Predicting Star Ratings for Amazon Movie Reviews

1. Introduction

Problem Definition:

The objective of this project is to accurately predict star ratings for Amazon Movie Reviews based on a dataset containing various review-related attributes. The task requires developing a classification model that accurately predicts these ratings, drawing insights from the data and avoiding neural networks or any deep learning methods.

Summary of Approach:

My approach combines traditional machine learning techniques with strategic feature engineering to maximize predictive accuracy. I enriched the dataset by creating additional features - such as text and summary lengths, helpfulness ratios, and product-specific average scores - to better capture review sentiment and relevance.

For model selection, I chose a RandomForestClassifier, which aligns with the competition requirements while providing strong performance and interpretability. Moreover, text-based features were captured using TF-IDF, combined with numeric features after scaling, and optimized using SMOTEEN to address class imbalance. This multi-faceted approach allows the model to effectively analyze review sentiment and predict star ratings with accuracy and robustness.

2. Data Exploration and Preprocessing

Initial Analysis:

I began by examining the structure and distribution of key attributes in the train dataset, including “Score”, “HelpfulnessNumerator”, “HelpfulnessDenominator”, “Text”, and “Summary”. A bar plot of “Score” values revealed an imbalance class distribution: class **5.0** was overwhelmingly dominant, while class **1.0** and **2.0** are underrepresented, prompting me to consider class balancing techniques later. Additionally, analyzing text length for “Text” and “Summary” indicated that these features might be useful predictors, as longer reviews or summaries often correlated with more detailed feedback and potentially higher helpfulness.

Feature Engineering:

To enhance the model’s predictive power, I derived several features that encapsulate meaning aspects of the data:

- Helpfulness Ratio: Calculated as “HelpfulnessNumerator” divided by “HelpfulnessDenominator”, this ratio captures the extent to which a review was deemed helpful. For reviews with no denominator value, the ratio was set to zero.
- Text and Summary Lengths: I calculated “Text_Length” and “Summary_Length” to quantify the verbosity of reviews, as longer reviews might indicate more informative content that influences rating.
- Product Average Score: To account for product-specific tendencies, I precomputed the average score per product (“Product_Avg_Score”) using the training data, providing context on whether the review aligns with the general perception of that product.

Preprocess Steps:

Data cleaning included filling or dropping missing values and ensuring all text-based data was processed appropriately:

- Handling Missing Values: I addressed missing values in “Score” by removing rows with NaN values in this column, as it represents my target variable. For “Text” and “Summary”, missing values were replaced with empty strings before applying TF-IDF.
- Scaling and Encoding: I scaled numeric features (e.g., “Text_Length”, “Summary_Length”, “Helpfulness”) using **StandardScaler** to ensure consistent ranges across all input variables, facilitating optimal performance for the model.

3. Model Selection and Offline Evaluation

Chosen Models:

I experimented with several classification models to identify the model best suited to predict star ratings in this context. Initial trials included:

- **K-Nearest Neighbors (KNN):** Chosen for its simplicity and interpretability, KNN served as the baseline model. However, it demonstrated limitations in handling high-dimensional data, particularly with sparse TF-IDF features.
- **XGBoost Classifier:** This gradient-boosting model was considered due to its ability to capture complex relationships in data, but it required substantial training time with the dataset, even after tuning parameters.
- **Latent Semantic Analysis (LSA) with Unigram TF-IDF:** Initially I applied TF-IDF with unigram representation and then used **TruncatedSVD** for dimensionality reduction, setting n_components=100. While this approach effectively captured meaningful relationships in text data and worked well with numeric features, it significantly increased the runtime.
- **Random Forest with Bigram TF-IDF:** Lastly, I used RandomForestClassifier with n_estimators=150, max_depth=20, and min_samples_leaf=2. This time, I expanded the text feature representation by including bigrams in the TF-IDF transformation to capture two-word phrases and maximizing features to 1500. After tuning, this modification helped the model better grasp context within reviews, especially for sentiment-driven language, and improved overall accuracy.

Ultimately, I went with **RandomForestClassifier** with tuned hyperparameters above. This ensemble method balanced accuracy and interpretability and efficiently handled the variety of features engineered from the dataset within a short period (about 2 hours).

Offline Evaluation:

For model evaluation, I employed a train-test split (75% training, 25% testing) and performed cross-validation within the training set to assess consistency. Accuracy was the primary metric, and I also analyzed confusion matrices to examine performance across star ratings. Since the dataset had class imbalances, I tried to apply **SMOTEENN** (a combination of **SMOTE (Synthetic Minority Over-sampling Technique)** and **ENN** (**Edited Nearest Neighbors)**) to balance the classes in the training data, but didn’t have enough time to test the model’s sensitivity to underrepresented ratings.

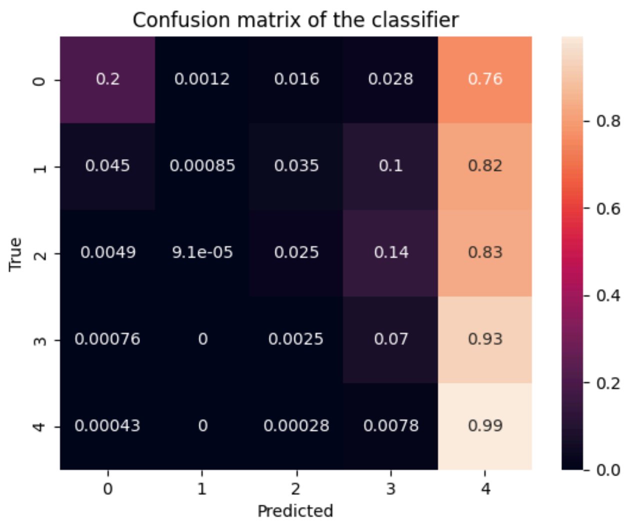
Feature Importance Analysis:

I analyzed feature importance based on the final RandomForest model’s insights:

- **Text_Length and Summary_Length** emerged as influential predictors, as expected, indicating that review verbosity correlates with sentiment or rating detail.
- **Product_Avg_Score** provided valuable context on product rating trends, helping the model align individual reviews with general product perceptions. Adding this feature significantly increased my model accuracy, even when tested with unigram TF-IDF features.
- **Bigram TF-IDF Features** enhanced the text-based feature set by capturing key phrases, significantly boosting predictive accuracy, especially when combined with numeric features.

These findings confirmed the value of my engineered features, and guided my feature selection and tuning for maximum model effectiveness.

4. Final Evaluation



The confusion matrix above provides insight into how well the model predicts each star rating. Each row represents the actual rating, while each column represents the predicted rating. High values along the diagonal indicate correct classifications, while off-diagonal values represent misclassifications.

- **Class Imbalance:** As seen, the model struggles with lower ratings (0 and 1), with substantial misclassifications into higher rating categories, particularly in the 4-star category. Applying **SMOTEENN** would help mitigate this to some extent by improving sensitivity to underrepresented classes, but challenges remain with accurately distinguishing between adjacent ratings.
- **High Accuracy for 4-Star and 3-Star Ratings:** The model shows strong accuracy in predicting 3-star and 4-star ratings, with the majority of instances in these classes being correctly classified. This could be due to the larger volume of examples in these classes, allowing the model to learn their patterns more effectively.
- **Performance on Middle Ratings:** The model has moderate accuracy for middle ratings (2-star), where there is noticeable misclassification into the neighboring 3-star and 4-star categories. This indicates that the model might struggle with reviews that have moderate sentiment, which could appear similar to both slightly positive and slightly negative reviews.