Amazon Alexa Reviews Classification

Capstone Project 2

Catherine Somers

Problem Statement

The purpose of this project is to create and train a model of Amazon Alexa reviews to determine how Amazon Alexa products can be improved upon. As a main focus, sentiment analysis will be used on the data. In doing so, determining and predicting the significance to the overall reception of all Amazon Alexa related products. As a consumer, we can to some extent grasp an insight on positive and negative reflecting reviews. From a business perspective this is useful to a product marketing or product team when trying to position a product in future version releases for the product line. Based on the insights inquired from reviews, a more informed decision can be made for the newer version of an existing product. A different but another use case would be to detect which features within the data determine what is categorized as positive or negative feedback. Product managers usually use reviews as a way to gauge how they could further improve a future product release down the line. With each new product iteration, an existing functionality is almost always made better to improve the product user's overall experience using the device. In addition, while this report may focus directly on Amazon Alexa reviews, the techniques used here could also be applied to any dataset composed of reviews with a rating system and written reviews.

Data Acquisition and Wrangling

The first step in preparing an analysis of Amazon Alexa reviews was to search for a suitable dataset. From a business standpoint, the overall sentiment of a product in a product line is important. Given the nature of how voice assistants have shaped the scope of how we live our lives, I wanted to explore the ways how these products have improved the quality of life. What made me consider Alexa enabled devices specifically was more to do with the interest in seeing what people had to say about Amazon Alexa over other competing voice assistants that exist in the market.

I found data for Amazon Alexa products that provide input data, ratings, review date, product variant, and feedback for a number of various Alexa products. I looked on Kaggle as it is one of the largest platforms to find public data and an overall great place to find data to solve a problem using classification. The dataset I chose for my analysis is not completely up to date since the last update was around 2 years ago. Even though it is not up to date, there's still the possibility of having a snapshot of customer impressions within the timeframe of the reviews in the data.

To start working with the data I found from Kaggle, I needed to download the data file. This data file was a .tsv file containing 3151 values for ratings and reviews on Amazon Alexa products from the past two years. Once the file was downloaded, I loaded and read it into a notebook. The columns in the data file were 'rating', 'date', 'variation',

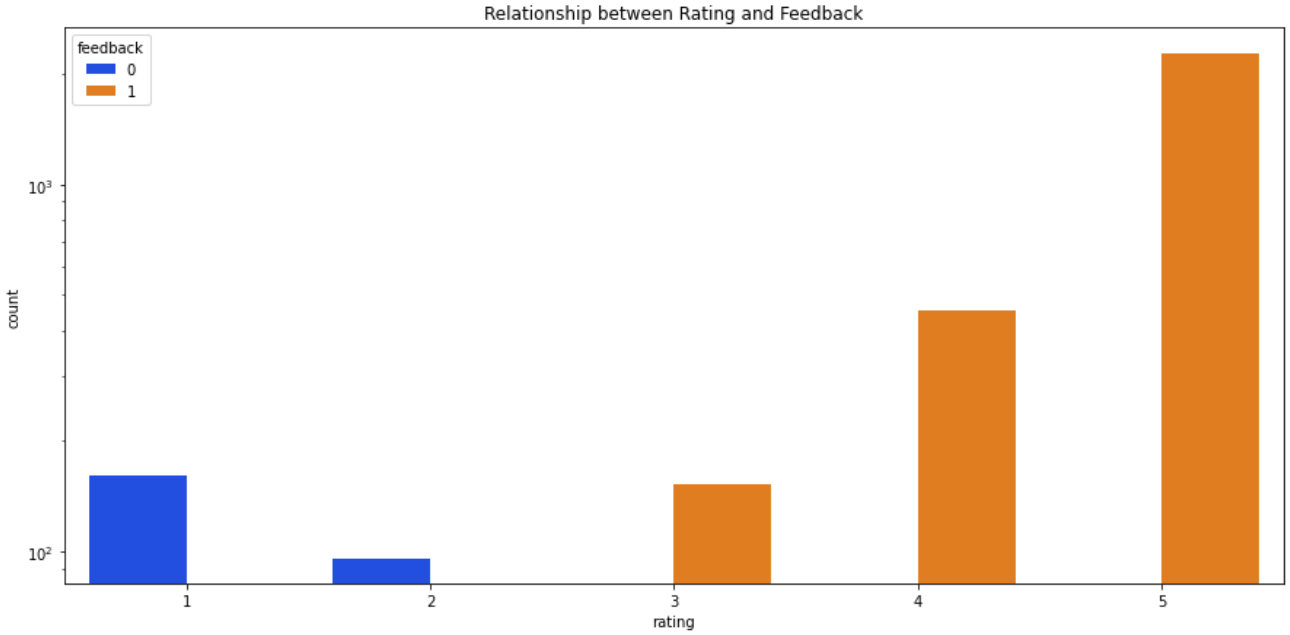
‘verified_reviews’, and ‘feedback’. After converting the .tsv file into a pandas dataframe, I did some initial examining to see if there were any null values. From this initial observation there were not really any null values in the data, so I could be certain that the dataset would be fairly clean. The average rating of reviews across every variation was around 4.46 out of 5. So for the most part, most device users have generally been pretty happy with the product.

To perform some feature engineering later in the model, year, month, day of the week, and review length were extracted into separate columns. Estimating review length is an important feature for text classification in Natural Language Processing (NLP). As a result from this, new columns were added to the data.

