CS506 Midterm Fall 2024 Kaggle Competition Report

Yanjia Kan
Boston University, MA, USA
kyjbu@bu.edu

Abstract

The objective of this competition was to predict the score associated with user reviews using the Amazon Movie Reviews dataset. In this project, I employed a random forest model, performed feature visualization and engineering, created new features, and vectorized textual features through natural language processing (NLP) techniques. Additionally, I experimented with various models and conducted parameter optimization. However, due to the large volume of the dataset and limitations in competition duration and runtime memory, my final results reached only about 60

1 Feature Selection and Engineering

To understand the dataset, the first step involved visualizing the distribution of its eight effective features. After basic deduplication and null removal, I handled each feature accordingly.

Helpfulness Initially, I used scatter plots to visualize the *HelpfulnessNumerator* and *HelpfulnessDenominator* features over time, as shown in the figure 1, where both features displayed almost identical distributions across time. I observed that these

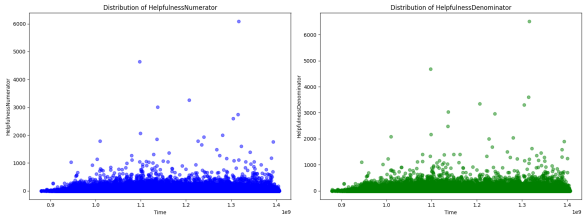

Figure 1: Feature Visualization for *HelpfulnessNumerator* and *HelpfulnessDenominator*

features primarily clustered below 1000. Thus, I re-visualized these two features below 1000 across different scores. The distribution remained consistent across the five rating levels. However, since both *HelpfulnessNumerator* and *HelpfulnessDenominator* represent the "helpfulness" of a review, I derived a new *helpfulness* feature by calculating the ratio of these two values.

ProductScore and UserScore I noted many duplicates within the *ProductId* and *UserId* features, indicating multiple reviews per movie and numerous reviews by individual users. Hence, I constructed the *ProductScore* and *UserScore* features to represent the average scores of the movie and the user, respectively.

Avg_Score_Per_Period Further, I observed that the scores given by the same user and the scores for the same movie vary over time. Thus, I segmented the *Time* feature based on review trends into five periods: *pre-2004*, *2004-2007*, *2008-2010*, *2011-2013*, and *post-2013*. I then calculated the average scores for each movie and user within these periods. *Product_Avg_Score_Per_Period* and *User_Avg_Score_Per_Period* , adding a temporal dimension to the features, which subsequent feature importance analysis confirmed as effective.

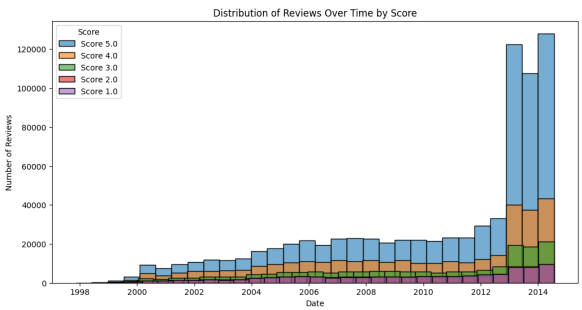

Figure 2: Scores by Top 5 Users

Expertise Inspired by the dataset authors' papers, I realized that users' rating preferences and expertise varied as show in figure 3. I attempted to quantify each user's expertise at a given data by aggregating data on review count, review timing, frequency of ratings, and review validity. However, this feature had minimal impact on model accuracy.

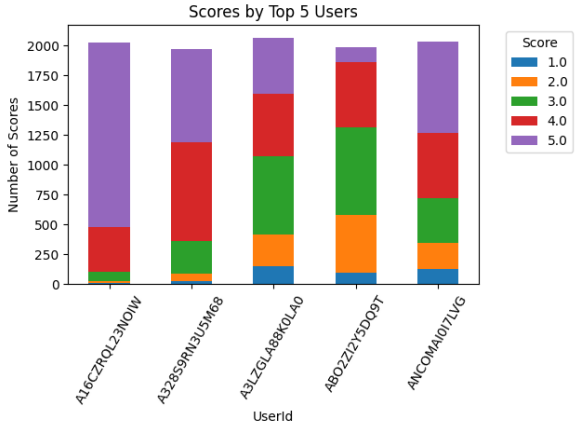

Figure 3: Reviews Over Time by Score

NLP For textual features in the dataset, I employed natural language processing to convert text into vectors for model prediction. I tested TF-IDF, Word2Vec, and bag-of-words models, ultimately selecting TF-IDF based on model accuracy. Additionally, I conducted sentiment analysis, testing both VADER and TextBlob methods, and chose TextBlob for its faster execution speed.

SMOTE and PCA Considering the imbalance caused by an overrepresentation of score 5 in the dataset as in figure 4, I used the SMOTE technique for balancing during testing with a portion of the data, which improved model accuracy. However, memory constraints during final testing prevented its application. I experimented with PCA for dimension reduction, but it significantly reduced the accuracy of classification, leading to its exclusion from the model.

