaces, even when the number of dimensions exceeds the number of samples.
  Margin Maximization for Generalization: SVM focuses on finding the hyperplane with the maximum margin, which helps in better generalization on unseen data. This property made SVM a choice for predicting movie ratings or classifying genres.

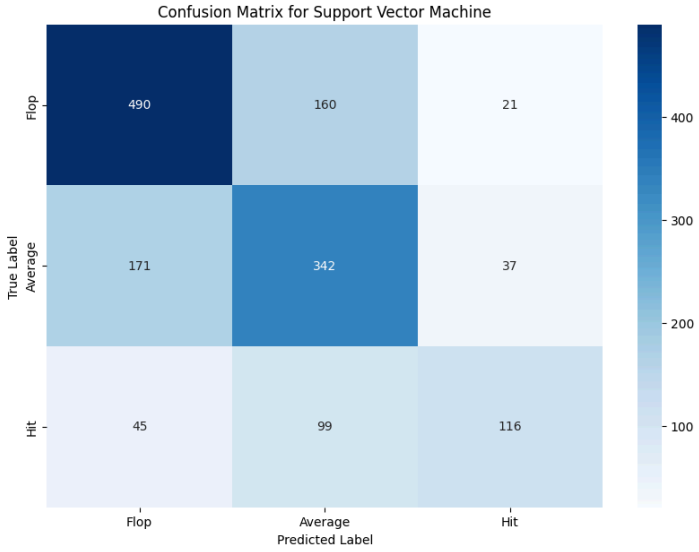
- Tuning/Training the Model

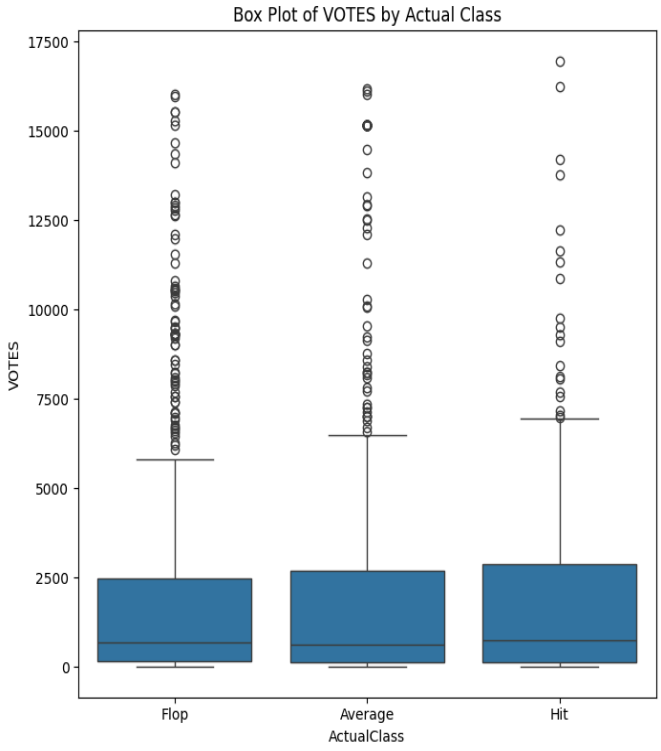
  1. **Pre-processing the Data**: Similar to Random Forest, we need to handle categorical and numerical data appropriately. For SVM, it's particularly important to scale numerical features because the algorithm is sensitive to the feature scales due to its dependence on calculating distances between data points.
  2. **Feature Engineering:** Converting text data into a numerical format using techniques like TF-IDF or word embeddings for the ONE-LINE and STARS columns. Additionally, creating new features or interactions that could help the SVM model in making accurate predictions.
  3. **Choosing the Kernel and Hyperparameters:** The choice of kernel (linear, polynomial, radial basis function (RBF), etc.) greatly affects the model's performance. Hyperparameters like C (regularization parameter) and gamma (for the RBF kernel) need to be tuned using cross-validation techniques to find the optimal values.

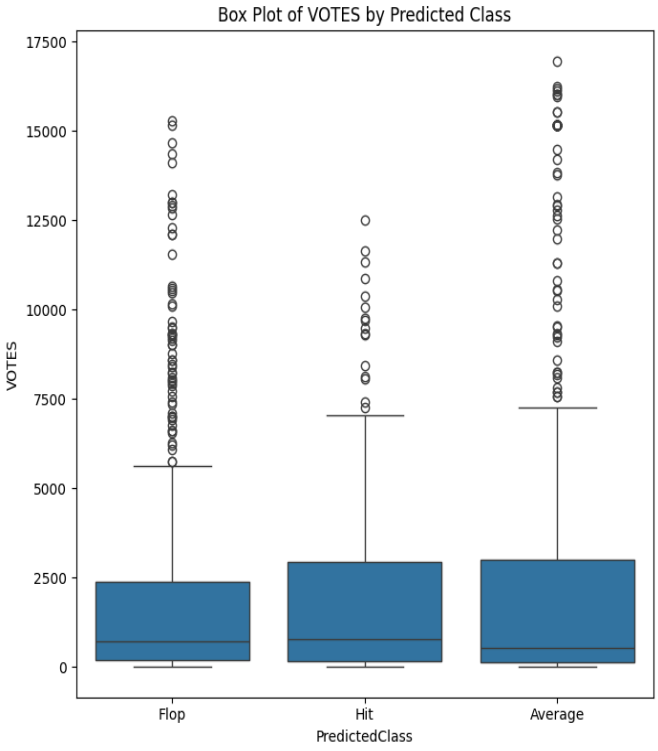
- Model Effectiveness

```
Support Vector Machine Evaluation:
              precision    recall  f1-score   support

        Flop       0.57      0.62      0.59       550
     Average       0.69      0.73      0.71       671
         Hit       0.67      0.45      0.53       260

    accuracy                           0.64      1481
   macro avg       0.64      0.60      0.61      1481
weighted avg       0.64      0.64      0.64      1481
Accuracy: 0.6401080351114112
```

The Support Vector Machine (SVM) model demonstrated competence in classifying movies into flops, average, and hits with an overall accuracy of 64%. Notably, its performance in categorizing movies as average was particularly strong, evidenced by higher precision and recall metrics in this category. This indicates that the SVM model is effectively using the dataset's features to verify relations between different levels of movie success. The ability to accurately classify average movies suggests that the model has captured essential attributes that predict moderate market performance. Such insights are crucial for filmmakers and professionals, as they offer a data-driven approach to optimizing movie production.







