SEPTMEBER 2019

Predicting Beer Ratings from Text Reviews

Submitted by: Soham Desai
Springboard Data Science Career Track

Introduction

According to the Alcohol and Tobacco Tax and Trade Bureau (TTB), there were over 10,000 breweries within the United States alone at the end of 2017. That's a lot of people making beer! With an increase in brands and styles of beer it can be overwhelming to keep up with latest trends and patterns. Therefore, consumers often rely on other user reviews and ratings to determine whether they may try a new product. With this project, I aimed to create a rating predictor based on user reviews that can help both brewmasters and beer connoisseurs choose the next beer they might want to endulge upon.

AUDIENCE

Any merchant, business or brewmaster would benefit from using a rating predictor such as this one to determine what beers to create or what beers to keep in their inventory.

DATA SOURCE

This data was obtained from Julian Mcauley, a UCSD Computer Science professor. It is a collection of beer reviews scraped off the website BeerAdvocate.com over a time period from January 1998 to November 2011. The overall dataset contains over 1.5 million reviews, but for this project a subset of 99,999 reviews were used.

Data Wrangling

When it comes to working with data, the first step needed to be taken is examining and tailoring the data to fit your questions and needs. Below are some of the issues I dealt with when working with this data set.

1. **Mismatched Rating Scales:**
   In total there were four scales: Aroma, Palate, Taste, Appearance and Overall. Aroma and Taste were on a scale of 10, Palate and Appearance were on a scale of 5 and Overall was unknown. To remedy this, I adjusted each rating to a 5-point scale. Then I created a new "Overall" rating by averaging the four sensory ratings.

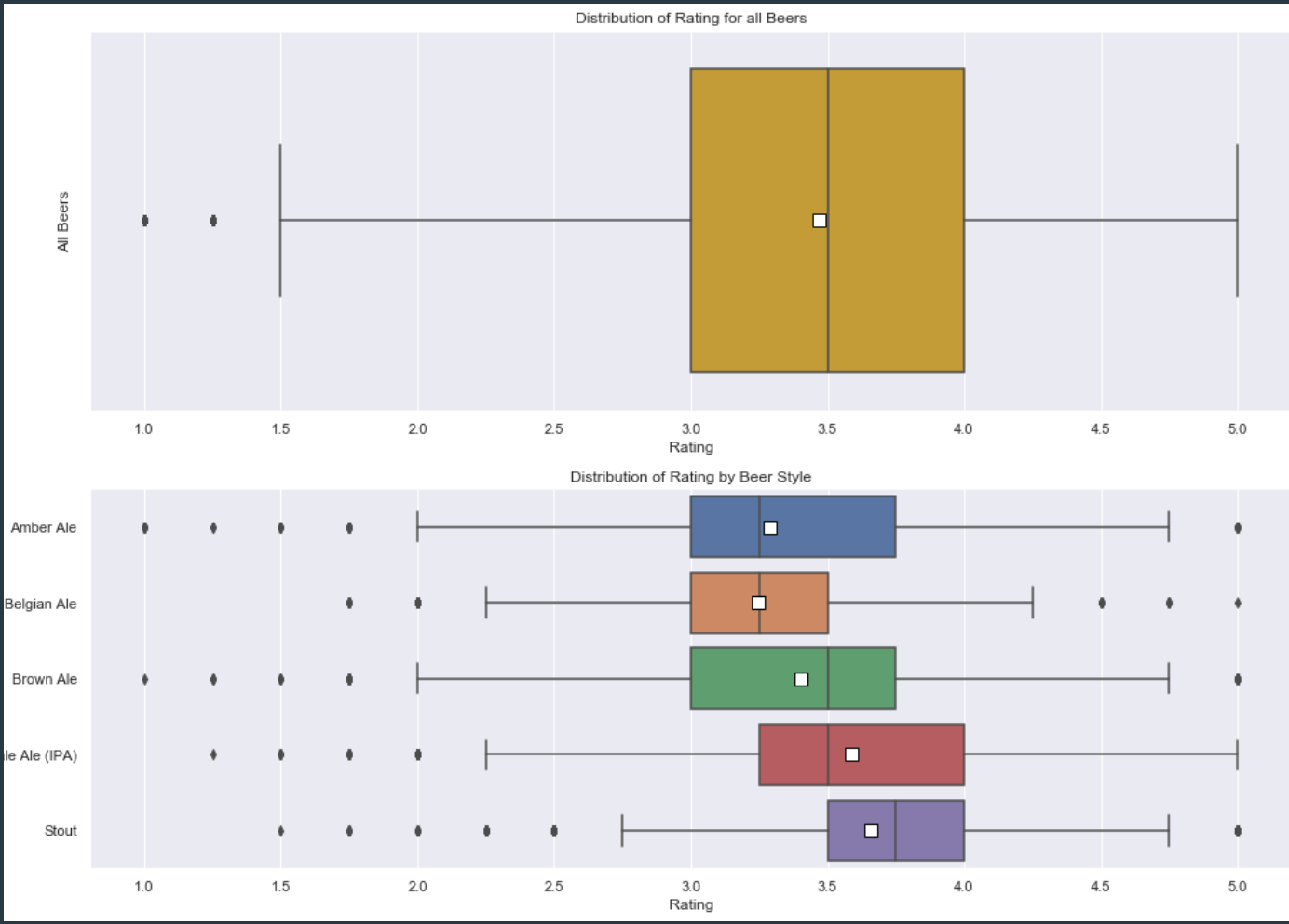
2. **Non-English Text Reviews**
   Some of the text reviews were not in English and this is an issue when using NLP because it can throw the model off by the non-English words. Using lang detect, I searched through the data and dropped any rows of data that were not identified as English.

