Yizhou Wang CS506 Midterm Report

Introduction

In this project, we aimed to implement a classification model to predict review scores. The approach involved preprocessing the data, selecting features, and experimenting with various classification algorithms. This report highlights the chosen and filtering of features after data analysis, and chosen algorithms, optimizations made to enhance model performance.

Feature Selection and Dimensionality Reduction

- Observations and Patterns Noticed

Numerical: I found that data relates to helpfulness and the helpfulness gap has some correlation with the score, as well as the different users tend to give different scores.

Textual: In the Text section, words like "excellent," "poor," "average" were highly indicative of certain scores, contributing to the model's interpretability. Also, a lot of comments contain emotional words.

- Initial Feature Creation

After analyzing the dataset, I found several patterns in numerous columns. According to the patterns, I added the following features into the feature set for further filtering.

- 'HelpfulnessNumerator': Representing the number of users that found the review helpful
- HelpfulnessDenominator': Representing the number of users that react on the review.
- Helpfulness': Representing the ratio of helpfulness of a review.
- 'CombinedHelpfulness': Use two values a and b to combine the helpfulness gap (Denominator - Numerator) and the ratio into one representation.

- 'EmotionalWordCount': Use nrc_emotion_lexicon to generate frequently used words to express emotions. The feature represents the count of emotional words in each star scoring.

  Source: https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

- 'user_combined_rating': Representing the users' rating habits. Calculated by 'weighted_user_average' + 'rating_std_dev' + 'user_rating_tendency' with some weights.
- 'SentimentScore': Representing the sentiment score of the text section of each comment. Using Vader to compute the sentiment score.

  Source: https://github.com/cjhutto/vaderSentiment

- Training Feature Selection

Through a series of testing, both testing one by one and pairwise in choosing different features to train on the model, I selected 'HelpfulnessNumerator', 'HelpfulnessDenominator', 'Helpfulness', 'user_combined_rating', 'EmotionalWordCount', 'SentimentScore' for the final feature. All of those features can increase the accuracy of the model. In which SentimentScore has the biggest boost on accuracy.

Algorithm Selection

I first use a cross validation to run different models, including 'Logistic Regression', 'Support Vector Machine', 'Naive Bayes', 'Random Forest', 'Gradient Boosting', 'Decision Tree', 'K-Nearest Neighbors (KNN)', on 5% of the dataset. However, knowing that the accuracy of the partial dataset is not the same as running on the full dataset, it can still be a relative source to refer to when choosing models. According to the result, I chose KNN, Logistic Regression and Gradient Boosting for the model according to the accuracy predicted.
After running the three models on the selected feature of the complete training set, I found Gradient Boosting has the highest test set accuracy.

Hyperparameter Tuning

The model's performance was further optimized by tuning hyperparameters using cross validation. I use this method to compare different hyperparameters, and result in the current values.

Correlation Matrix / Confusion Matrix Analysis

An analysis of the confusion matrix was performed to identify specific score levels that were commonly misclassified. This analysis led to adjustments in feature selection. I also used a correlation matrix to find the most influential features using importance scores.

Overfitting in Initial Trials

Early iterations showed overfitting due to the extensive feature set. Either the test set accuracy lowered after adding too many features, or the accuracy score on Koggle is lower than the one resulting from the local test set. I found that it is also possible that certain features contain too many unique values, such as in my case, using product id as a feature. I also use the correlation matrix after resulting in an unusually high accuracy, to find the features that are a possible cause of overfitting.

After reducing and regularization adjustments helped to alleviate this issue.