This concludes the feature analysis for this project. Only the features *HelpfulnessNumerator*, *HelpfulnessDenominator*, *Helpfulness*, *ProductScore*, *UserScore*, *Time*, *tfidf*, *Summary_sentiment*, and *Text_sentiment* were

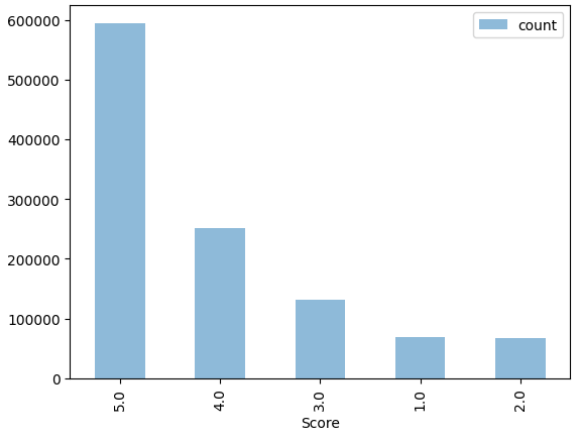

Figure 4: Labeling statistics

selected. The *Expertise* feature, having no significant impact, was excluded, and the features *Product_Avg_Score_Per_Period* and *User_Avg_Score_Per_Period* were dropped to avoid overfitting due to the lack of a viable solution within the time constraints.

2 Model Selection and Optimization

The processed features were normalized after being split in a 0.75:0.25 ratio for training and testing.

Model Selection The final model adopted for this competition was the Random Forest model. I tested various classification models including KNN, Logistic Regression, GBM, XGBoost, LightGBM, and Random Forest. Table 1 shows their respective accuracies.

Table 1: Model Accuracies

Model	Accuracy
KNN	0.4091
Logistic Regression	0.4541
GBM	0.4922
LightGBM	0.5175
XGBoost	0.5194
RandomForest	**0.5524**

Parameter Tuning I employed grid search to optimize three parameters for the Random Forest model: *max_features_options*, *max_depth_options* and *min_samples_split_options*.

Tabel 2 shows the optimal parameters were found to be *sqrt* for *max_features_options*,

Table 2: Random Forest Parameter Tuning Results

Parameter Pair	Accuracy
(sqrt, None, 2)	**0.6450**
(sqrt, None, 5)	0.6447
(sqrt, None, 10)	0.6448
(sqrt, 20, 2)	0.6286
(sqrt, 20, 5)	0.6286
(sqrt, 20, 10)	0.6276
(sqrt, 30, 2)	0.6411
(sqrt, 30, 5)	0.6415
(sqrt, 30, 10)	0.6419
(log2, None, 2)	0.6253
(log2, None, 5)	0.6229
(log2, None, 10)	0.6230
(log2, 30, 2)	0.6086
(log2, 30, 5)	0.6106
(log2, 30, 10)	0.6092

None for *max_depth_options*, and *2* for *min_samples_split_options*. The model achieved an accuracy of ***0.6450291531449039*** on the test set derived from the training data. The confusion matrix is shown in figure 5 and the feature importances shows in figure 6.

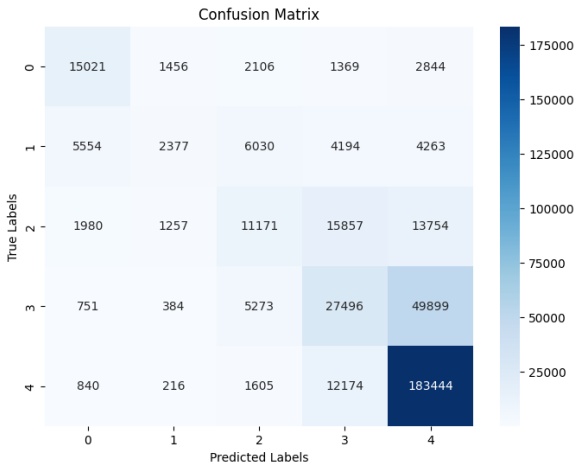

Figure 5: Reproduced Model Confusion Matrix

However, the model was overfitted, resulting in poor performance on the submitted test set. Despite attempts to adjust more parameters, overfitting issues were not resolved due to time constraints.

Innovative Attempt Constrained by memory limitations, which precluded the use of data balancing. I hypothesized that instead of handling *5*594838*feature_columns* data points simultaneously, I could first reclassify data with Scores of 1, 2, 3, and 4 as a non-5 label using a binary classification approach to separate score *5* data. This reduced the data points needing simultaneous processing to *2*594838*feature_columns*. Next, I classified Score 4 data and so forth until finally, binary classification was used to differentiate between scores *1* and *2*. The final accuracy of this innovative approach was only 0.58, due to my failure to address the issue of incorrectly classified data being unable to be reclassified into the correct category. Time constraints prevented further attempts to use weights to solve the misclassification issue.

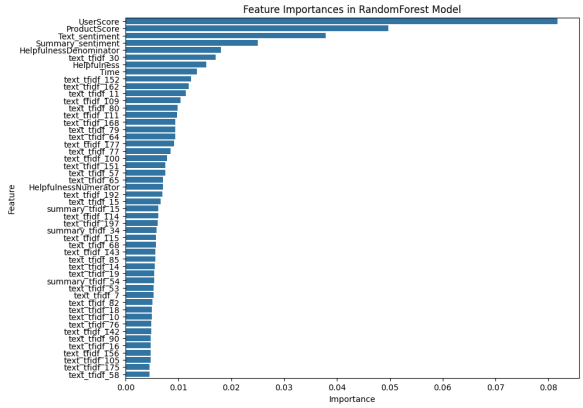

Figure 6: Reproduced Model Feature Importances

3 Conclusion

In this project, I engaged in extensive feature engineering, creating several features of high importance. I also tested multiple classification models and conducted parameter optimization, achieving the final classification results. However, this result was not optimal. Constraints imposed by memory capacity and competition time limited the use of data balancing and resolution of model overfitting and further modifications to the innovative model. Nonetheless, this project deepened my understanding of the importance of feature engineering. Different combinations of features can construct more significant characteristics, providing valuable experience and direction for my subsequent data analysis endeavors.