3. **Text Preprocessing for User Reviews**
   Text reviews have a bunch of miscellaneous information such as spelling errors, punctuation, contractions, capitalization, whitespace, etc. that is not needed. In addition, words can be in different verb tenses. For example the word play can be written as played, playing, plays. Since I only care about the base word, I used a process known as lemmatization to derive the base of each word. Lastly, there are common words that can be found in many reviews that do not help in creating insight within the model known as stopwords. To remove all of these from the text I created a pipeline to clean the text and prep it for analysis and machine learning.

Exploratory Data Analysis

RATINGS

In total there were 5 rating scales. I chose to focus on the "Overall" rating because it was an average of the rest. Along with examining the entire data set I selected five beers by length of data and style of beer to examine. You can see in the figure below that majority of reviews fell between 3 and 4 star ratings on the 5-point scale. The black line represents the median and the white square represents the mean. Moving forward I decided to split the Overall rating into a binary grouping. Anything lower than a 3.5 star rating was considered to be "low" and the rest were considered to be "high".

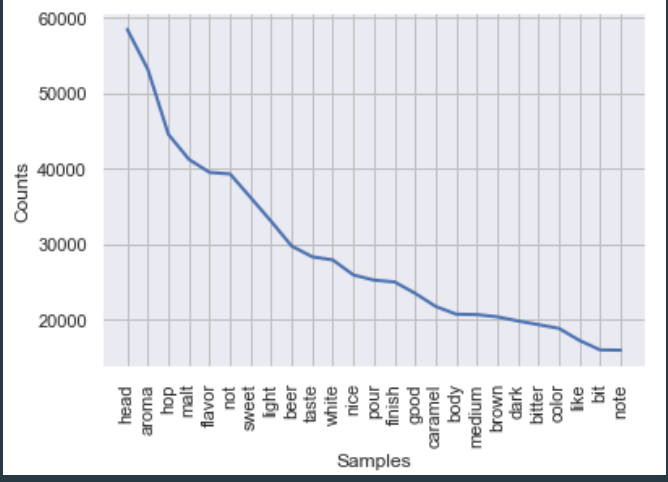


EDA (cont.)