3. K Nearest Neighbour(KNN) Classifier

- Justification for Choosing KNN

  KNN is a good option for simple prediction problems where the relationship between features and outcomes can be captured through feature space proximity because of its inherent simplicity.
  Non-parametric Nature: KNN doesn't assume a specific form for the data distribution, making it suitable for the movie dataset where relationships between features (like GENRE and POPULARITY) and outcomes (ratings, hit status) may not follow a known distribution.

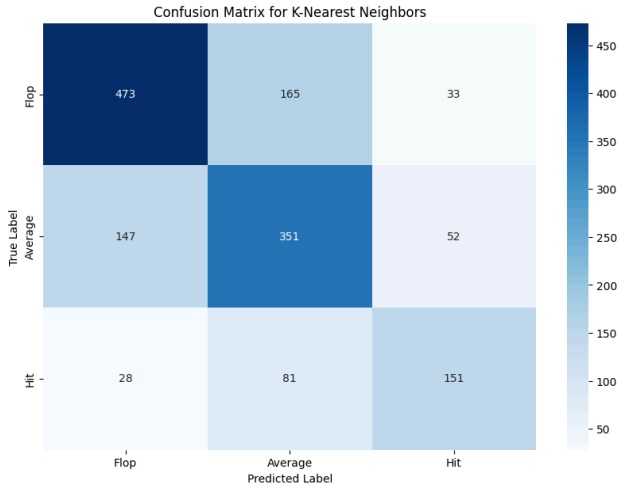
- Tuning/Training the Model

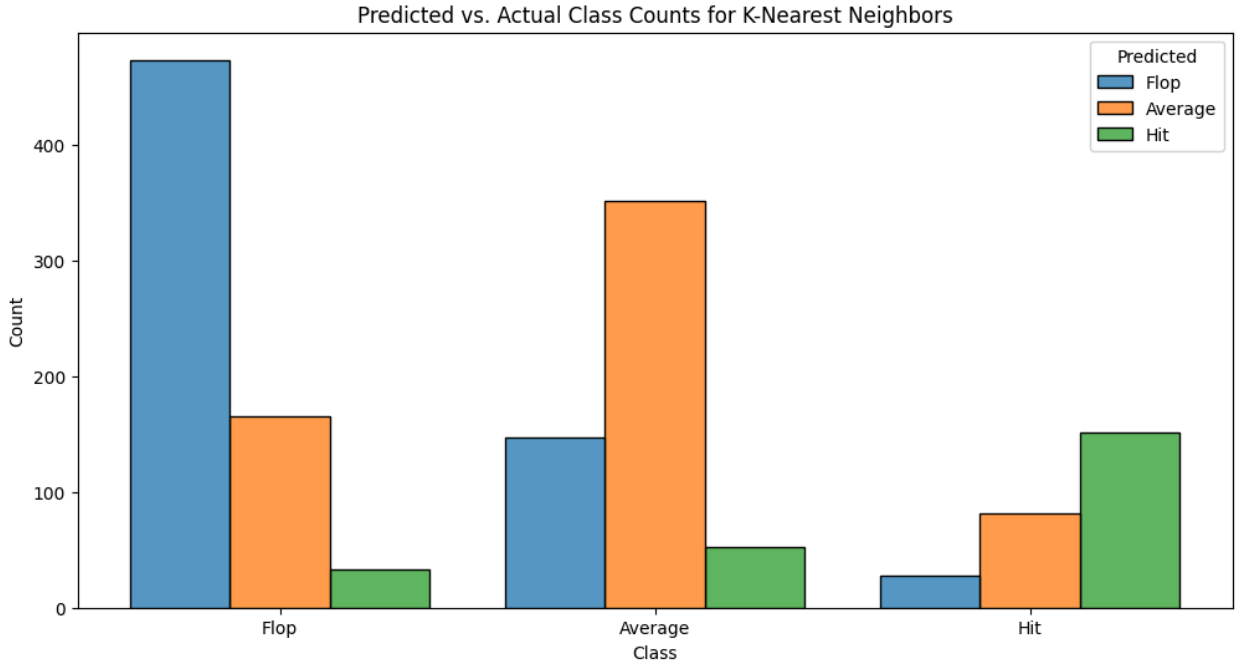
  1. **Genre Transformation:** Converted genres to multiple classifications to quantify their presence/absence, aiding KNN in recognizing patterns across different genres. By dividing into hit,flop and average status.
  2. **Year and Votes Processing:** Extracted numeric values from the year and converted votes to a numeric format, ensuring these important features contribute accurately to distance calculations.
  3. **K Selection**: Experimented with different values of K to find a balance between underfitting and overfitting.

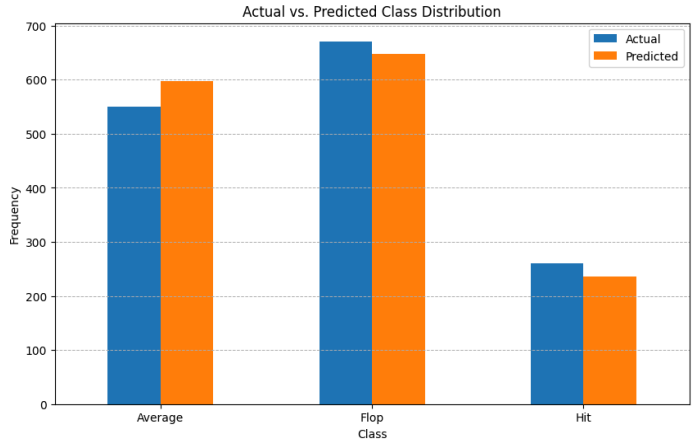
- Model Effectiveness

```
K-Nearest Neighbors Evaluation:
              precision    recall  f1-score   support

        Flop       0.59      0.64      0.61       550
     Average       0.73      0.70      0.72       671
         Hit       0.64      0.58      0.61       260

    accuracy                           0.66      1481
   macro avg       0.65      0.64      0.65      1481
weighted avg       0.66      0.66      0.66      1481

Accuracy: 0.6583389601620526
```

The K-Nearest Neighbors (KNN) model showed a solid ability to classify movies as flops, average, or hits, achieving an accuracy of around 66%. It performed well across all categories but was particularly effective in identifying average movies, with a balance between precision and recall indicating its strength in both accurately predicting and covering the scope of this category. The model's success in this classification task highlights its capability to utilize the dataset's features to make nuanced distinctions between different levels of movie success


Confusion Matrix for K-Nearest Neighbors
True Label
Predicted Label
Flop
Average
Hit
473
165
33
147
351
52
28
81
151
450
400
350
300
250
200
150
100
50


Predicted vs. Actual Class Counts for K-Nearest Neighbors
Predicted
Flop
Average
Hit
Count
Class
0
100
200
300
400


Actual vs. Predicted Class Distribution
Actual
Predicted
Frequency
Class
0
100
200
300
400
500
600
700

4. Logistic Regression

- Justification

  Binary Outcome for Movie Success Prediction: Logistic Regression is inherently suited for binary classification problems, making it an ideal choice for determining whether a movie is a hit based on specific criteria (e.g.,RATING).
  Rating as a Categorical Outcome: Though movie ratings are continuous, they can be categorized (e.g., high vs. low) for a classification approach. Logistic Regression can then be applied to predict these categories.
  VOTES: The number of votes, after normalization, helps Logistic Regression assess the popularity and audience engagement of a movie, which are crucial indicators of a movie's success.

