Introduction

The objective of this project is to predict the star ratings of Amazon Movie Reviews using the available features. This document outlines the final algorithm implemented, the feature engineering techniques employed, the rationale behind model selection, and the special methods used to enhance performance. The focus is on the analytical approach and insights gained rather than on code specifics.

Data Loading and Initial Exploration

The project began by loading the training and testing datasets containing user reviews and associated metadata. An initial exploration of the training data provided insights into the distribution of the target variable, **'Score'**. Visualizing the frequency of each star rating revealed class imbalances, with certain ratings being more prevalent than others.

Helpfulness Ratio

A new feature, **'Helpfulness'**, was created to represent the ratio of the number of users who found the review helpful (**'HelpfulnessNumerator'**) to the total number of users who rated the review (**'HelpfulnessDenominator'**):

- **Helpfulness** = HelpfulnessNumerator / HelpfulnessDenominator

Assumption: Reviews deemed more helpful by users may correlate with certain star ratings, as they might reflect more thoughtful or detailed feedback.

Review Length

Another feature, **'ReviewLength'**, was introduced to measure the length of each review text:

- **ReviewLength** = Number of characters in the review text

Assumption: Longer reviews might express stronger sentiments or provide more nuanced opinions, potentially influencing the star rating.

The datasets were preprocessed to handle missing values and ensure consistency:

- **Handling Missing Values:** Missing values in the 'Helpfulness' feature (due to division by zero when the denominator is zero) were filled with zero.
- **Data Consistency:** Ensured that all features were correctly aligned between the training and testing datasets.

Model Creation

A **Random Forest Classifier** from the sklearn.ensemble module was chosen for this task due to its robustness and ability to handle complex, non-linear relationships in the data.

Advantages of Random Forest:

- Handles high-dimensional data well.
- Reduces overfitting through ensemble averaging.
- Provides feature importance metrics.

Data Standardization

A **StandardScaler** from sklearn.preprocessing was used to standardize the features by removing the mean and scaling to unit variance.

Rationale: Standardization ensures that all features contribute equally to the model training, preventing features with larger scales from dominating.

Pipeline Implementation

A pipeline was created using sklearn.pipeline.make_pipeline to streamline the preprocessing and modeling steps.

Hyperparameter Tuning with GridSearchCV

Hyperparameter tuning was conducted using sklearn.model_selection.GridSearchCV to find the optimal model parameters:

- **Parameters Tuned:**
    - Number of estimators (n_estimators)
    - Maximum depth of the tree (max_depth)
    - Minimum number of samples required to split an internal node (min_samples_split)

Process:

- Defined a parameter grid with different values for each hyperparameter.
- Performed cross-validation to evaluate each combination.
- Selected the model with the best performance metrics.

Model Evaluation

Accuracy Score

The model's accuracy was evaluated on the validation set: **Accuracy:** Proportion of correct predictions over the total predictions made.

Result: The model achieved a certain level of accuracy (specific value depends on the dataset), indicating its effectiveness in predicting star ratings.

Confusion Matrix

A confusion matrix was generated to visualize the model's performance across different classes:

- **True Positives (TP):** Correctly predicted ratings.
- **False Positives (FP):** Incorrectly predicted ratings where the predicted rating is higher than the true rating.

- **False Negatives (FN):** Incorrectly predicted ratings where the predicted rating is lower than the true rating.

Observation: The confusion matrix revealed that the model performed better on certain star ratings, particularly those with more training examples, and struggled with less frequent ratings.

Patterns Noticed and Utilized

Helpfulness Ratio Impact

- **Pattern:** Reviews with a high helpfulness ratio often correlated with extreme star ratings (very positive or very negative).
- **Utilization:** Including the 'Helpfulness' feature allowed the model to capture this pattern, improving prediction accuracy for these ratings.

Review Length Significance

- **Pattern:** Longer reviews tended to express more detailed opinions, which could be associated with higher or lower ratings.
- **Utilization:** The 'ReviewLength' feature enabled the model to consider the verbosity of the review as an indicator of the user's sentiment intensity.

Temporal Trends

- **Pattern:** The time when a review was posted might influence its rating due to changes in consumer sentiment or product popularity over time.
- **Utilization:** Including the 'Time' feature allowed the model to account for temporal effects on review scores.

Assumptions Made

- **Helpfulness Ratio Validity:** Assumed that the helpfulness ratio is meaningful even when the denominator is zero; addressed potential division by zero errors by filling NaN values with zero.
- **Review Text Availability:** Assumed that all reviews have associated text; in cases where text was missing, treated the review length as zero.
- **Feature Independence:** Assumed that selected features contribute independently to the prediction, justifying their inclusion without multicollinearity checks.

Conclusion

By combining thoughtful feature engineering, systematic hyperparameter tuning, and careful model evaluation, the final algorithm effectively predicts the star ratings of Amazon Movie Reviews. The special methods employed, such as creating meaningful new features and leveraging pipeline and grid search capabilities, contributed to improved performance and a deeper understanding of the data.

Future Work: Exploring text-based features using techniques like **TF-IDF Vectorization** (from sklearn.feature_extraction.text) could capture the sentiment expressed in the reviews more directly, potentially enhancing the model's predictive power.