TEXT REVIEWS

As mentioned previously, the main feature for the model was the text reviews. I decided to explore these reviews using word frequency plots and the predicted probabilities a given word or phrase would be considered a "high" or "low" overall rating.

First, the word frequency plot showed sensory words to describe beers such as "sweet", "light" and "caramel". There are also words in relation to beer such as "hops" and "head" (in reference to the foam at the top of a beer). A surprising word to see was "not" because it can change the polarity of a sentence. For example the sentence, "This is good", can change from positive to negative by adding the word not, "This is not good". To account for this I went back to edit the stopwords to ignore the words "no" and "not". You can see the 25 most frequent words in the plot below.



I also looked at the predicted probabilities of 1-2 word phrases. When examining individual words, the probabilities were not that strong. There were words like "boring", "bland", "excellent" and "wonderful".

Examining both individual words and two word phrases provided stronger probabilities. There were two word phrases such as "great brew" and "light watery" but also one word phrases like "meh". This discovery led me to explore ngrams when creating my machine learning model.

Machine Learning

VECTORIZERS

The first step in creating a machine learning model with text data involves vectorizers. It is hard for a machine to understand and perceive words like we do as humans. Vectorizers encode the text data in a more compatible context. I examined two different ones: CountVectorizer and TfidfVectorizer. CountVectorizer uses a bag of words approach, counting the occurrence of each token while the TfidfVectorizer creates weighted term frequencies based on how frequent a token appears. For example if the word "IPA" appears in almost all reviews, the weight put on this word is decreased because it is not as beneficial for predictions because it appears frequently. I used the ROC-AUC score as the metric to compare the models and used a Multinomial Naive Bayes model as the classifier. I determined that the TfidfVectorizer hypertuned with GridSearch performed the best. The results can be seen in the table below.

VECTORIZER	ROC-AUC	BEST PARAMETERS
CountVectorizer	0.853	min_df=1, alpha=1, fit_prior=True
TfidfVectorizer	0.848	min_df=1, alpha=1, fit_prior=True
CountVec w/ GridSearch	0.853	min_df=.0001, alpha=1, fit_prior=True
TfidfVec w/ GridSearch	0.857	min_df=.0001, alpha=1, fit_prior=True

Machine Learning (cont.)

CLASSIFIERS

After determining the best vectorizer, I went on to examining different classifiers. Again, I used the ROC-AUC score as the metric to test performance by each model. I fit and tuned four classifiers: Logistic Regression, Support Vector Machines, Multinomial Naive Bayes, and Random Forest Trees. Each classifier used an ngram_range of (1,2) and a min_df of 0.0001. For Random Forest Trees, and Naive Bayes I used GridSearchCV for cross-validation and hyper paramter tuning. For Logistic Regression I used a classifier that had cross validation already built into it (LogisticRegressionCV). Lastly, I did not continue with the SVM classifier because of the amount of time it took to run the model.

Logistic Regression had the highest scoring model at 0.873. The score and parameters of each model are in the table below.