Data Exploration and Storytelling

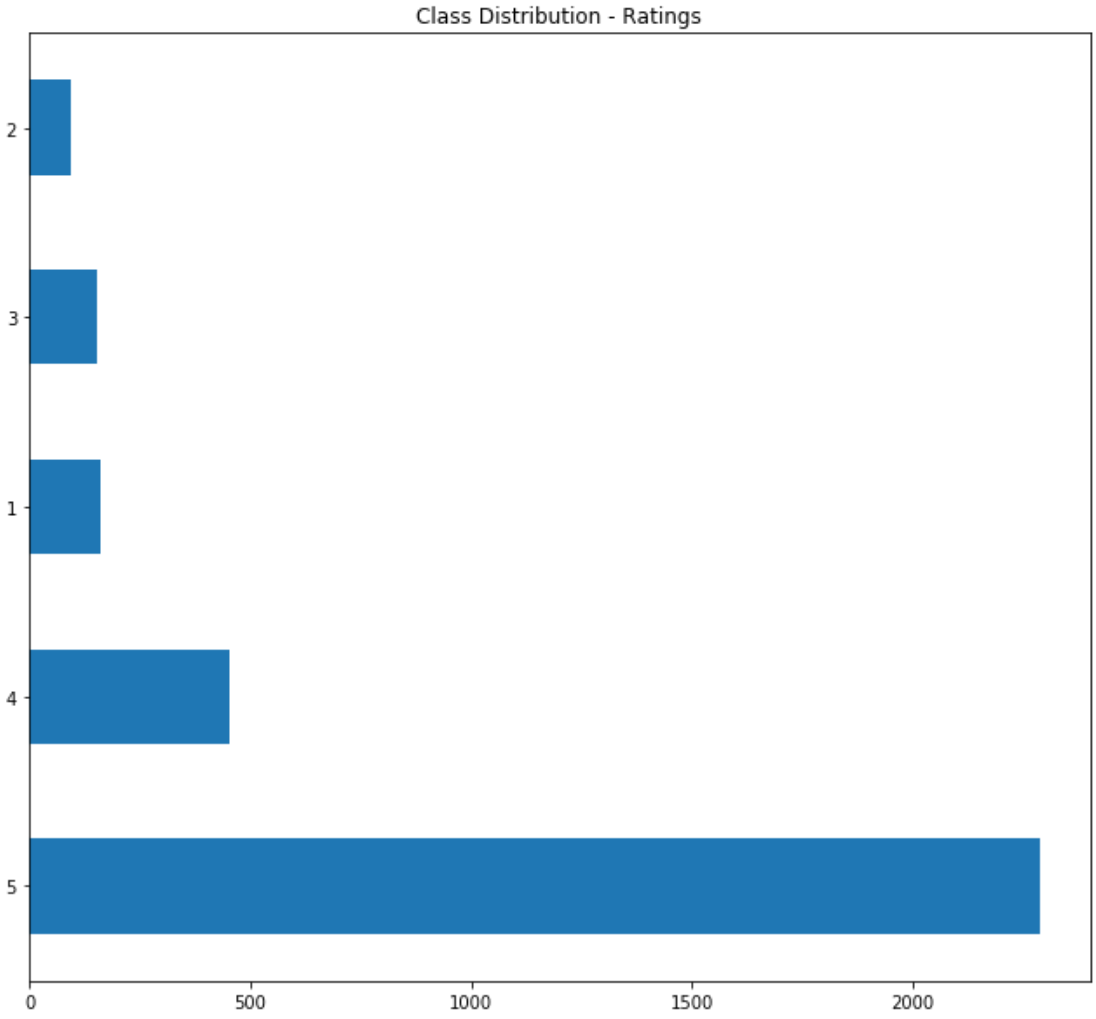
Upon completing the precious step, it was next time to explore the dataset through visualizations. I hoped to find some patterns among product variation and overall feedback. I wanted to also find out more about the relationship between rating and overall feedback.

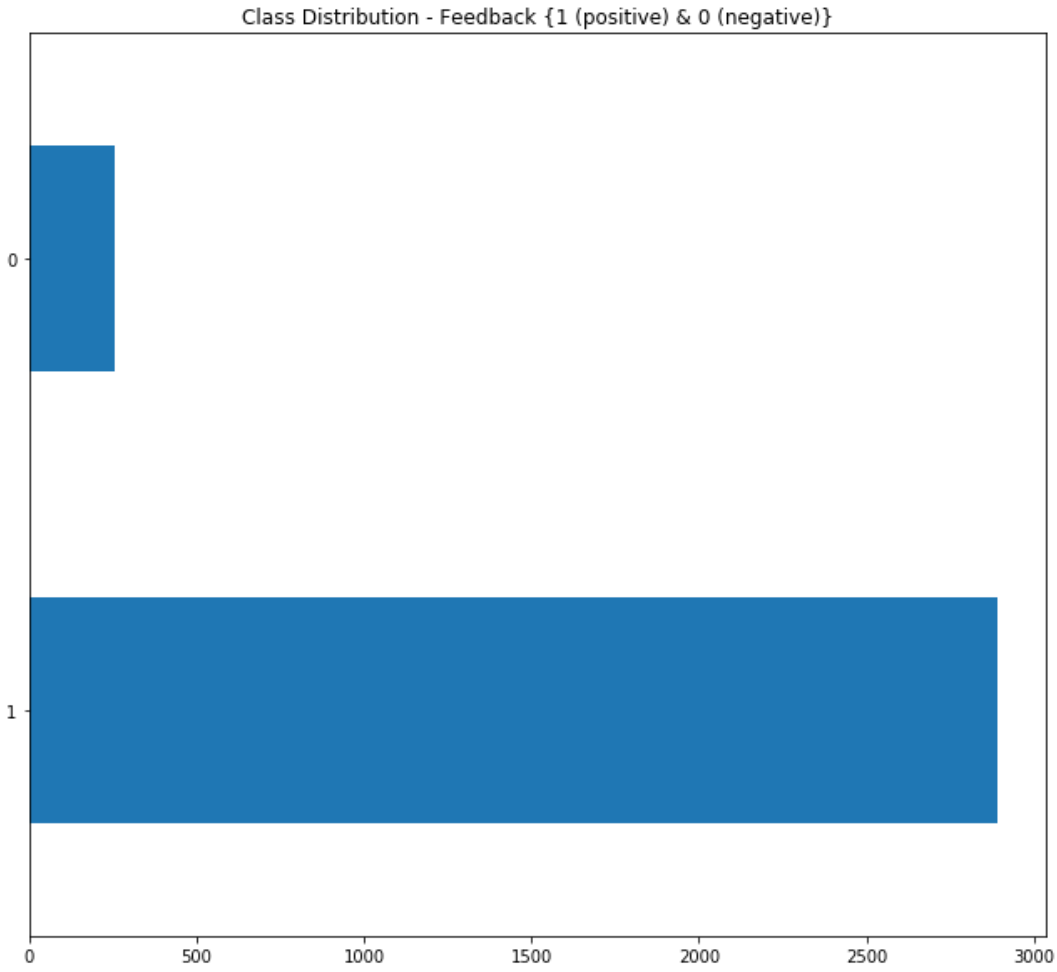
I began looking into the relationship between rating and feedback by plotting both in a bar graph. In order for feedback to be considered positive, its ratings need to be greater than 3.



