**Project Report | Movie Rating Prediction**

1. **Introduction**

Text

1. **Methodology**

In this section, we describe the approach taken to preprocess the data, engineer features, and build and evaluate various machine learning models for predicting IMDB movie ratings.

* 1. **Data Preprocessing**

The initial dataset comprised various features related to movies, such as director names, actor names, genres, plot keywords, and others. To prepare the data for model training, the following preprocessing steps were undertaken:

* **Data Cleaning:** Columns that were deemed irrelevant or redundant for the prediction task were removed. These included director name, genres, actor names, plot keywords, movie title, language, country, content rating, and title embedding.
* **Handling Missing Values:** Missing values in the dataset were imputed using the median values of the respective columns. This ensured that the dataset remained complete without introducing significant bias.
* **Normalization:** To ensure that all features contributed equally to the model training process, numerical features were normalized using MinMaxScaler. This scaled the features to a range between 0 and 1, which helps in improving the performance of various machine learning algorithms.
  1. **Feature Engineering**

Feature engineering involved the selection and transformation of features to enhance the predictive power of the models. The features used in the final models included:

* 1. **Model Selection and Training**

We explored a variety of machine learning models to identify the most effective approach for predicting IMDB movie ratings. The models implemented include:

* Support Vector Machine (SVM): A robust classifier that works well with high-dimensional spaces and is effective in cases where the number of dimensions exceeds the number of samples.
* Random Forest Classifier: An ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes for classification tasks. Random forests are known for their high accuracy and ability to handle large datasets with higher dimensionality.
* Gradient Boosting Classifier: This model builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. Gradient Boosting is known for its high predictive accuracy.
* AdaBoost Classifier: Another ensemble technique that combines multiple weak classifiers to create a strong classifier. The model assigns weights to instances, adapting to difficult-to-classify instances over iterations.
* Voting Classifier: An ensemble model that combines the predictions from multiple models (Random Forest, Gradient Boosting, SVM) to produce a final prediction based on majority voting. This approach leverages the strengths of each individual model to improve overall accuracy.
  1. **Evaluation**

The performance of each model was evaluated using accuracy as the primary metric. Cross-validation was employed to ensure the robustness of the models. The dataset was split into training and validation sets, with 70% used for training and 30% for validation.

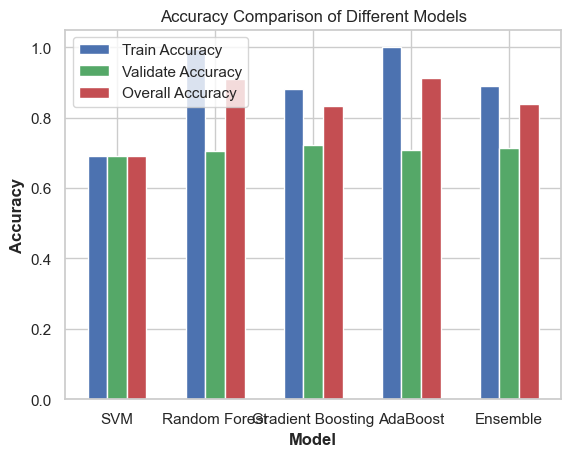
1. **Result**

The performance of the classifiers was evaluated using accuracy as the primary metric. The models were assessed on their training accuracy, validation accuracy, and overall accuracy on the entire dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Train Accuracy** | **Validate Accuracy** | **Overall Accuracy** |
| SVM | 0.690771 | 0.691796 | 0.691079 |
| Random Forest | 0.996194 | 0.703991 | 0.908455 |
| Gradient Boosting | 0.881541 | 0.721729 | 0.833555 |
| AdaBoost | 1.000000 | 0.708426 | 0.912450 |
| Ensemble | 0.890105 | 0.715078 | 0.837550 |

**Table 1 -** Classifier Performance Comparison

* Train Accuracy: The accuracy of the model on the training set. This metric indicates how well the model learned from the training data.
* Validate Accuracy: The accuracy of the model on the validation set. This metric provides an indication of the model's performance on unseen data during training.
* Overall Accuracy: The accuracy of the model on the entire dataset. This metric shows the overall performance of the model across all data points.



