Amazon Movies Reviews Star Rating Classifier

Analyzing data

After loading data, I first look through it and see which columns have the most correlation with star ratings. I ended up with "Summary" and "Text," which are useful for sentiment analysis in predicting star ratings.

Below I will show why I disregarded the other columns

The "HelpfulnessNumerator" and "HelpfulnessDenominator" show how helpful the review is to the users, which only tells us whether the movie review is on point or not. It doesn't tell us whether the movie is good or bad.

For the "UserId," with enough data, it can show us certain user's rating pattern. If we see enough reviews and ratings of the same user, then it can show us whether this user tends to rate high or low. However, we don't have enough of those data.

For the "ProductId," with enough ratings of the same product, we can tell whether this product is overall good or bad. However, we don't have enough data for each product.

The "Time" and "Id" don't correlate with the star ratings.

Problems within data set

I realized that the data set is really imbalanced because most reviews have high ratings with mostly 5 star ratings. At first, I tried to use SMOTE to oversample the non-majority class data, but it did not improve the accuracy much and it increases the data set by a lot and takes forever to train. Instead of trying to oversample the dataset, I keep the imbalance dataset as it is because it also shows us how most people tend to submit high rating reviews.

In offline evaluation, I only take around 30000 data entries to first find out the best data cleaning process, vectorization techniques, text classification models, and the best hyper-parameters, before I run it through the whole dataset.

Data Preprocessing

I first replace all the Nan values in “Summary” and “Text” with the empty string and then I combine the Summary and Text together into one column. I then remove all the rows with empty strings.

Text cleaning process is shown below:

1. Tokenized the text
2. Lower casing
3. Remove stop words
4. Remove punctuations
5. Strip spaces
6. Stemming

After text cleaning, I first use train_test_split to split my data into 80% training and 20% testing. I then vectorized the train and test data with TfidfVectorizer with ngram range of 1-3, min_df = 3, max_df = 0.9, which reduced the training vector sizes, and I also and apply sublinear tf scaling. Now the data is ready to be trained by classification models. Different parameters are tried out for the vectorizer and these are the best parameters I find.

Model training

I tried out different models such as Linear SVC, boosting, Logistic Regression, KNN, Random Forest Classifier, and etc, and I find that Linear SVC and Logistic Regression perform the best. However, Logistic Regression tends to not converge and I’ll need to increase the max iteration to overcome that, which makes it run forever. With Linear SVC, I set the class_weight as balanced to accommodate with the imbalanced dataset. After training with the training data set, I use the model to predict the test dataset and use mean square error to calculate my performance.