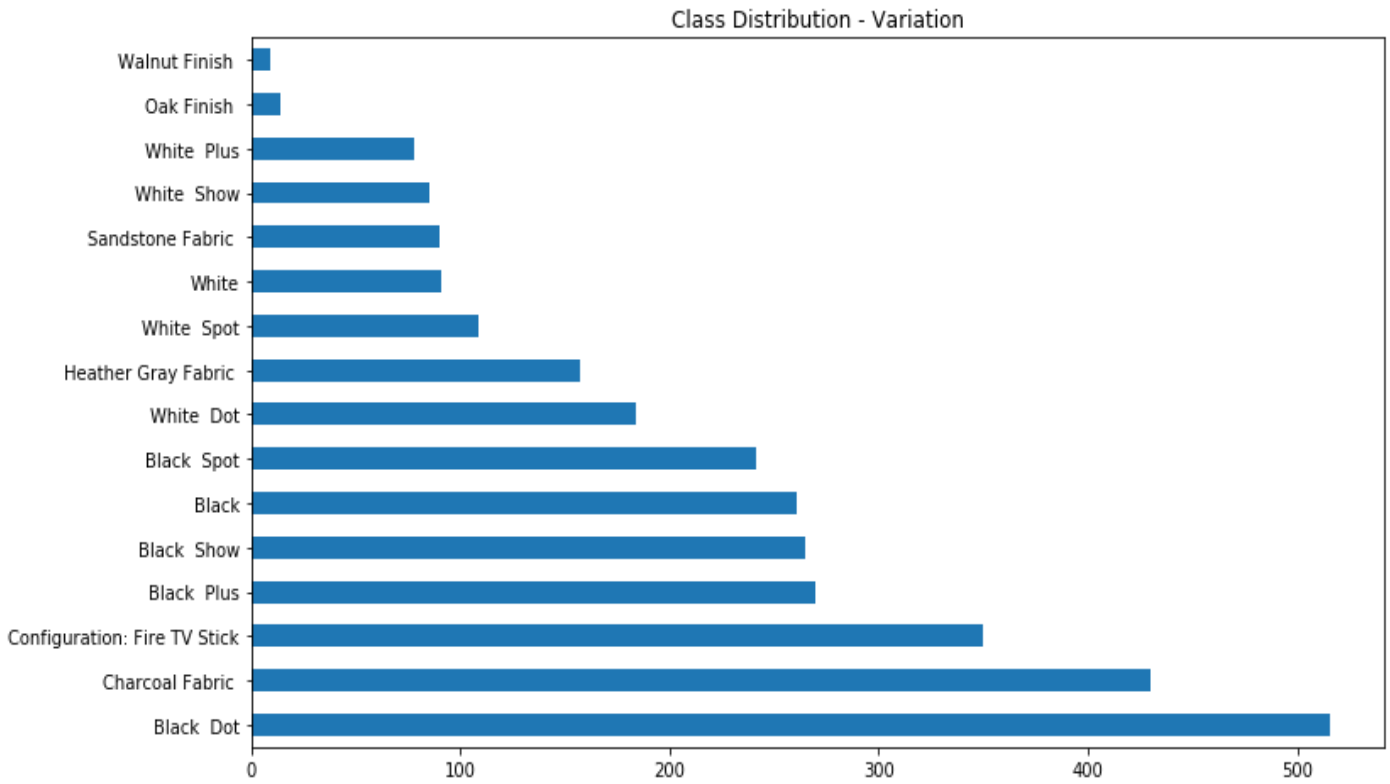
This gives more of a general overview of the relationship between rating and feedback. As we see here, the value range for negative feedback is 2 and below and value range for positive feedback is 3 and above. I felt this did not give me as much insight as I

would like to see so I wanted to look additionally at feedback and ratings in their own respective bar plot to see the distribution of the feedback and ratings.



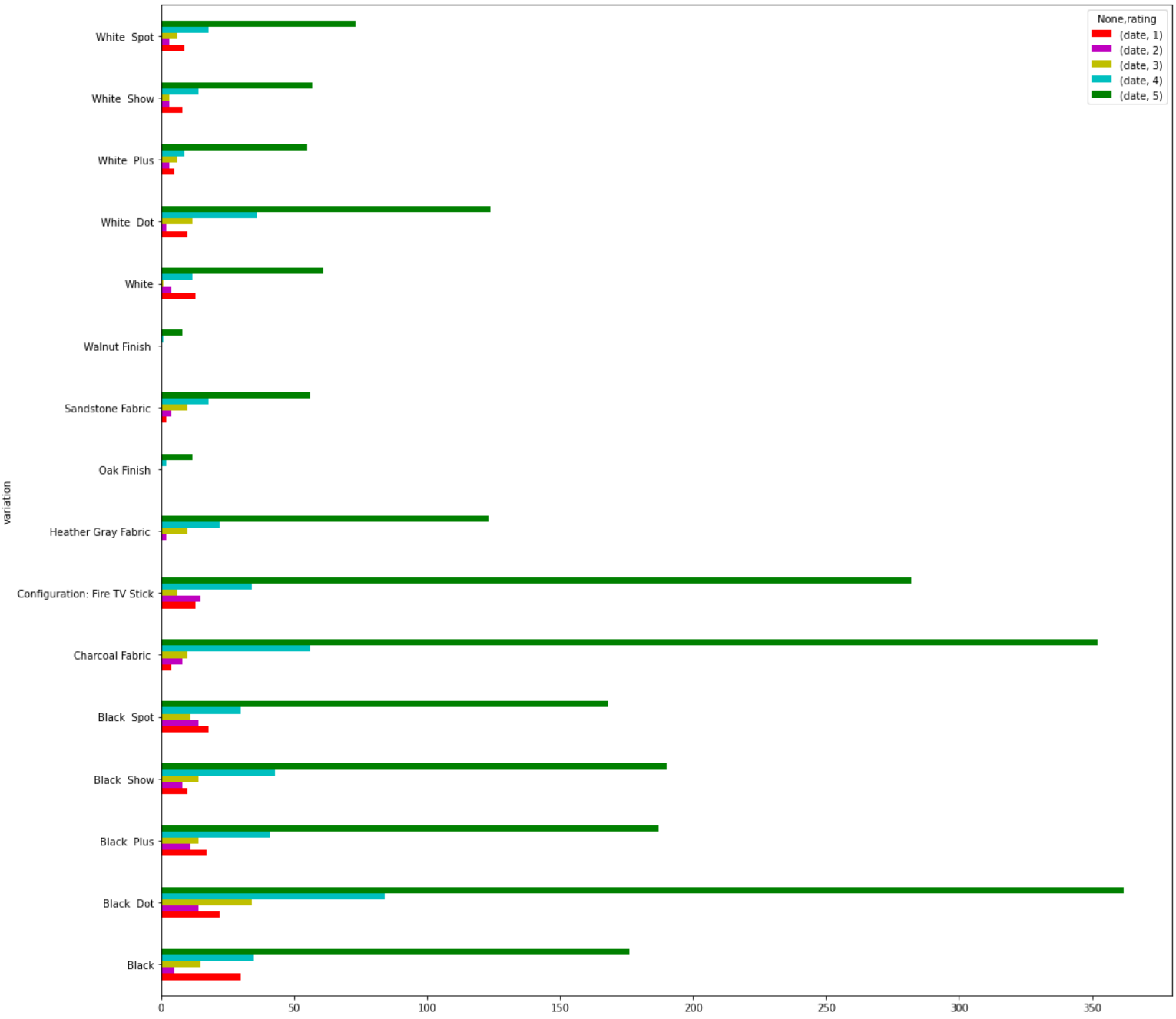


In both cases, we see the distribution among feedback and ratings is highly skewed on the positive side. For the most part, products have been well received by the customers. What may help in the future to review models is stratifying the data to help avoid having a class imbalance. This is something to keep in mind when working with imbalanced data in future cases.