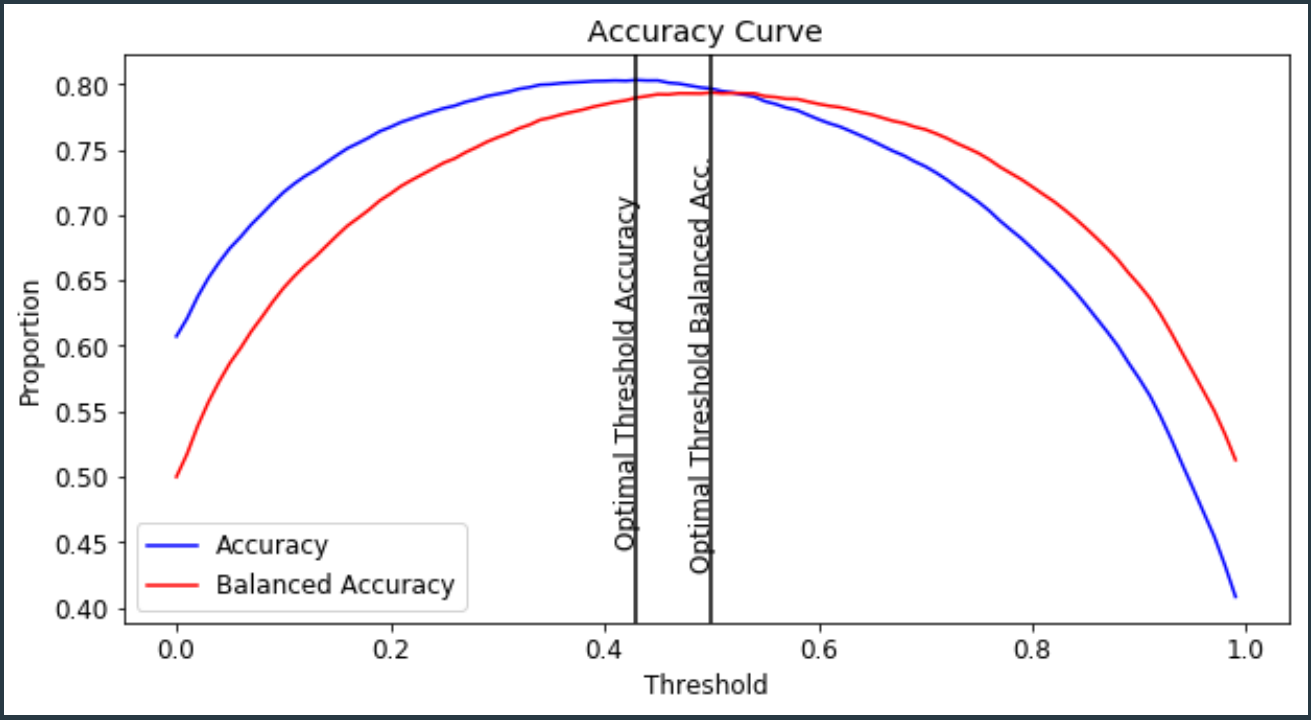
CLASSIFIER	ROC-AUC	BEST PARAMETERS
MultinomialNB	0.859	alpha=1, fit_prior=True
LogisticRegresionCV	0.873	Cs=10, max_iter=1000
RandomForestClassifier	0.851	max_features=500, min_samples_leaf=5
SVC	0.868	C=1, kernal='linear', degree=3

Improving Classifications

The default threshold for each model was at 0.5, meaning any predicted probability less than 0.5, is predicted"low" and the rest are predicted to be "high". This may not be the optimal model depending on the business problem that is trying to be solved. To demonstrate this I adjusted the threshold for two different business cases.

BUSINESS CASE 1: ACCURACY

Let's say in this situation you want to examine new comments from newer reviews, social media or other websites that incorporate beer reviews. You'd want a model that is as accurate as possible. The metrics I used as a measure of performance were "accuracy" and "balanced-accuracy". What I discovered was that adjusting the threshold to 0.43 increases the accuracy of the model by 1%. This can be seen in the graph to the right.