- Tuning/Training the Model

  1. **Approach,** genres were transformed into binary features, and numerical values were extracted and normalized for features like YEAR and VOTES.
  2. **Given Logistic Regression's linear nature**, features were carefully selected and engineered to ensure they have a linear relationship with the log odds of the outcomes.
  3. Continuous features were normalized to ensure they're on a similar scale, reducing the risk of large-scale features unduly influencing the model.

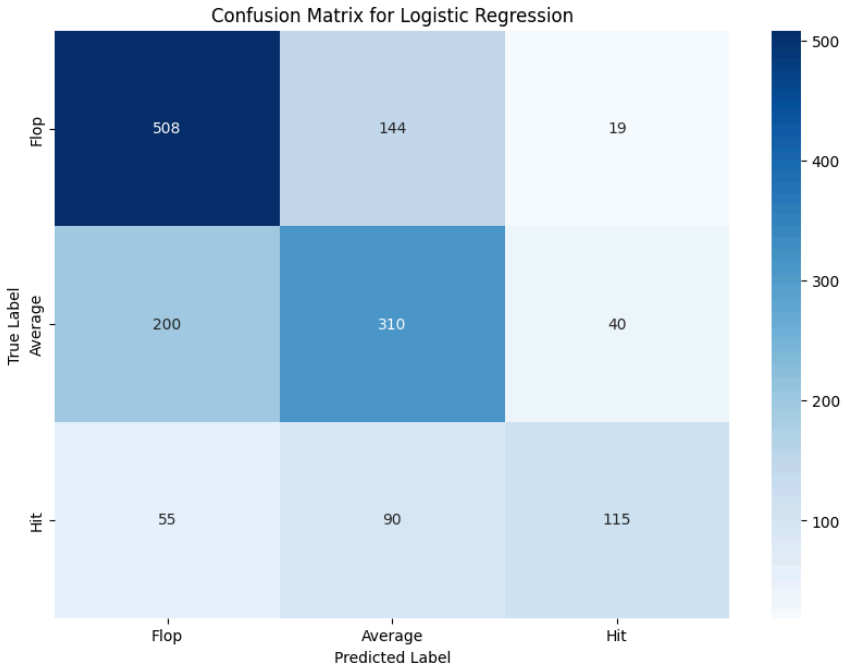
- Model Effectiveness

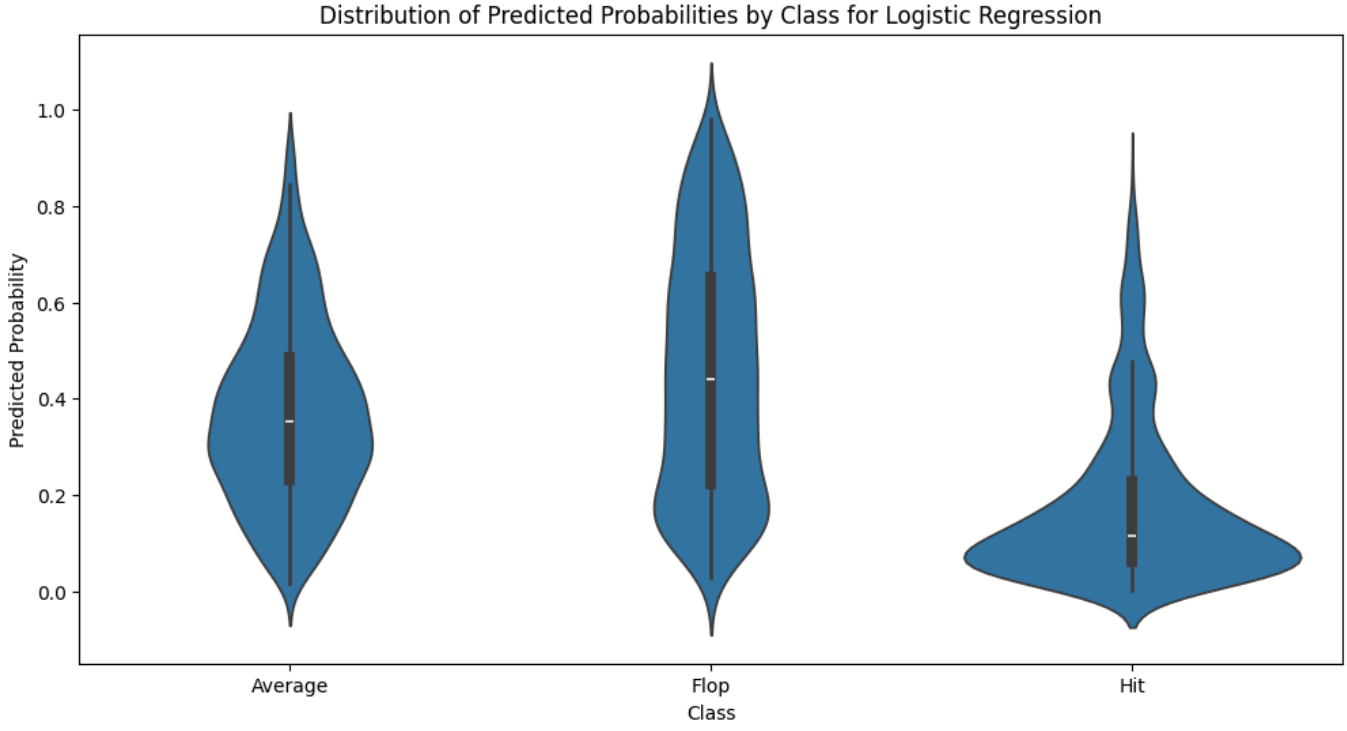
Logistic Regression Evaluation:

	precision	recall	f1-score	support
Flop	0.57	0.56	0.57	550
Average	0.67	0.76	0.71	671
Hit	0.66	0.44	0.53	260
accuracy			0.63	1481
macro avg	0.63	0.59	0.60	1481
weighted avg	0.63	0.63	0.62	1481

Accuracy: 0.6299797434166104

The Logistic Regression model demonstrated a notable ability to categorize movies into flops, average, or hits, with an accuracy rate of approximately 63%. Its performance was particularly commendable in identifying average movies, as indicated by a high precision and recall in this category, suggesting a strong capacity to accurately predict and cover a broad range of average movies.