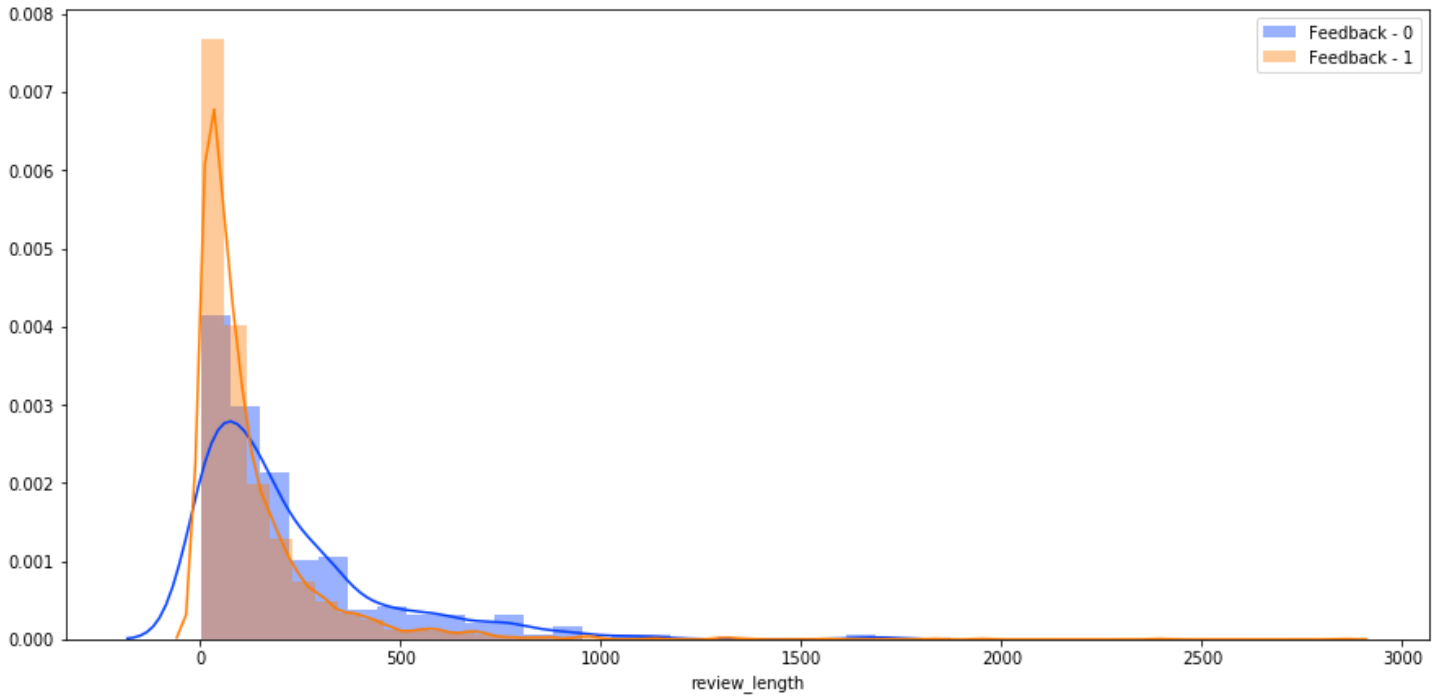
There are distinct bins here for each product type in the Alexa enabled line of devices show a pattern for the model preference. The Black Dot model is the most popular one out of all the different variations.

From the plot below, the Black Dot is the Amazon Alexa product with the most ratings of 5. This shows that more people may want to spend less than the normal version to experience using Alexa as a voice assistant in their home as the central control hub for a person's entertainment system, kitchen, or a children's room.

