Amazon Movie Review Rating Prediction

Project Overview

The primary goal was to develop a classification model that accurately predicts the star rating (Score) based on the provided features. The project involved extensive data preprocessing, feature engineering, and several machine learning models with hyperparameter tuning.

Final Algorithm and Steps

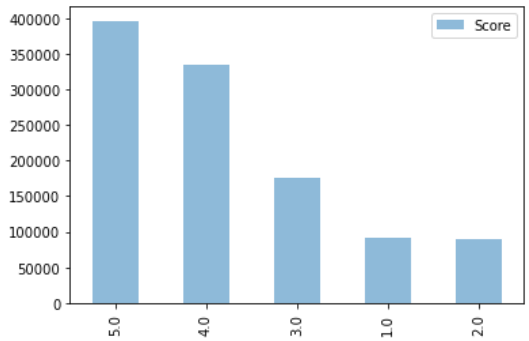
1. Data Preprocessing

Effective data preprocessing was critical given the size and complexity of the dataset. The dataset had 1.7 million rows, and key columns included:

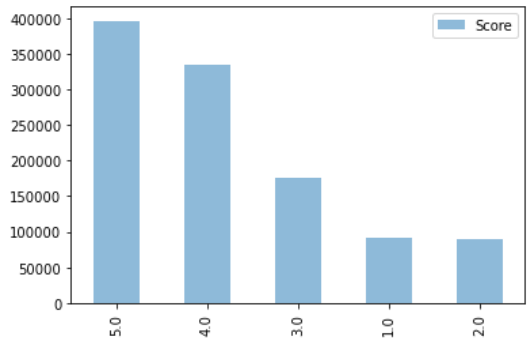
- Text (Full review text)
- Summary (A summary of the review)
- Helpfulness Numerator & Helpfulness Denominator (metrics about the usefulness of the review)
- Time (Timestamp of the review)
- Score (Star rating to be predicted)

Key preprocessing steps:

- **Handling Missing Values and Cleaning:** The Text and Summary columns contained missing values. I replaced missing entries with empty strings since they are essential for text-based feature extraction. And clean_text function is created to keep only text and in lower case.
- **Down-sampling the Dominant Class**: Since the 5-star rating was overrepresented in the dataset (almost 50% of the data), it could bias the model. To address this, I down sampled the 5-star reviews to balance the dataset by only taken 50% of the original 5-star entries.



Before Down-sampling



After Down-sampling

2. Feature Engineering

- **Helpfulness Ratio**: as HelpfulnessNumerator / HelpfulnessDenominator. This feature captures how helpful users found the review.
- **Review Length**: The length of the review text was captured as a feature (ReviewLength)
- **Year Extraction**: From the Time column (given in Unix timestamp format)
- **Textual Info**: Review text and summary are combined to create a more informative text feature and create a stronger signal for prediction

3. Feature Vectorization

Since the review text and summary were in raw form, I used **TF-IDF vectorization** to convert the textual data into numerical features. It was done both separately and combined for Text and Summary.

- **TF-IDF for Text**:
    - Set max_features to 5000 to limit the number of terms and reduce memory overhead.
    - Removed common stopwords (e.g., "the", "is") using NLTK's English stopword list to ensure only meaningful words contribute to the features.
- **TF-IDF for Summary**:
    - Similar but limited max_features to 2000 due to the shorter nature of summaries.
- **TF-IDF for Textual Info**
    - Created after initial modelling since text and summary are both strong predictors, set max_features to 5000.

4. Scaling and Combining Features

Numeric features (Helpfulness, ReviewLength, and Year) were standardized using StandardScaler to provide a uniform range. After vectorizing text and scaling numerical features, all components were combined using scipy.hstack(), resulting in a sparse matrix representation that optimized memory usage.

5. Model Selection and Training

For efficiency, I use the subsample includes **10%** of the X_train after 80-20 split for initial modeling, and and performed 5-fold cross-validation on various models (measured on Accuracy):

- **K-Nearest Neighbors (KNN)**: **0.5722**
- **Decision Tree**: **0.5701**
- **Random Forest**: **0.5944**
- **Logistic Regression**: **0.6301**

Final Model - Logistic Regression: Logistic Regression emerged as the most effective model, achieving superior accuracy and faster training times compared to tree-based models.

6. Hyperparameter Tuning

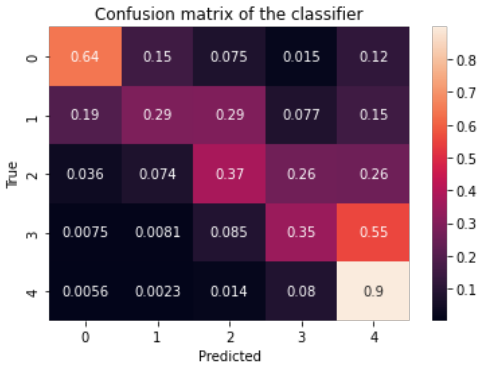
After selecting Logistic Regression, I explored two configurations:

- **Model 1:** *Down-sampled, with separate Text and Summary features, using Grid Search for hyperparameter tuning.*
    - **Objective**: Balance class distribution by down-sampling the 5-star class, aiming to prevent the model from being biased towards the most frequent score.
    - **Approach**: Applied Grid Search with cross-validation on a balanced subsample (50%)

```
param_grid_log_reg = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization strength
    'penalty': ['l1', 'l2', 'elasticnet'],  # Types of regularization
    'solver': ['liblinear', 'saga', 'lbfgs'],  # Solvers to explore
    'l1_ratio': [0.5],  # Only used if penalty is elasticnet
    'max_iter': [100, 200, 500]  # Number of iterations for convergence
}
```

    - **Insights:**
        - This configuration achieved 0.5953 accuracy on X_test, while achieving 0.6374 on the Kaggle test set. The slightly lower accuracy on X_test may be due to the down-sampling, which limits overall information in the training set.
        - Higher performance on the Kaggle test set suggests that the larger dataset helped the model capture diverse patterns, balancing class bias while improving generalization.

- **Model 2**: *No down-sampling, combined Textual Info feature, without hyperparameter tuning.*
    - **Objective**: Retain the full, imbalanced dataset to leverage all available information, with the goal of maximizing the 5-star prediction accuracy.
    - **Approach**: Combined Text and Summary into a single Textual Info feature for efficiency and skipped tuning to speed up training.
    - **Insights**:
        - The combined feature space allowed the model to train faster and provided a simplified structure without extensive hyperparameter tuning.
        - Model 2 showed stable performance across the Kaggle test set, reflecting the imbalanced nature of the data. High accuracy in predicting 5-star reviews, a large portion of the data, contributed to higher overall accuracy on the Kaggle grading.



References

1. Scikit-Learn Developers. *RandomForestClassifier.* Retrieved from the official documentation: https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

2. Scikit-Learn Developers. *Logistic Regression.* Retrieved from the official documentation: https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html.

3. Scikit-Learn Developers. *GridSearchCV*. Retrieved from the official documentation: https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.GridSearchCV.html.