Amazon Movie Reviews Rating Prediction

Name: Min Sung Park

1 Introduction

The objective of this project is to predict star ratings for Amazon movie reviews using machine learning techniques, specifically focusing on a Random Forest Classifier. The dataset comprises user reviews with associated metadata, and the goal is to leverage this information to build a predictive model without utilizing neural networks or deep learning methods.

2 Data Exploration and Preprocessing

2.1 Dataset Overview

I have two primary datasets:

- **train.csv**: Contains 1,697,533 unique reviews with associated star ratings and metadata.
- **test.csv**: Contains 212,192 unique review IDs for which I need to predict star ratings.

2.2 Initial Exploration

I began by loading the datasets and examining their structure:

```
trainingSet = pd.read_csv("./data/train.csv")
testingSet = pd.read_csv("./data/test.csv")
```

A distribution plot of the 'Score' feature revealed a skew towards higher ratings, indicating class imbalance.

2.3 Data Preprocessing

To prepare the data for modeling, I performed the following steps:

- **Handling Missing Values**: Filled missing 'Text' and 'Summary' fields with empty strings.
- **Feature Engineering**: Created a 'Helpfulness' ratio by dividing *HelpfulnessNumerator* by *HelpfulnessDenominator*, handling division by zero and filling NaN values with zero.
- **Text Features**: Calculated text lengths and word counts for both 'Text' and 'Summary' fields.

3 Feature Selection

I selected the following features for the model:

- **Numerical Features**: 'HelpfulnessNumerator', 'HelpfulnessDenominator', 'Time', 'Helpfulness'.
- **Text Features**: 'Text'.

4 Text Vectorization and Dimensionality Reduction

4.1 TF-IDF Vectorization

I transformed the 'Text' data into numerical form using TF-IDF vectorization with a maximum of 10,000 features to capture the most significant terms while managing computational load.

```
text_vectorizer = TfidfVectorizer(max_features=10000)
text_vectorizer.fit(X_train['Text'])
```

4.2 Truncated SVD

To reduce dimensionality and capture latent semantic information, I applied Truncated Singular Value Decomposition (SVD) with 100 components.

```
svd = TruncatedSVD(n_components=100, random_state=42)
svd.fit(X_train_text_tfidf)
```

5 Combining Features

I combined the numerical and text features into a single dataset for training.

```
X_train_final = np.hstack((X_train_num, X_train_text_svd))
```

6 Model Training

6.1 Random Forest Classifier

I trained a Random Forest Classifier with 300 estimators.

```
model = RandomForestClassifier(n_estimators=300, random_state=42, n_jobs=-1)
model.fit(X_train_split, Y_train_split)
```

Special Trick: Setting `n_jobs=-1` allows parallel computation, utilizing all available CPU cores to speed up the training process.

7 Model Evaluation

7.1 Validation Accuracy

The model achieved a validation accuracy of approximately 58%. While this is below the desired target of 60%, it indicates that the model is learning meaningful patterns from the data.

7.2 Confusion Matrix

I plotted a confusion matrix to analyze the model's performance across different classes. The matrix showed better performance on majority classes (ratings 4 and 5) and some confusion between adjacent ratings, which is expected due to the subjective nature of star ratings.

8 Assumptions, Special Tricks, and Thought Process

8.1 Assumptions

- **Feature Importance**: I assumed that the 'Helpfulness' ratio and text-related features are significant predictors of star ratings.

- **Class Imbalance**: Recognizing the skewed distribution of ratings, I proceeded without specific handling for class imbalance due to computational constraints.

8.2 Special Tricks and Thought Process

Throughout the project, I implemented several techniques to improve the model's performance:

- **Caching Processed Data**: To save time on subsequent runs, I saved the processed datasets to CSV files. This way, I avoided reprocessing the data every time the script ran.
- **Selective Feature Inclusion**: Initially, I considered including 'ProductId' and 'UserId' as features. However, due to their high cardinality and the potential introduction of noise, I decided to exclude them. This helped reduce overfitting and simplified the model.
- **Dimensionality Reduction with Truncated SVD**: Applying Truncated SVD reduced the dimensionality of the TF-IDF matrix, capturing the most significant semantic information while reducing computational complexity.
- **Parallel Processing**: By setting `n_jobs=-1` in the Random Forest Classifier, I utilized all available CPU cores, significantly speeding up model training.
- **Balancing Model Complexity and Performance**: I chose to limit the maximum features in TF-IDF and the number of components in SVD to manage computational resources while retaining essential information.

8.3 Patterns Noticed and Utilized

During the exploration phase, I noticed the following patterns:

- **Skewed Rating Distribution**: The majority of reviews had high star ratings (4 and 5 stars). This imbalance could bias the model towards predicting higher ratings.
- **Correlation Between Text Length and Rating**: Longer reviews often provided more detailed opinions, which could correlate with more extreme (high or low) ratings.
- **Helpfulness Ratio as an Indicator**: Reviews with higher helpfulness ratios might be more informative or impactful, potentially influencing the star rating.

I utilized these observations to focus on specific features and adjust the model accordingly.

9 Conclusion

Despite limitations and constraints, the Random Forest Classifier demonstrated reasonable performance in predicting star ratings. The project highlighted the importance of data preprocessing, feature engineering, and thoughtful model selection.

In future iterations, I would consider:

- **Hyperparameter Tuning**: Implementing techniques like Grid Search to find the optimal model parameters.
- **Handling Class Imbalance**: Applying methods such as resampling or using class weights to address the skewed rating distribution.
- **Advanced Text Processing**: Incorporating n-grams, stop-word removal, or sentiment analysis to enrich the text features.

Final Thoughts: This project reinforced the value of understanding data patterns and making informed decisions throughout the modeling process. Balancing computational efficiency with model performance was crucial in developing an effective predictive model.