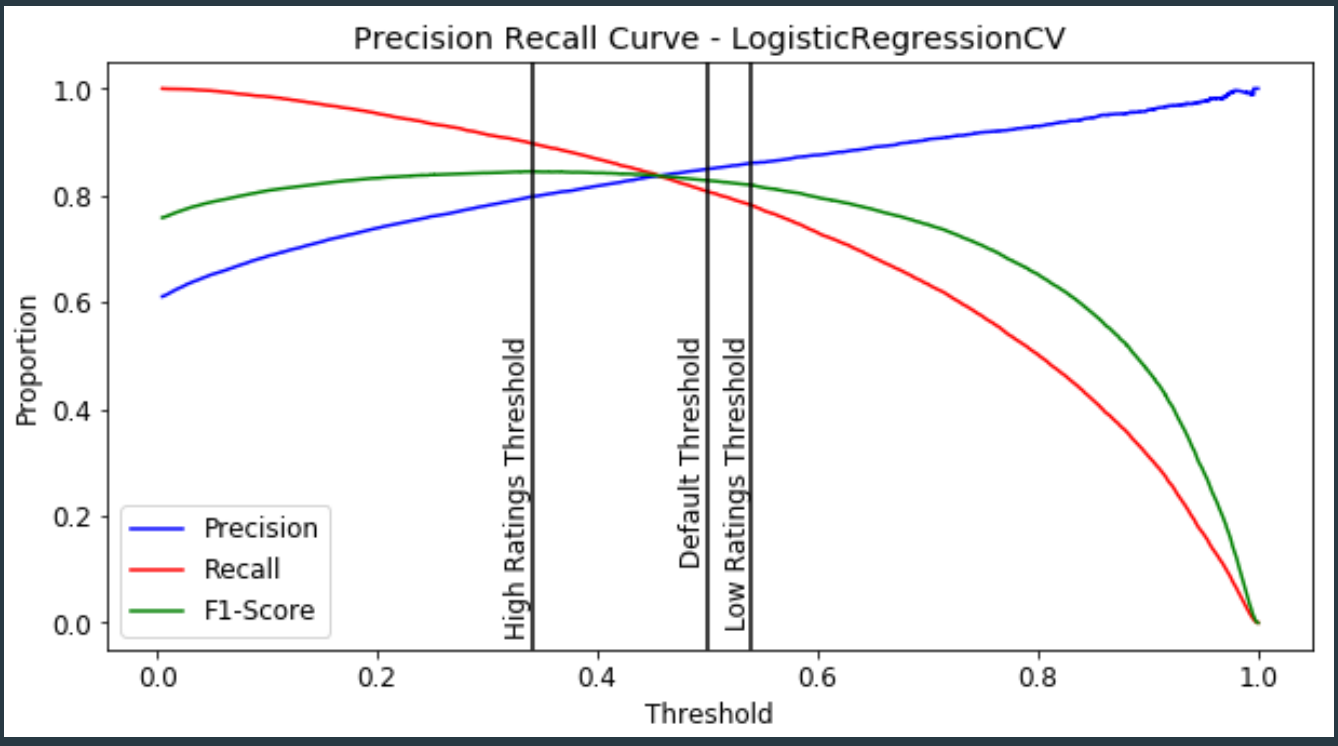


Classifications (cont.)

BUSINESS CASE 2: FOCUS ON "LOW" RATINGS

Now let's say you're looking into expanding your customer service team. To do so you need to know what negative remarks are being made by users in order to remedy them appropriately. In this situation you'd want to focus on increasing the correct number of predicted "low" reviews. To measure this I used the metric F1-Score. F1 combines precision and recall, meaning the higher the F1-Score, the better the model will perform.

Since the focus is on the "low" ratings or the "0" values in the prediction it is important to note that when calculate the f1_score you have to make the parameter pos_label=0. In the opposite, which I will mention momentarily, you may leave it at the default value pos_label=1. After examining various thresholds, the optimal threshold for "low" ratings is at 0.54. It outputs an F1-Score of 0.75. In comparison looking at the opposite, the optimal threshold for "high" ratings is 0.34 with an F1-Score of .82. This can be seen in the plot below.



Further Improvements

-> Increase the data set from 99,999 subset to the entire 1.4 million reviews. Also I could use more recent beer reviews from any current websites.

-> Spend more time focusing on individual styles of beers. There are over 100. different styles.

-> Explore the other four rating scales: Appearance, Aroma, Palate, Taste.

-> Explore word connections more with advanced NLP techniques such as Word2Vec and LDA

-> Model using the 5-point scale rather than a binary "low" and "high"

For the full report and files visit my github: sdesaidata