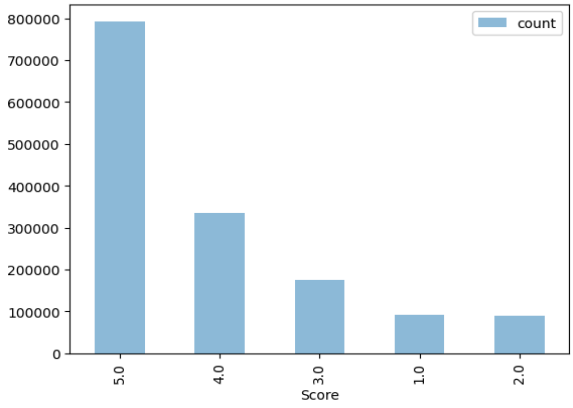
rdsam@bu.edu Rehan Samaratunga U42782539

# Amazon Movie Review Analysis Report

Kaggle Display Name: rehansamaratunga

1. **Preliminary Analysis**




The training dataset provided consisted of around 1,697,533 entries with each review consisting of a ProductId, UserId, HelpfulnessNuerator, HelpfulnessDenominator, Time, Summary, Text, and Score. By exploring the data, I could fully understand the structure and identify potential features that could be used for developing my predictive model and features that I would deem less useful. Right away I realized an imbalance with the significant proportions of the reviews being rated as 5 stars, while ratings from 2 to 4 were less common. This observation influenced my decision to adopt models capable of effectively handling imbalanced data.

2. **Data Preprocessing and Feature Engineering**

The first step involved preparing the data by importing and exploring the provided training and testing dataset. During my initial exploration, I noticed that understanding user behavior and product-specific factors could help improve the predictive power of the model. This led to a few specific actions:

**2.1 Sentiment Analysis Features**

- I used VADER SentimentIntensityAnalyzer from the **nltk** library to calculate a sentiment score for each review (both Text and Summary). This feature provided a quantitative measure of the sentiment which was useful in providing direct insight into the emotional polarity of the text, ranging from negative to positive values.

**2.2 User-Based Features**

- To better understand user behavior, I aggregated the scores for each user to derive features such as user_mean_score, user_std_score, and user_review_count. This allowed me to identify user-specific tendencies and biases, providing valuable context to each review. Users with consistently high or low ratings influenced the model differently compared to those with more diverse rating patterns.

**2.3 Product-Based Features**

- Similarly, product-based statistics included product_mean_score, product_std_score, and product_review_count. These features provided insight into how well each product was generally received by the customer base,

allowing the model to understand product popularity and the consistency of product reviews.

**2.4 Review Length and Helpfulness Features**

- Lastly, I also took into account features, like review length (word count) and helpfulness scores (based on HelpfulnessNuerator and HelpfulnessDenominator values). Longer reviews could express stronger sentiment, while helpfulness scores showed that the review carried more weight in determining the overall score.

**3. Model Selection and Hyperparameter Tuning**

Initially, I explored a few models, including K-Nearest Neighbor, Random Forest classification, and XGBoost classification. The starter code provided a baseline KNN model that achieved an accuracy of approximately 0.4, which was further improved to 0.57 by adjusting the value of k and removing irrelevant features like Time. Furthermore, a basic Random Forest classification produced an approximate prediction value of 0.6. However, I found that XGBoost was the most promising in terms of its ability to handle the features effectively and its overall flexibility in terms of hyperparameter tuning. Therefore my final model selected was XGBClassifier, which i optimized to improve performance.

**3.1 Grid Search for Hyperparameter Tuning**

- To achieve the best possible performance, I conducted hyperparameter tuning using GridSearchCV. This allowed the model to identify the most effective combination of multiple parameters for optimal accuracy. The cross-validation process also ensured that the model was not overfitting to specific folds of the training data.

**3.2 Feature Selection**

- To reduce noise and improve efficiency, I selected a subset of the features I believed to be most relevant. By narrowing the feature set, the model's performance improved, and this also reduced the training time.

**3.3 Handling Missing Values**

- Some users or products had only one review, resulting in missing values for the standard deviation calculations. Therefore these missing values were replaced with 0s which was a reasonable assumption since a single review does not provide variability.