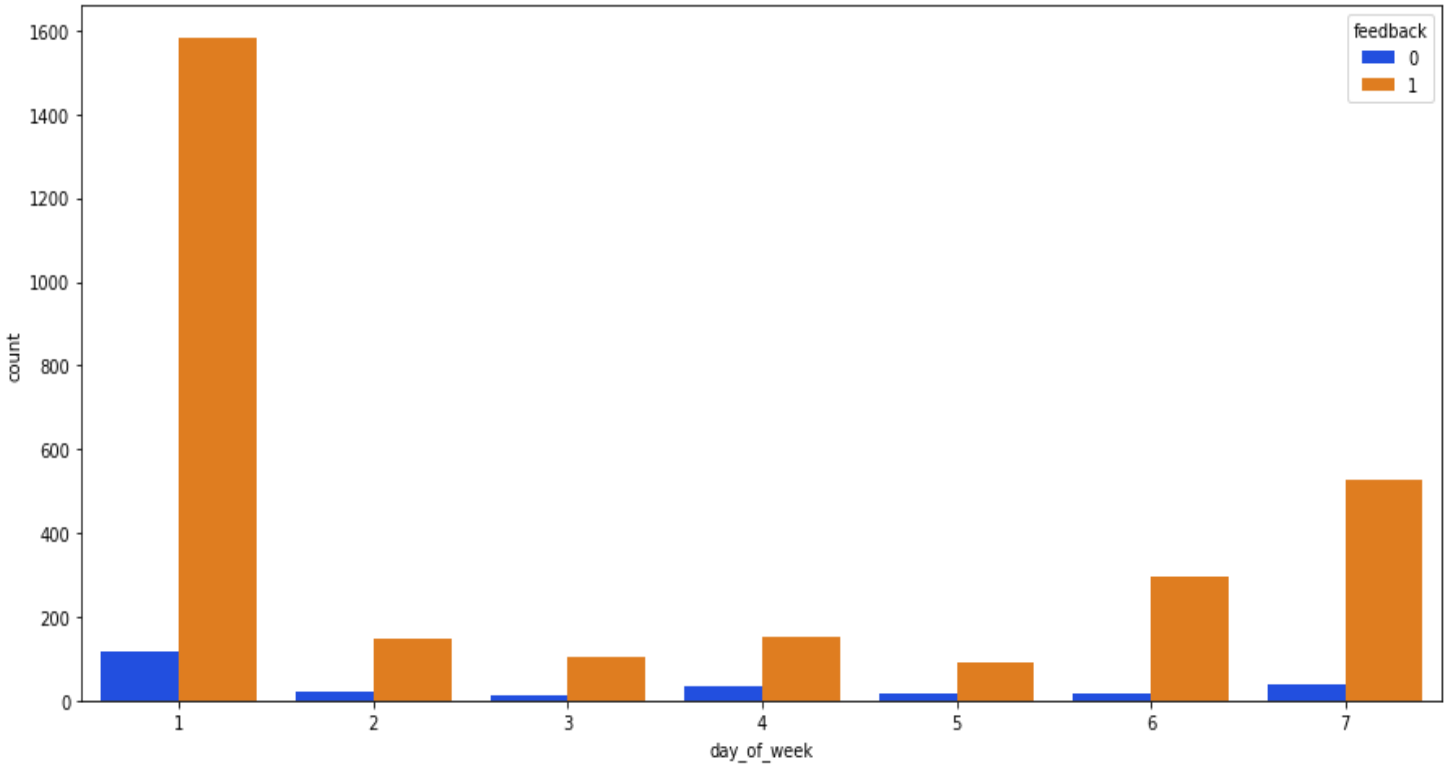


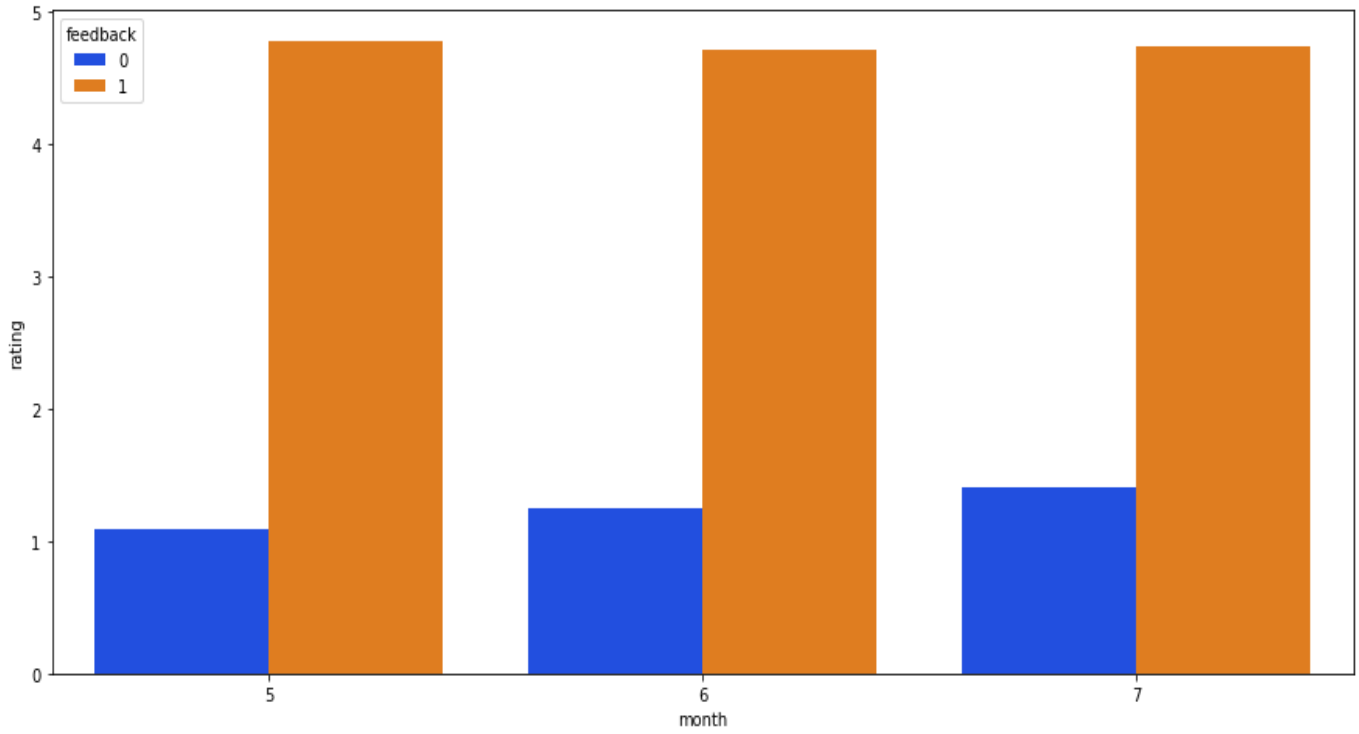
Application of Inferential Statistics

Once I explored the data visually, I began looking to perform statistical analysis on the data. I explored the relationship between the length of the review and the feedback type. Looking at the plot below, we see that customers with negative reviews have a tendency to write longer reviews.



In my data visualization notebook, I looked at the frequency of which day and month customers were more likely to write reviews. I found that customers were most likely to write reviews on Monday and gave feedback on their devices mostly during the month of July.



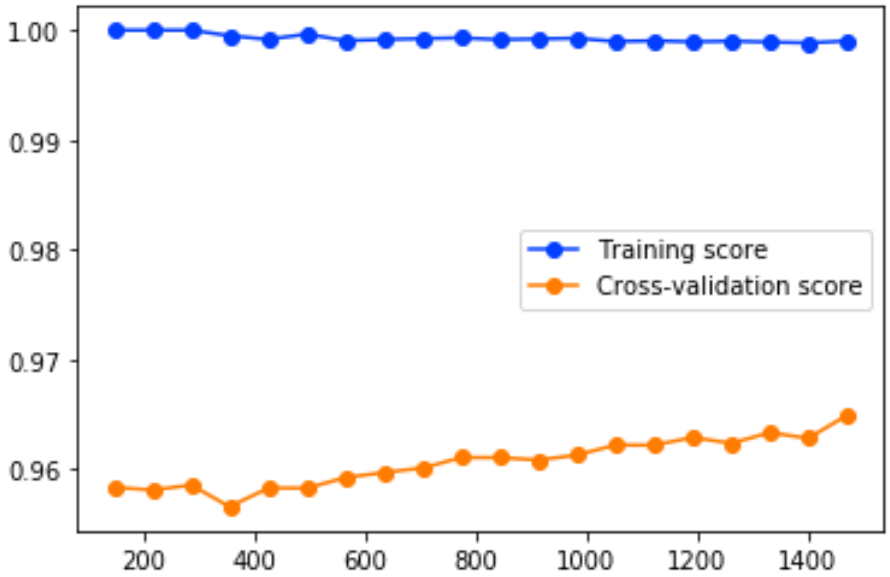


Predictive Modeling

For variation, one hot encoding was also used in my analysis. One hot encoding is a type of vector representation where all the facts in a vector are 0, except for one, which has a 1 in its value. The one important thing to make note of in one hot encoding is that no matter how many dummy variables you end up with, you make sure not to drop any one variable. Doing so may end up in a dummy variable trap. With that being said, it means that there is a possibility of being a case for perfect multicollinearity. This multicollinearity may occur when the independent variables in a regression model are correlated. The correlation becomes a problem due to independent variables needing to be independent. Coefficient estimates may swing based on which other independent variables in the model. These coefficients can become very sensitive to small changes made to the model. Multicollinearity reduces the precision of the coefficients and due to this I may not be able to trust any p-values to identify independent variables that are considered statistically significant.