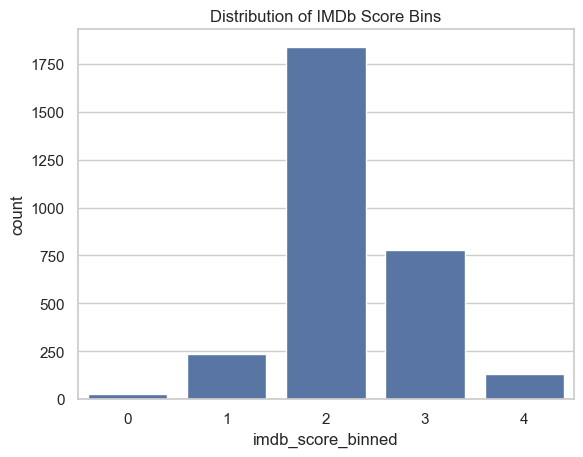
**Figure 1 -** Accuracy Comparison of Different Models

|  |  |  |
| --- | --- | --- |
| **Model** | **Mean Accuracy** | **Standard Deviation** |
| SVM | 0.6703 | 0.0222 |
| Random Forest | 0.6822 | 0.0221 |
| Gradient Boosting | 0.6960 | 0.0268 |
| AdaBoost Classifier | 0.6827 | 0.0225 |
| Ensemble Classifier | 0.6960 | 0.0212 |

**Table 2 –** Cross-Validation Result Comparison

1. **Discussion and Analysis**

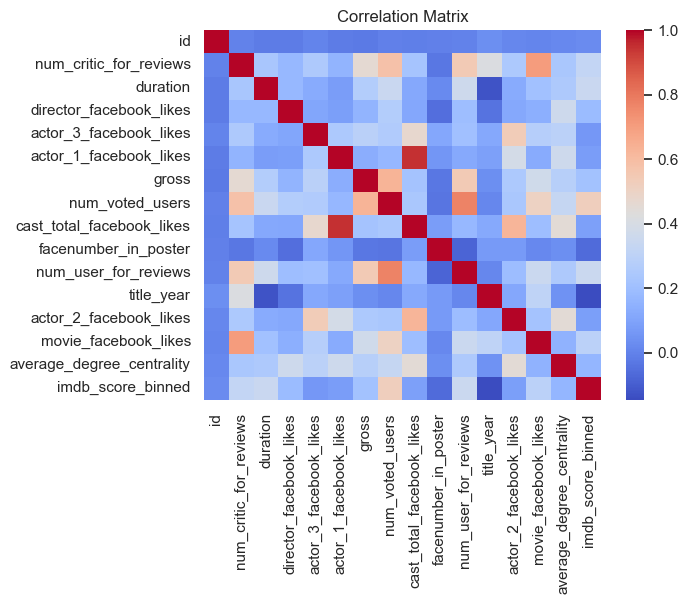
**Imbalanced Distribution of IMDb Scores**



**Figure 2 -** Distribution of IMDb Score

The histogram of IMDb score bins reveals an imbalanced distribution, with most movies clustered in the middle ratings (2 and 3). This imbalance can skew model training, causing models to favor the majority classes. To address this, I applied class weight balancing to the Random Forest (RF) and Support Vector Machine (SVM) models. The RF model showed a slight improvement in accuracy by 1.1%, whereas the SVM model's accuracy significantly dropped by 28.9%.

The slight improvement in RF can be attributed to its ensemble nature, where balancing class weights helps individual trees within the forest to focus more on minority classes, thereby improving overall generalization without heavily impacting performance on the majority class. In contrast, the significant drop in SVM's accuracy can be explained by its sensitivity to class weight adjustments. SVM relies on finding a hyperplane that maximizes the margin between classes, and introducing class weights can disrupt this margin, leading to poorer performance, especially in cases where the data is highly imbalanced.



**Figure 3 –** Correlation Matrix of Features

The correlation matrix reveals strong relationships, particularly among cast\_total\_facebook\_likes with actor\_1\_facebook\_likes, and title\_year with duration, indicating potential multicollinearity. While models like SVM are sensitive to multicollinearity, leading to performance issues, tree-based models like Random Forest, Gradient Boosting, AdaBoost, and Ensemble models are generally less affected by this issue. I trained additional SVM models removing variables with potential multicollinearity, and the result difference is negligible (<1%).

**String Features Extraction**

In an attempt to enhance model performance, I tried incorporating additional features extracted from .npy files, including actor1\_features, actor2\_features, director\_features, plot\_features, and title\_features, generated using various methods like count vectorization, Doc2Vec, and FastText. Despite these efforts, the inclusion of these pre-extracted features did not lead to any significant improvement in the models' accuracy or overall performance. This outcome suggests that these additional features either did not provide sufficient new information or introduced noise.