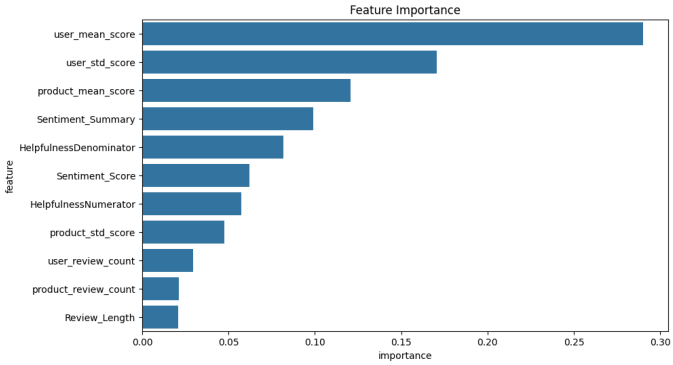
**3.4 Adjusting Score for XGBoost**

- Due to XGBoost preferring zero-based labels, I adjust the Score values to go from 0 to 4 instead of 1 to 5.

**4. Model Evaluation and Results**

I was able to evaluate my model using accuracy, confusion matrices, and classification reports. During my evaluation, the final XGBClassifier achieved the best accuracy among the tested models. Furthermore, to further understand the importance of the features, I analyzed feature importance as provided by the trained XGB model. The features with the highest importance were

the user and product average scores, followed by the sentiment score. Lastly, the test set resulting predictions were saved with the Score values adjusted back to the original scale. I believe the overall performance was satisfactory, and the model was able to capture general trends in user reviews, effectively predicting sentiments based on both textual and statistical data.




## 5. Assumptions and Observations

Throughout my process, several assumptions were made:

### 5.1 Sentiment Analysis as a Proxy for Score

- The sentiment of the text was assumed to be closely related to the overall review score. While sentiment analysis provides good insights, it might not always directly map to the given rating due to individual differences in how people express sentiment.

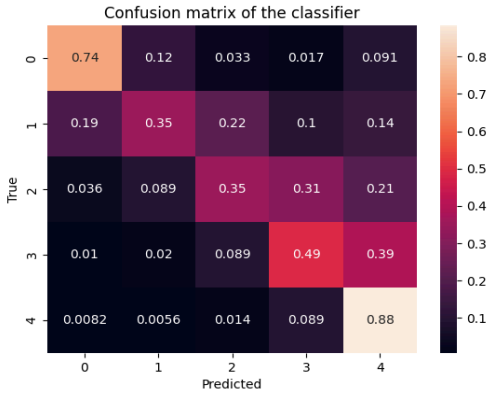
### 5.2 User and Product Behavior as Predictors

- I assumed that user and product historical behavior would have a strong influence on future ratings. This assumption held, as statistical features significantly contributed to the model's overall accuracy.

### 5.3 Feature Importance and Interpretation

- Analyzing feature importance highlighted that user and product statistics were more informative than text-based sentiment scores alone. This suggests that understanding broader user and product trends might be more beneficial in predicting ratings than analyzing individual reviews.

## 6. Conclusion




In conclusion, the final implementation utilized an XGBoost classifier with feature engineering, particularly focusing on user and product behavior, sentiment scores, and review-specific metrics. From the CM we can see that high values for classes 0 and 4 are well represented in the data while the lower performance on 1, 2, and 3 may suggest few samples or difficulty in distinguishing between these classes. One trick that improved my score was hyperparameter tuning and user/product feature aggregation as this increased my score from ~0.67 to ~0.69 locally. I believe that future improvements could include experimenting with more NLP techniques for text analysis and implementing class rebalancing (e.g. oversampling, undersampling) or regularization to improve the balance from class 0 to 4 and also improve the predictor accuracy.

**Citations**


- https://xgboost.readthedocs.io/en/stable/
- https://github.com/liannewriting/YouTube-videos-public/blob/main/xgboost-python-tutorial-example/xgboost_python.ipynb
- https://www.nltk.org/api/nltk.sentiment.vader.html
- https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://www.youtube.com/watch?v=XXHhrlL-FWc
- https://www.youtube.com/watch?v=GrJP9FLV3FE&t=2239s
- https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/
- Prior internship experience with creating NLP models
- ChatGPT provided helpful suggestions regarding engineering techniques and exploration methods to improve my model performance