I decided to look and use both a random forest classifier and gradient boosting classifier in my analysis here. The data was split up into a training set and a testing set first before getting started with random forest. One of the most important methods of random forest classifier in sci-kit learn is feature_importances. With the use of a random forest classifier, I looked at the top 10 features and created the y_pred for the training and test

set. From there, the learning curve of the model was compared between the training and cross-validation score.



As a result from running the model with random forest, the following was the output.

Random Forest Classifier:
Accuracy Score: 0.9216931216931217
Precision Score: 0.9284940411700975
Recall Score: 0.9907514450867052
F1 Score: 0.9586129753914988
Confusion Matrix:
[[14 66]
[8 857]]

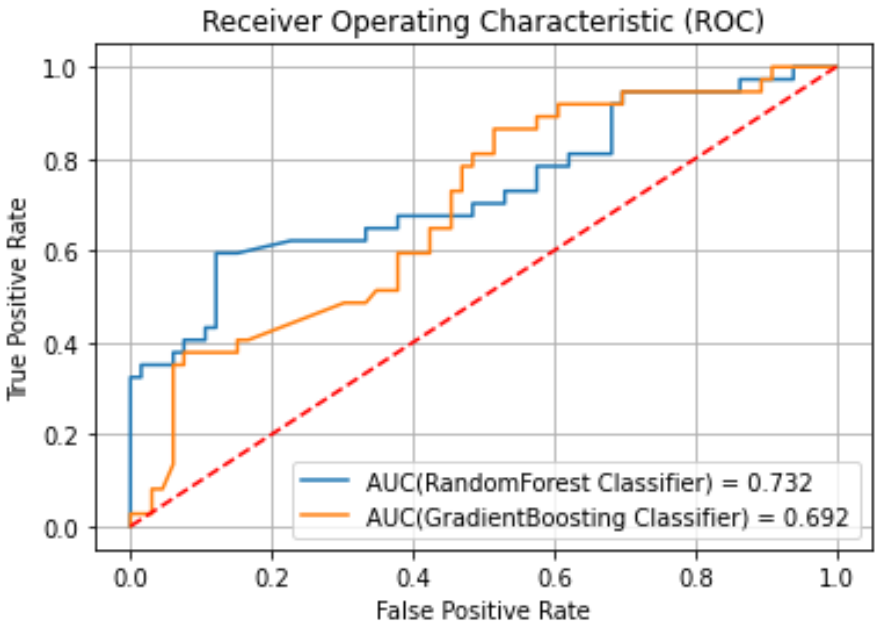
As a result from running the model with gradient boosting, the following was the output.

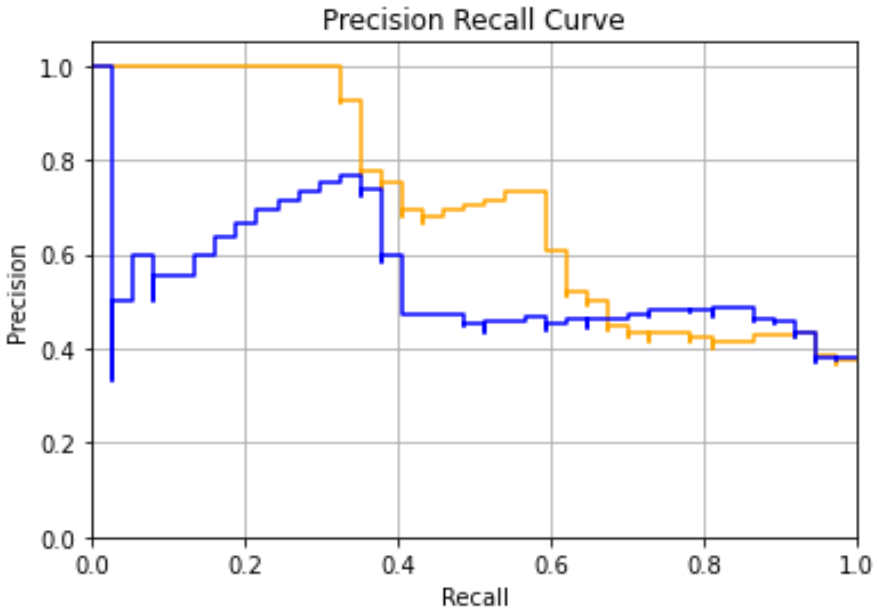
Gradient Boosting Classifier:
Accuracy Score: 0.928042328042328
Precision Score: 0.9289558665231432
Recall Score: 0.9976878612716763
F1 Score: 0.9620958751393535
Confusion Matrix:
[[14 66]
[2 863]]

The models performed well using both methods and scored fairly high in each scoring criteria. From my analysis with the data so far, I can conclude that feature engineering is one of the most crucial steps when it comes to Natural Language Processing. It is now time to further dive into more Natural Language Processing.

Model Performance

I decided to split the dataset again in another notebook. This time I was looking specifically at only the bad reviews to help aid in how to better improve future product versions. The next step involved testing and training models using both random forest classifier and gradient boosting classifier once more. The target variable selected was the 'feedback' column. When running the model using random forest, the accuracy of reviews being classified as negative was around 73%, the precision or the positive predicted value of reviews being negative was 71%, with the recall or sensitivity of relevant reviews being received was around 41%. Under gradient boosting, the accuracy of reviews being negative was around 69%, the precision was around 61%, and the recall was about 38%