Confusion Matrix for Logistic Regression
Flop
Average
Hit
True Label
508
144
19
200
310
40
55
90
115
Flop
Average
Hit
Predicted Label
500
400
300
200
100


Distribution of Predicted Probabilities by Class for Logistic Regression
Predicted Probability
1.0
0.8
0.6
0.4
0.2
0.0
Average
Flop
Hit
Class

5. Decision Tree Classifier

- Justification

  Handling both Numerical and Categorical Data: Our dataset comprises a mix of numerical (e.g., YEARS, VOTES) and categorical variables(ONE-LINE, GENRE). Decision tree can natively handle these variable types without the need for Extensive preprocessing, simplifying the modeling process.
  Non-linear Relationships and Interaction Effects:Decision tree do not assume Linear relationships between features and target variables.

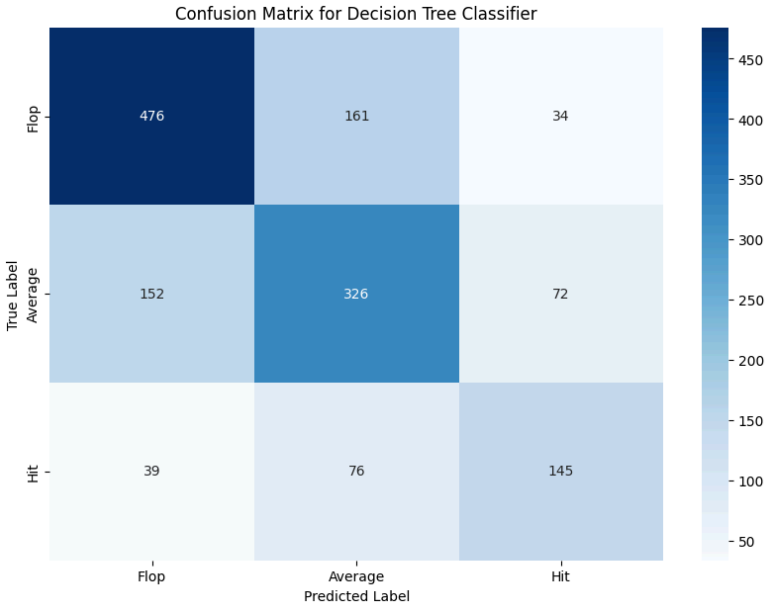
- Tuning/Training the Model

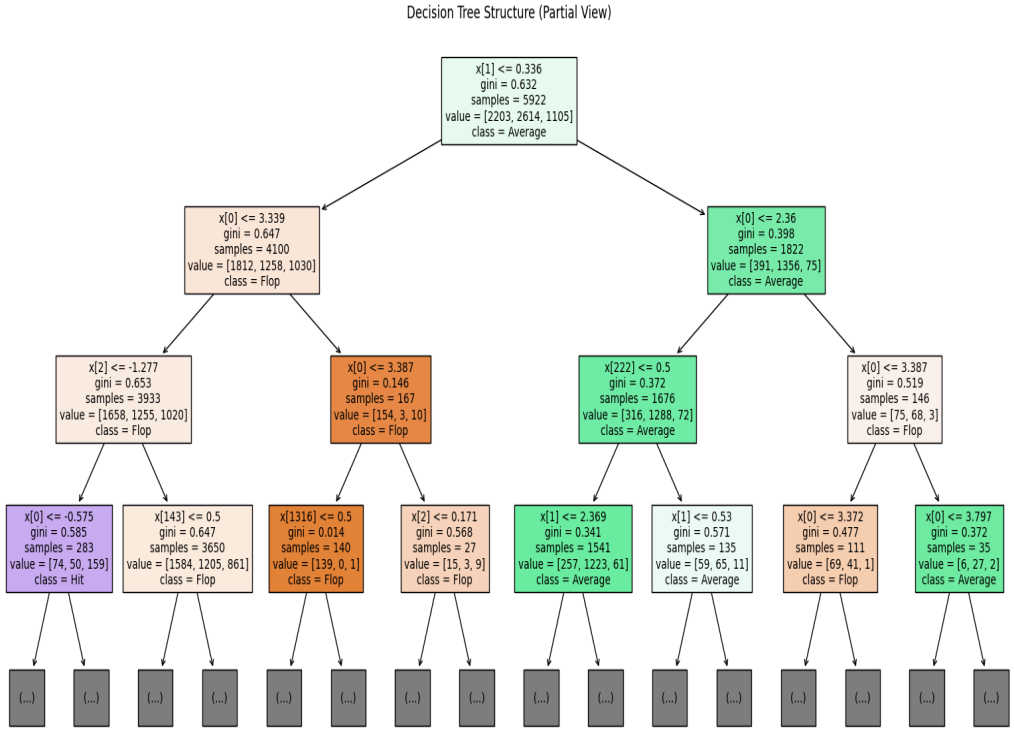
1. **Feature Preprocessing:** Critical preprocessing steps involved transforming categorical genres into binary features and normalizing numerical variables, which facilitated the Decision Tree's ability to efficiently process the dataset's multifaceted features.
2. **Pruning and Cross-Validation:** Adjusting pruning parameters, including the tree's maximum depth and the minimum samples per node, was crucial to averting overfitting. Cross-validation techniques further refined these parameters, optimizing the model's hyperparameters through iterative training and validation on segmented data folds. This approach ensured the derivation of a model configuration that adeptly balanced between model complexity and predictive performance.