**Outliers**

I examined box plots for all variables to identify potential outliers. This visual inspection helped in detecting anomalies that could skew the model's performance. To address these outliers, I calculated the z-scores for each feature and filtered out rows with z-scores greater than 3.5 in the subset of movies with IMDb scores of 2 and 3. Due to the scarcity of observations with IMDb scores of 0, 1, and 4, I only filtered those with IMDb scores of 2 and 3. This cleaning process retained only the data points with z-scores below 3.5, effectively removing significant outliers. It was then recombined with the subset of movies with IMDb scores of 0, 1, and 4 to form a refined dataset for model training and evaluation.

* 1. **Support Vector Machine (SVM)**

***Theoretical Properties:*** SVMs are known for their effectiveness in high-dimensional spaces and their ability to find the optimal hyperplane that maximizes the margin between classes. This helps prevent overfitting, especially in cases where the number of features exceeds the number of samples.

***Cross-Validation Results:*** The SVM model achieved a mean cross-validation accuracy of 67.03% with a standard deviation of 2.22%. The relatively close values between training, validation, and overall accuracies (69.08%, 69.18%, and 69.11%, respectively) indicate that the SVM model did not overfit the training data, as expected.

***Error Analysis:*** Figure 2 indicates that the model has a high accuracy for the most common class (rating 2). However, there are notable misclassifications, especially for ratings 3 and 4, demonstrating difficulty in distinguishing between similar classes. This is consistent with the theoretical properties of SVM, which can struggle with imbalanced datasets.

* 1. **Random Forest**

***Theoretical Properties:*** Random Forests are ensemble learning methods that reduce variance through averaging multiple decision trees. However, they can overfit if the number of trees or the depth of trees is too high.

***Cross-Validation Results:*** The Random Forest model achieved a mean cross-validation accuracy of 68.22% with a standard deviation of 2.21%. The significant difference between training accuracy (99.71%) and validation accuracy (71.06%) indicates overfitting.

***Error Analysis:*** Figure 3 shows that while the model correctly identifies many instances of the most frequent class (rating 2), it also misclassifies a considerable number of instances in classes 3 and 4. This suggests a bias towards the majority class and difficulties with less frequent classes.

* 1. **Gradient Boosting**

***Theoretical Properties:*** Gradient Boosting sequentially builds models to correct errors made by previous models, leading to improved accuracy. However, it can be prone to overfitting, especially if the number of boosting stages is too high.

***Cross-Validation Results:*** The Gradient Boosting model achieved a mean cross-validation accuracy of 69.60% with a standard deviation of 2.68%. The moderate difference between training accuracy (81.30%) and validation accuracy (72.17%) suggests some overfitting but still generalizes well to the validation set.

***Error Analysis:*** Figure 4 shows a better balance across all classes compared to Random Forest, but still has misclassifications, especially in distinguishing between ratings 2 and 3. This indicates improved generalization but room for improvement in handling class similarities.

* 1. **AdaBoost Classifier**

***Theoretical Properties:*** AdaBoost combines multiple weak classifiers to form a strong classifier. It is adaptive and focuses on difficult-to-classify instances, which can lead to overfitting if the model learns the training data too well.

***Cross-Validation Results:*** The AdaBoost model achieved a mean cross-validation accuracy of 68.27% with a standard deviation of 2.25%. The perfect training accuracy (100.00%) and lower validation accuracy (69.62%) indicate significant overfitting.

***Error Analysis:*** Figure 5 demonstrates high accuracy for the majority class (rating 2) but shows significant misclassifications for classes 3 and 4. This further highlights the model's overfitting issue, as it fails to generalize well to unseen data.

* 1. **Ensemble Classifier**

***Theoretical Properties:*** Ensemble methods combine the strengths of individual models to improve overall performance. This approach reduces bias and variance, leading to better generalization.

***Cross-Validation Results:*** The Ensemble model achieved a mean cross-validation accuracy of 69.60% with a standard deviation of 2.12%. The closer training accuracy (83.11%) and validation accuracy (72.17%) indicate better generalization compared to individual models.

***Error Analysis:*** Figure 6 shows a balanced performance, reducing the bias and variance observed in individual models. It provides more accurate predictions across all classes, demonstrating the effectiveness of combining different classifiers.

1. **Conclusion**

Text

1. **References**

Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*, volume 4. Springer.