Amazon Review Prediction Report

Introduction

The objective of this project is to predict Amazon movie review star ratings using XGBoost based on metadata and review text. By accurately predicting review scores, we aim to classify sentiment effectively without neural networks. Since the data is massive, ChatGPT suggest to use XGBoost, which efficiency on large datasets and its ability to handle structured data made it a suitable choice.

XGBoost

XGBoost is an implementation of gradient-boosting decision trees. It builds an ensemble of decision trees sequentially, where each new tree attempts to correct the errors of previous ones. XGBoost includes regularization techniques (L1 and L2) to prevent overfitting and supports parallel processing, making it a popular choice for competitive machine learning tasks. Its flexibility allows fine-tuning of various parameters, making it adaptable to diverse prediction tasks like classification, regression, and ranking.

Dataset Description

The dataset includes 1,697,533 reviews, each labeled with a score and associated metadata fields (ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, Text). Some scores are missing, requiring prediction. The test set contains 212,192 reviews with missing Score values to be predicted.

Methodology Constraints

Only traditional machine learning approaches (excluding deep learning) were allowed. XGBoost was chosen for its flexibility and ability to manage large, sparse datasets effectively, allowing us to focus on feature engineering for enhanced predictive accuracy.

Data Preparation and Feature Engineering

Data Preparation

Our data preparation process included handling missing values, engineering text features, computing aggregate statistics, and scaling numeric features.

- **Missing Values**: HelpfulnessNumerator and HelpfulnessDenominator were filled with 0 and 1, respectively, to prevent division errors. We calculated HelpfulnessRatio and dropped the original columns. Missing Text values were replaced with empty strings.

Text-Based Features

- **ReviewWordCount** and **ReviewCharCount** provided insights into review length, while **AvgWordLength** and **UniqueWordRatio** captured vocabulary diversity and text complexity.
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: TF-IDF was applied to the Text column to capture the importance of words within each review. This method transforms text data into numerical features by emphasizing words that are unique to a particular review, enhancing the model's ability to understand content-specific context.

Sentiment Analysis

Using TextBlob, we extracted **Polarity** (sentiment from -1 to 1) and **Subjectivity** (0 to 1) from the Text column, adding indicators of tone and emotional content.

Helpfulness Indicator

We created an IsHelpful binary feature based on whether HelpfulnessRatio was greater than or equal to 0.5, classifying reviews as helpful or not.

Aggregate Statistics by Product and Reviewer

We calculated **ProductAvgRating** and **ProductReviewCount** for each ProductId and **ReviewerAvgRating** and **ReviewerReviewCount** for each UserId to capture trends across products and reviewers.

Interaction Terms

To capture complex relationships, we created interaction terms such as HelpfulnessRatio * Polarity, HelpfulnessRatio * Subjectivity, and Polarity * ReviewerReviewCount.

Scaling Numeric Features

Using StandardScaler, we normalized all numeric features, including interaction terms, to ensure consistency and stability during model training.

Modeling Approach

Data Preparation with DMatrix

We converted both X_train and X_test into XGBoost's DMatrix format to optimize performance and memory usage. Zero-based indexing was applied to labels for compatibility with XGBoost's multi-class classification objective.

Hyperparameter Tuning

Using RandomizedSearchCV, we explored various hyperparameters efficiently to maximize classification accuracy while minimizing overfitting. Key parameters included:

- **n_estimators**: Number of boosting rounds, tested from 100 to 500.
- **learning_rate**: Ranged from 0.01 to 0.2, controlling gradient step size.
- **max_depth**: Tested depths from 3 to 10 to capture varying levels of data complexity.
- **min_child_weight**: Set between 1 and 7 to prevent splits on minor data points.
- **subsample** and **colsample_bytree**: Fraction of samples and features per tree, tested at 0.6, 0.8, and 1.0 for generalization.
- **Regularization**: gamma, reg_alpha, and reg_lambda imposed penalties on model complexity, enhancing robustness.

Early Stopping

Early stopping with early_stopping_rounds=10 prevented overfitting by halting training if no improvement was observed in 10 rounds.

Model Prediction and Target Adjustment

Post-training, predictions were adjusted by adding 1 to return to the original scale (1–5), ensuring consistency with the dataset's scoring format.

Results and Conclusion

The model demonstrated high classification accuracy due to effective data preparation, feature engineering, and hyperparameter tuning. In future work, exploring additional sentiment analysis techniques or incorporating temporal data (e.g., review date) may further enhance performance. The model might still exist overfitting problem, since my local score is above 65, and the score on Kaggle is around 57.

Sources:

1. https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article#:~:text=XGBoost%20is%20a%20robust%20machine,optimize%20their%20machine%2Dlearning%20models.
2. https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/