- Model Effectiveness

```
Accuracy: 0.6394328156650911
Decision Tree Classifier Metrics:
              precision    recall  f1-score   support

     Average       0.58      0.59      0.59       550
        Flop       0.71      0.71      0.71       671
         Hit       0.58      0.56      0.57       260

    accuracy                           0.64      1481
   macro avg       0.62      0.62      0.62      1481
weighted avg       0.64      0.64      0.64      1481
```

The Decision Tree Classifier's performance, with an accuracy of approximately 64%, reveals its capacity to classify movies into average, flop, and hit categories effectively. The model demonstrates a balanced proficiency across these categories, as indicated by the precision and recall metrics. Notably, it achieves a comparable level of precision and recall in identifying average and hit movies





6. Gradient Booster Classification

- Justification

  Gradient Descent Optimization: At its core, GBC utilizes gradient descent on the loss function to identify the direction that minimizes prediction errors. This approach is particularly beneficial for our dataset, as it allows for the optimization of complex loss functions that can be used for specific requirements of our predictive tasks.
  Sequential Learning and Error Correction: GBC's methodology of building trees sequentially, where each tree aims to correct the errors of its predecessors, aligns with the objective of developing a highly accurate model that can adaptively learn from the nuances in the dataset.

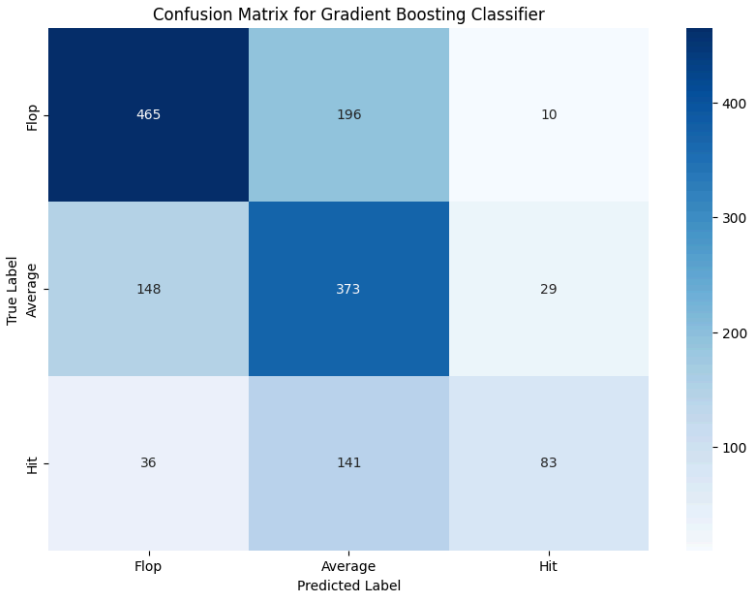
- Tuning/Training the Model

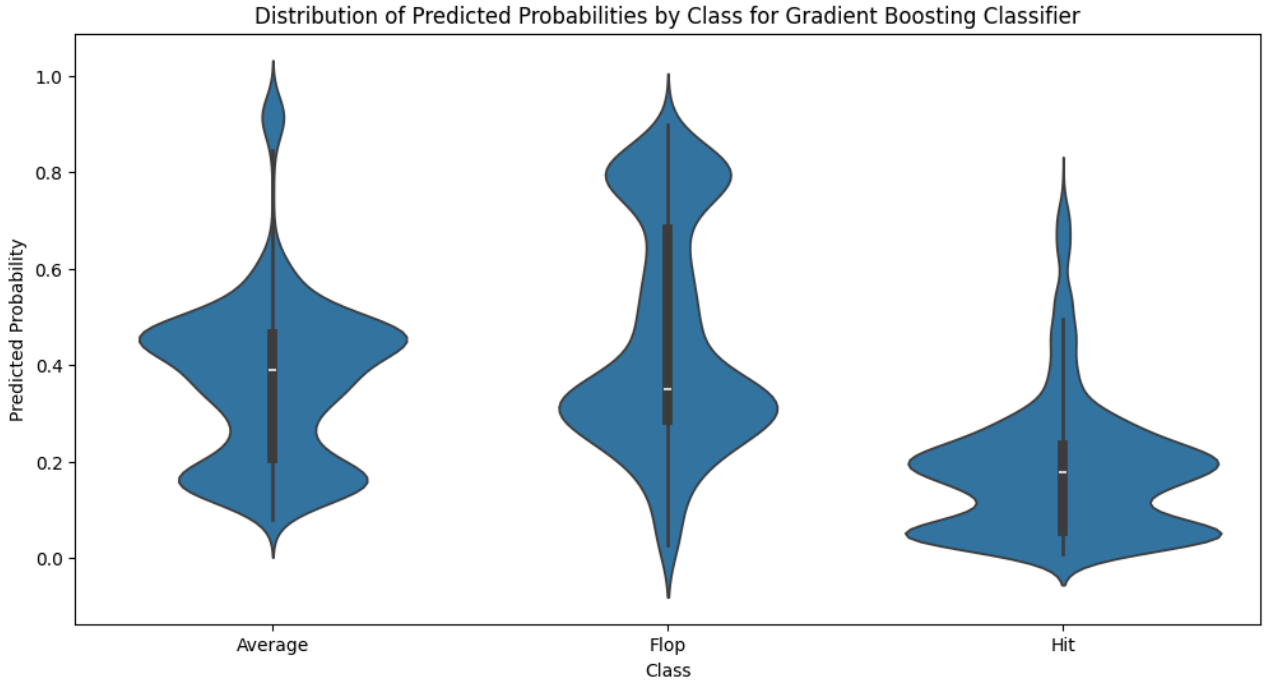
  1. **Addressing Model Variance and Bias:** Essential to the training process was the strategic management of the bias-variance trade-off, a crucial consideration in preventing the model from either underfitting (high bias) or overfitting (high variance) to the dataset. By adjusting the ensemble size (number of trees) and the learning depth of each tree, the model was meticulously calibrated to maintain an optimal balance.
  2. **Robust Model Validation**: Beyond standard cross-validation techniques, the validation process for the GBC model incorporated a layered approach that included out-of-bag (OOB) estimates and validation set performance.

- Model Effectiveness

```
Gradient Boosting Classifier Metrics:
              precision    recall  f1-score   support

     Average       0.53      0.68      0.59       550
        Flop       0.72      0.69      0.70       671
         Hit       0.68      0.32      0.43       260

    accuracy                           0.62      1481
   macro avg       0.64      0.56      0.58      1481
weighted avg       0.64      0.62      0.62      1481

Accuracy: 0.6218771100607697
```

The metrics from the Gradient Boosting Classifier provide insightful observations into its performance in classifying movies as average, flop, or hit. With an overall accuracy of approximately 62%,
A key insight is the model's higher precision in predicting flops and hits compared to average movies, suggesting that it effectively utilizes the dataset's features to identify clear indicators of movies that are likely to underperform or excel.