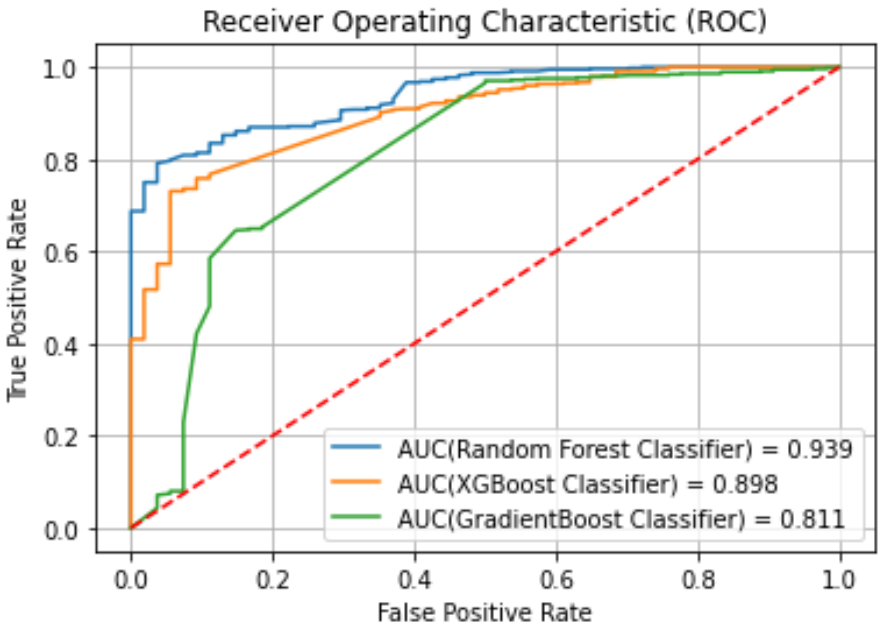


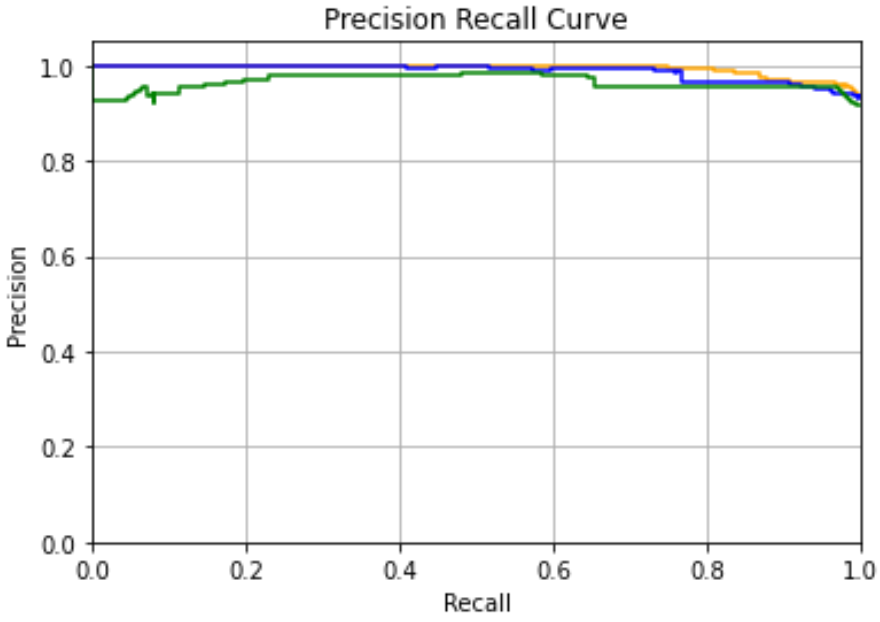
Random Forest: The area under the curve was reported as around 0.73 and the F1 score was around 0.51

Gradient Boosting: The area under the curve was reported as around 0.69 and the F1 score being about 0.47.

Sentiment analysis with BERT was also used in my assessment of this review dataset. As a result, looking at each rating class, the accuracy of being correctly predicted was pretty high at 96.2% for rating 5, 24% for rating 4, 0% for rating 3, 0% for rating 2, and 79.1% for rating 1. As we see here, the accuracies here are varied. To further inspect, I ran 3 models (random forest, XGBoost, and gradient boosting) and plotted the output on an roc curve and precision recall curve.

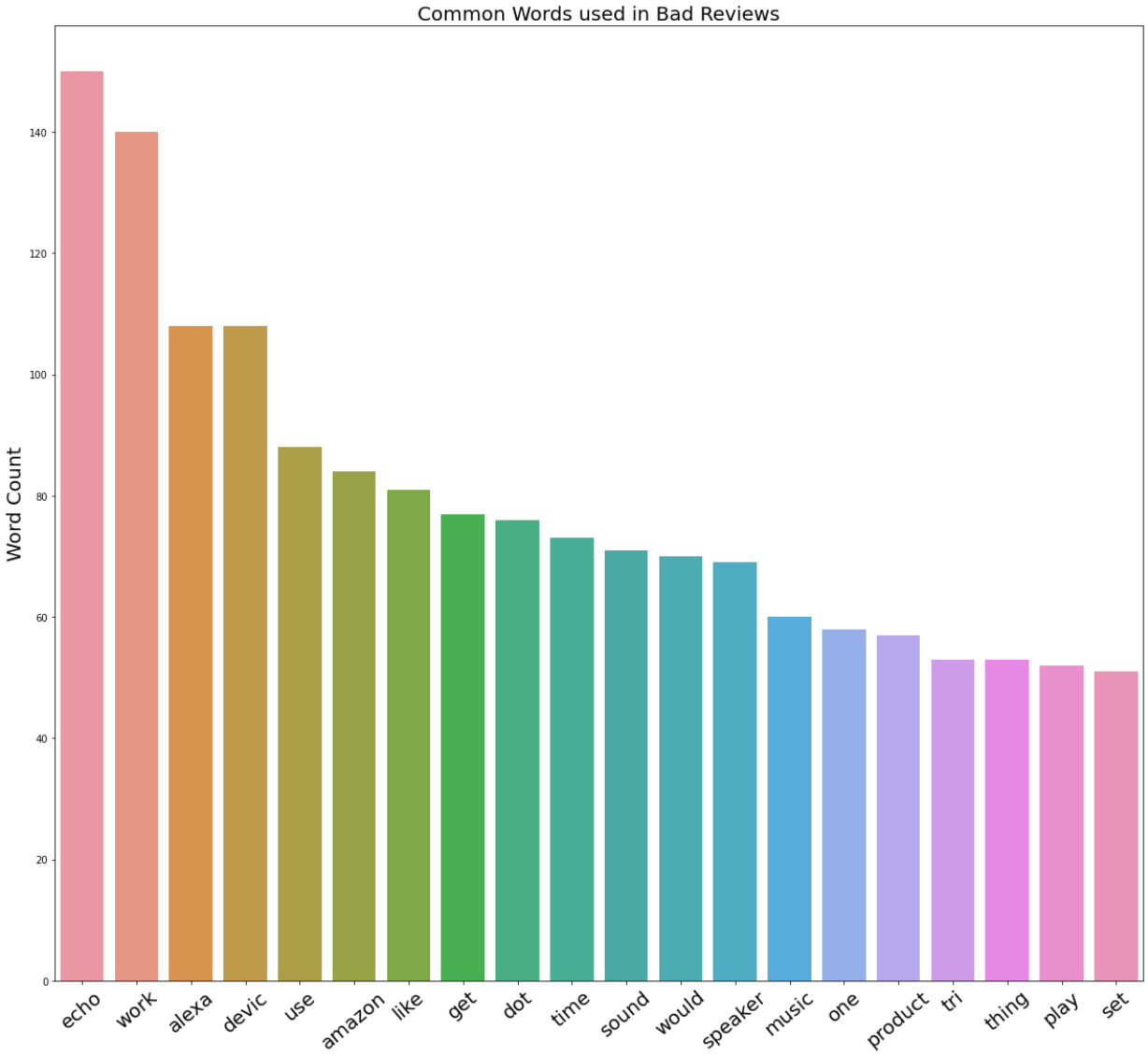
For random forest, the model had an accuracy score of about 94%, precision score of about 95%, and recall score of 99.4%. When running the XGBoost model, it had an accuracy score of about 94%, precision score of about 93.2%, and a recall score of 100%. And lastly, while running the gradient boosting model, the accuracy score was about 93%, precision score was about 95.2%, and the recall score was around 97%. Collectively the three models and their auc score were also pretty high as seen in the curve below.