7. K means Clustering

- Justification

  Unsupervised Learning Approach: Unlike supervised learning models that require labeled data, K-means Clustering is an unsupervised algorithm, ideal for exploring underlying patterns in the dataset without predefined labels. “GENRE” The diverse range of genres in your dataset is a prime candidate for clustering. Ratings and Votes the inclusion of ratings and votes offers a quantitative measure of a movie's reception and popularity. By applying K-means Clustering, it's possible to identify clusters of movies that achieved similar levels of success or reception, aiding in the identification of the attributes that contribute to high ratings and votes.

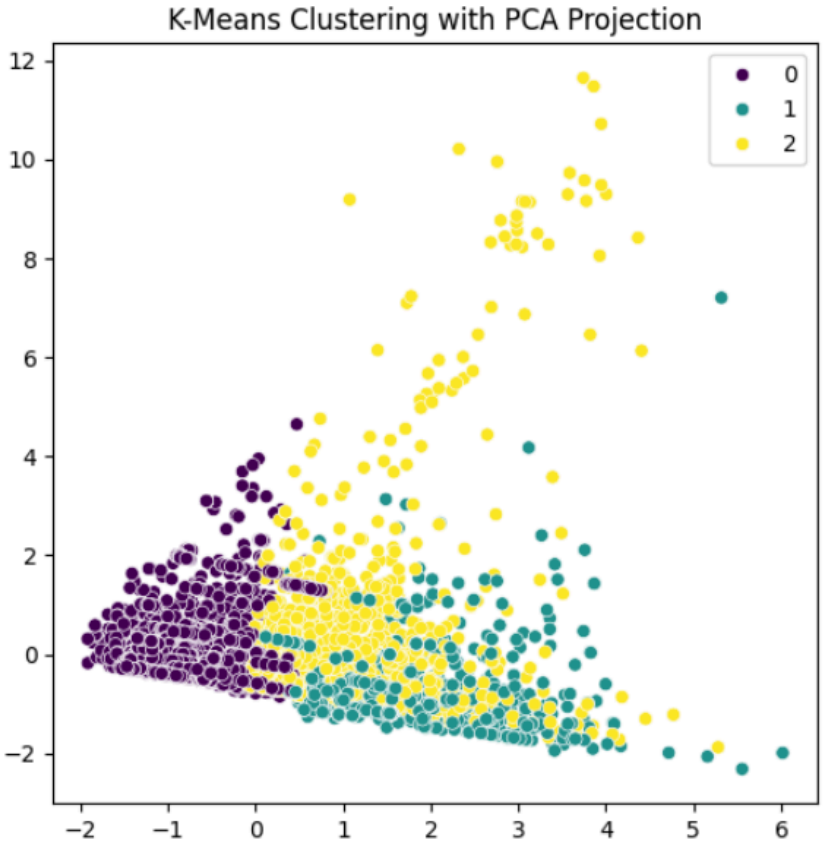
- Tuning/Training the Model

  1. **Determining the Optimal Number of Clusters**: The choice of the number of clusters (k) is pivotal in K-means Clustering. Methods such as the Elbow Method were employed to analyze the variance within clusters as a function of k, seeking a point where increasing the number of clusters does not significantly improve the variance explained by the clusters.
  2. **Cluster Analysis:** After establishing the optimal k, the K-means algorithm was applied to segment the movies into clusters based on their features. The analysis of these clusters involved examining the centroid of each cluster (i.e., the mean value of the features within a cluster) to understand the defining characteristics of the movies grouped together.

- Model Effectiveness

```
Silhouette Score for K-Means: 0.1808019894088216
Inertia for K-Means: 30783.900867871063
Homogeneity Score for K-Means: 0.07714802314421625
Completeness Score for K-Means: 0.09215586749940531
V-Measure Score for K-Means: 0.08398676452964667
```

Silhouette Score (0.1808): The Silhouette Score, which measures the cohesion and separation of the clusters, is relatively low. This suggests that while there are distinguishable clusters within the dataset, the boundaries between them may not be very distinct.Inertia (30783.9009): The inertia, representing the sum of squared distances of samples to their closest cluster center,Homogeneity Score (0.0771): This metric indicates a low homogeneity level, meaning that each cluster contains members that are not entirely similar to each other.V-Measure Score (0.084): This is the harmonic mean of homogeneity and completeness, providing an overall measure of the clustering quality.Completeness Score (0.0922): Similarly low, the completeness score signifies that all movies that are part of a true category (based on some inherent similarity)



8. DBSCAN

- Justification

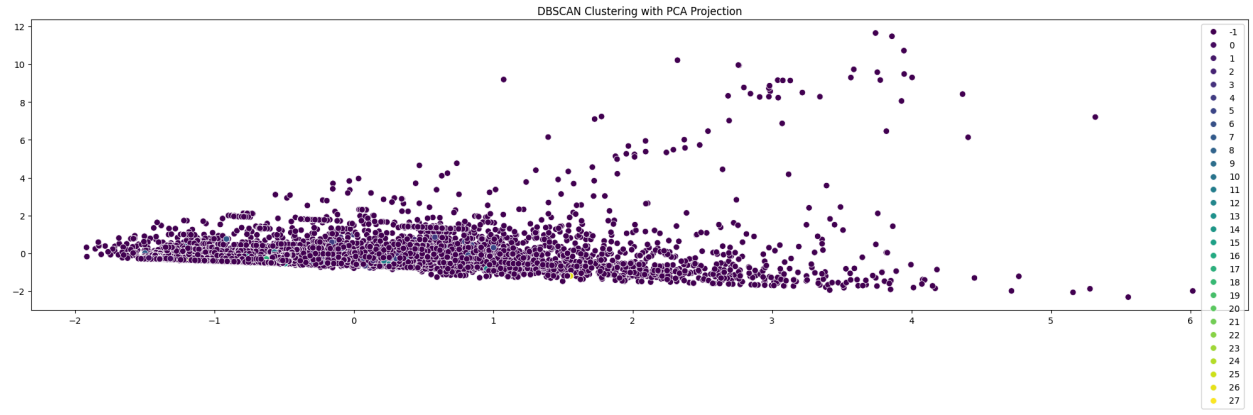
  Ability to Handle Noise: DBSCAN's inherent design to differentiate between core, border, and noise points makes it exceptionally suited for the movie dataset, where anomalies or outliers (ONE-LINE, GENRE) are notified.
  Parameter Selection over Cluster Number Determination: DBSCAN requires setting parameters (ε and minPts) instead of specifying the number of clusters a priori, an advantage for exploratory data analysis where the inherent clustering structure is unknown.

- Tuning/Training the Model

  1. **Determining ε (Epsilon) and minPts (Minimum Points):** The core of tuning DBSCAN involves determining the appropriate ε, which defines the radius of a neighborhood around a point, and minPts, the minimum number of points required to form a dense region.
  2. **Normalization and Encoding:** Critical preprocessing steps included the normalization of numerical features to prevent disproportionate influence on distance calculations and the encoding of categorical variables such as “GENRE”.

- Model Effectiveness

```
Silhouette Score for DBSCAN: 0.4931763106146411
Number of clusters found by DBSCAN: 28
Number of noise points found by DBSCAN: 6986
```

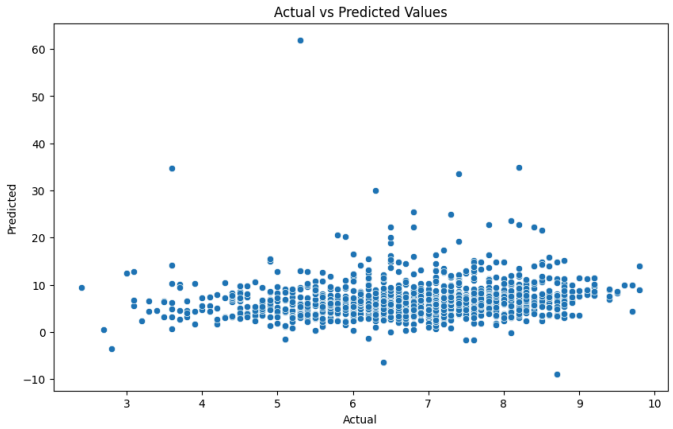


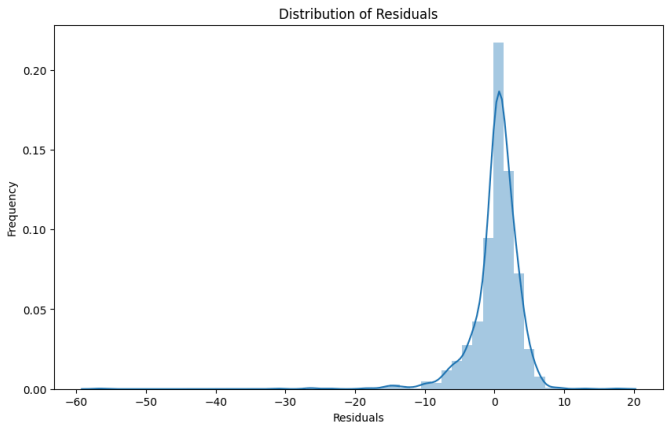
We have also trained various regression models for the data frame but due to the lack of sufficient data and also due to low measure of correlation between the numeric groups to the rating column we have not been able to get the desired results. But nonetheless, we have experimented with them and shall also keep them in mind for future considerations.

Linear regression :

Opting for linear regression for its straightforwardness, our goal was to create a foundational analysis. Through feature selection, data normalization, and cross-validation, we sought to refine the model's accuracy. However, the significant Mean Squared Error, R^2, and Mean Absolute Error metrics revealed our dataset's complexity and the linear model's limitations in capturing all details, suggesting a pivot towards more advanced models or better feature engineering might be necessary.

```
Mean Squared Error:  13.973281347427594
R2 Score:  -8.99976297878093
Mean Absolute Error:  2.274850001642607
```

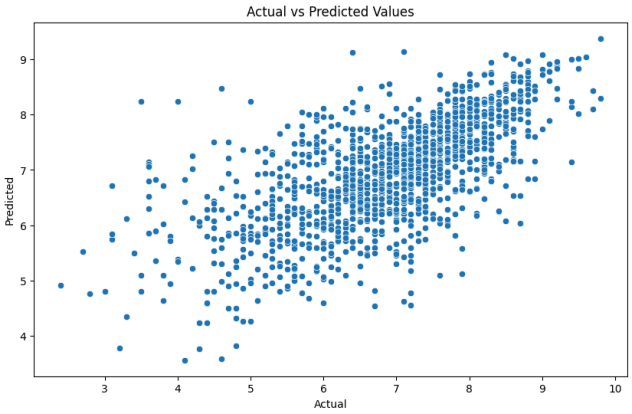


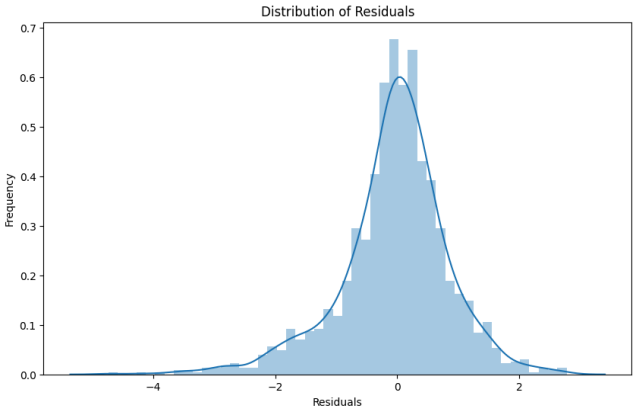


KNN Regressor

We used a K-Nearest Neighbors (KNN) regressor because of its ability to handle complex, non-linear data patterns. By carefully adjusting the model's settings, especially how it determines the 'neighborhood' of data points, we saw solid performance: our Mean Squared Error was 0.81, R2 Score was 0.42, and Mean Absolute Error was 0.64. These results suggest that the KNN model did a decent job at predicting outcomes, significantly improving upon simpler models. It effectively dealt with the data's complexities, offering insights that help refine our analysis further. The KNN model stands out for its straightforward yet powerful approach to uncovering relationships in the data.

```
Mean Squared Error:  0.8099837947332883
R2 Score:  0.4203476075089886
Mean Absolute Error:  0.642255232950709
```