This sort of makes sense being that there is a larger volume of positive feedback. It does not help us enough though with gaining more insight on negative feedback.

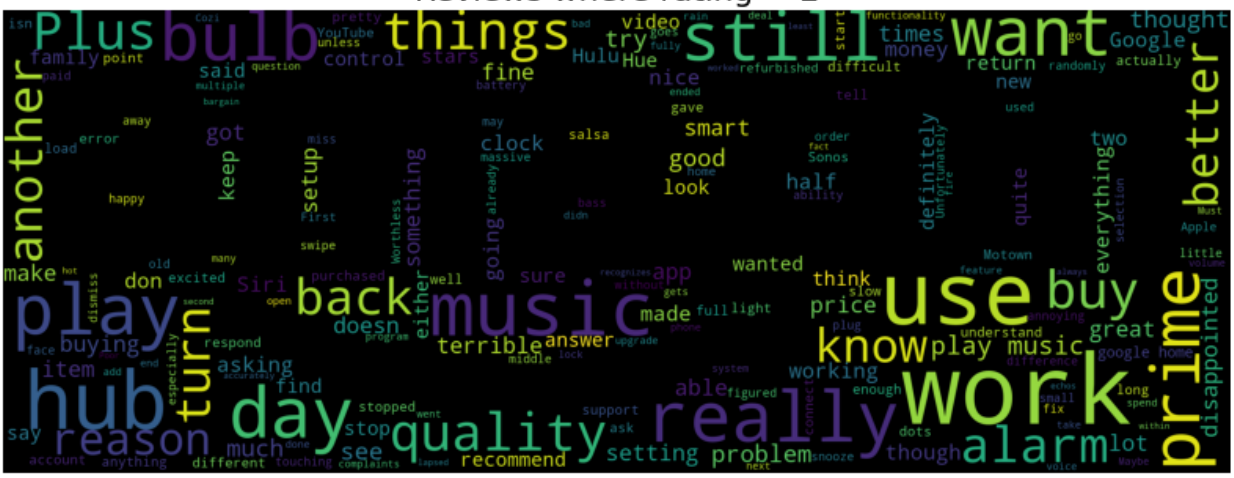
Time to go even deeper and look at word frequency with words in ‘verified_reviews.



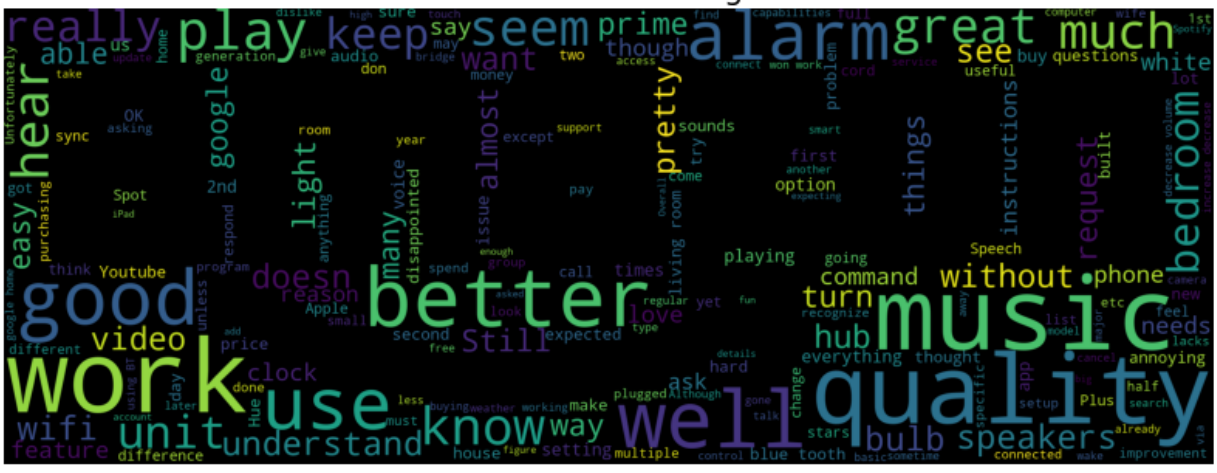
As shown above, the highest count of common words in bad reviews is echo and work. They have been used at least 150 and 140 times in written product reviews across all product variations. A Tfidvectorizer was used to get a general idea of which words show up when using feature extraction from the text. I could also get a look at which words were used the most in the majority of negative feedback using word clouds. To broaden out the frequency of words, reviews with rating 3 were also included since a rating of 3 is not fully positive and could still be classified as negative feedback.

Reviews where rating = 1

Reviews where rating = 2



Reviews where rating = 3



Some of the most frequent common words across ratings 1 through 3 are ‘work’, ‘quality’, and ‘music’.

Conclusion and Recommendations

The takeaway from this exploration is that predicting what determines negative feedback is more than just looking at the rating. Looking at the text used helps get more insight into why a customer chose a certain rating. While the results from the sentiment analysis using BERT were not as helpful as I had initially thought, there were more improvements I could make in my analysis by focusing more on the negative feedback.

Based on performance, the model I would pick would be XGBoost since it had the highest prediction in its confusion matrix for predicting negative feedback with 576 true negatives. With a recall of 1.0, it means that the feedback was relevant pertaining to negative reviews within the dataset.

If I wanted to see if the model could perform better, I could also consider looking at logistic regression in my analysis. When the dataset is not very large, the scope of review feedback can be limiting. So if I were to look at another classification problem, I would probably consider one larger than around 3000 total values in a dataset.