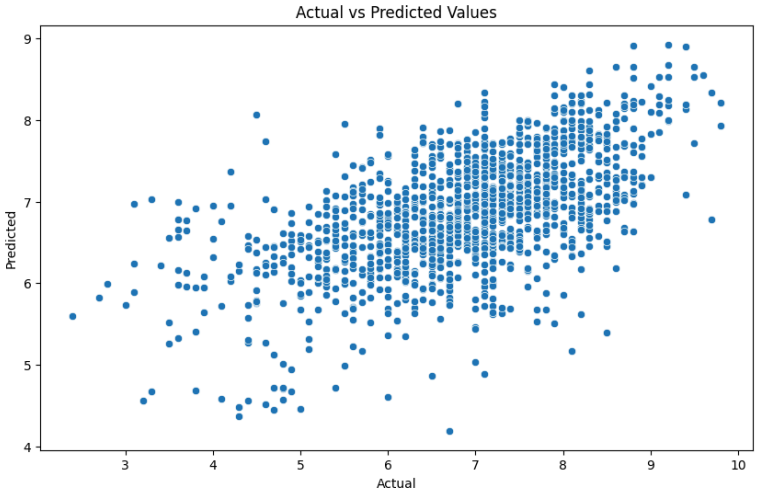


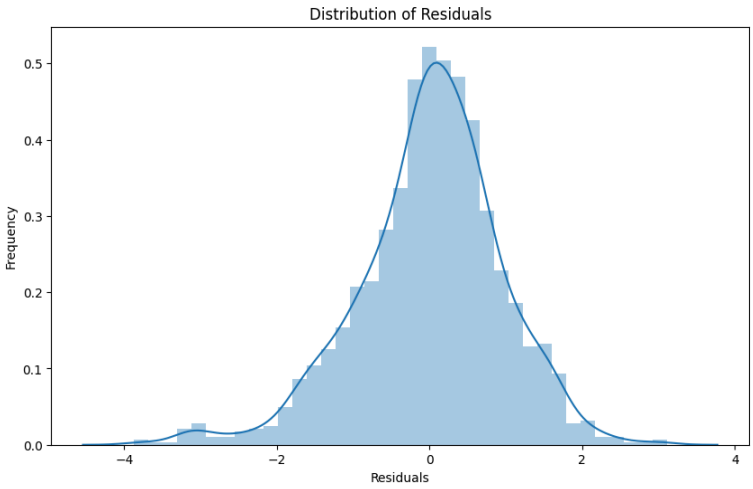


Single Vector Regressor

We used a Support Vector Machine (SVM) regressor for its ability to handle complex patterns in our data, adjusting key settings such as regularization and the kernel type. The model yielded a Mean Squared Error (MSE) of 0.902, an R2 Score of 0.354, and a Mean Absolute Error (MAE) of 0.716. These outcomes indicate a decent predictive performance by the SVM, offering an improvement over simpler models and showcasing its effectiveness in dealing with the dataset's intricacies. Although there's potential for further enhancement, the SVM regressor has proven to be a good addition to our analysis.

```
Mean Squared Error:  0.9022978668694266
R2 Score:  0.354284467576763
Mean Absolute Error:  0.7163932087063964
```

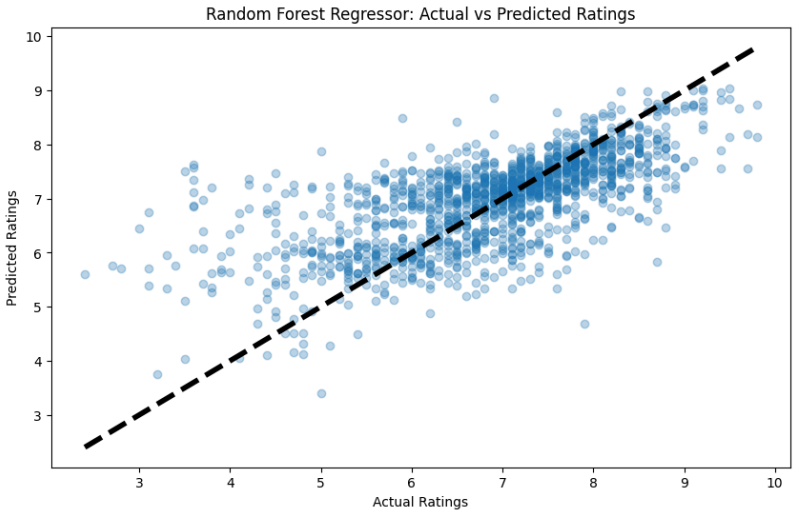




Random Forest Regressor

We used a Random Forest Regressor, known for its ability to analyze complex data through combining multiple decision trees. Adjusting its parameters resulted in a Mean Squared Error (MSE) of 0.770, an R2 Score of 0.449, and a Mean Absolute Error (MAE) of 0.619. These metrics show a good level of prediction accuracy, highlighting the model's effectiveness in tackling the dataset's challenges. The Random Forest Regressor has been good in improving our predictions, guiding future improvements in our analysis approach.

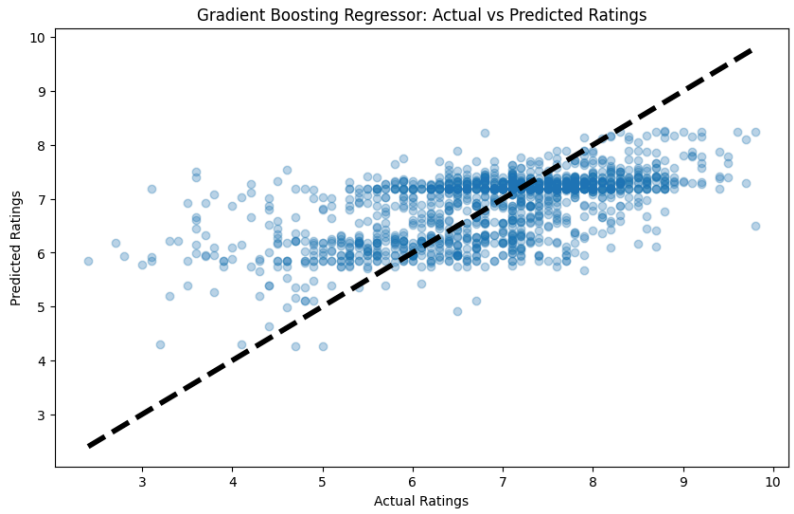
```
Random Forest Regressor
Mean Squared Error:  0.7695292876500834
R2 Score:  0.449298250682715
Mean Absolute Error:  0.6191651104466095
```



Grading Booster Regressor

We applied a Gradient Boosting Regressor, known for its technique of gradually improving predictions by fixing past mistakes. This method resulted in a Mean Squared Error (MSE) of 0.914, an R2 Score of 0.346, and a Mean Absolute Error (MAE) of 0.737. These figures suggest the model did well in making predictions, showing it can handle our complex dataset.

```
Gradient Boosting Regressor
Mean Squared Error:  0.9140862999262068
R2 Score:  0.3458482575321745
Mean Absolute Error:  0.7365332368765224
```



References :

sklearn.impute.SimpleImputer — scikit-learn 1.4.1 documentation

https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics

https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics

https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation