Amazon Review Analysis

Springboard Data Science Career Track: Capstone 2

By: Mythri Partha

I. Introduction/Project Goal

Amazon is one of the largest global conglomerates in the world that specialized in online shopping, electronics, streaming media, and so much more. Amazon revolutionized many industries, and one of the most important features is Amazon reviews, they make it easier to gauge the quality of the product or service you want to purchase. As someone who enjoys technology, I wanted to see if I could make a model that predicts the rating to an electronic product based on its review. Amazon has revolutionized media with products like their Kindle E-Readers, Fire tablets, and Amazon Echo (AKA Alexa), these are the products that I will be observing. This model can be used in order to help electronics companies more accurately see how word choice affects whether or not a review is positive or negative and possibly assign the review a star rating if one were missing. It could also be used by business owners who want to learn how programs respond to word choice when assigning a positive or negative review to their product or service.

II. Data Collection/Wrangling

The complete dataset was found on Data.World and had three csv files that contained the name of the product, rating, review text, review title, number of people who found the review useful, and more. The data was found at this link: https://data.world/datafiniti/consumer-reviews-of-

amazon-products and it was split into three csv files. I read the csv files using the Pandas Package and combined the three dataframes into one. I cleaned my data by seeing where there were missing values and I removed columns that had way too many values missing. I then cleaned up the categories of the electronics by observing their names, making all the letters lowercase, removing special characters, tokenizing the text, finding out more about the products, then setting up a loop to writeup coherent categories such as "E-Reader", "Tablets", and "Personal Assistants". The dataframe started out with over 34,000 rows and 24-27 columns, this was eventually reduced to 12, then to four by the time I got to the preprocessing stage.

III. Exploratory Data Analysis (EDA)

After cleaning the data, I first observed the distribution of star ratings to see how the distribution of ratings were before stepping into modeling. When plotting the star ratings using Python's seaborn package, we can see that there is an exorbitant amount of five-star ratings.

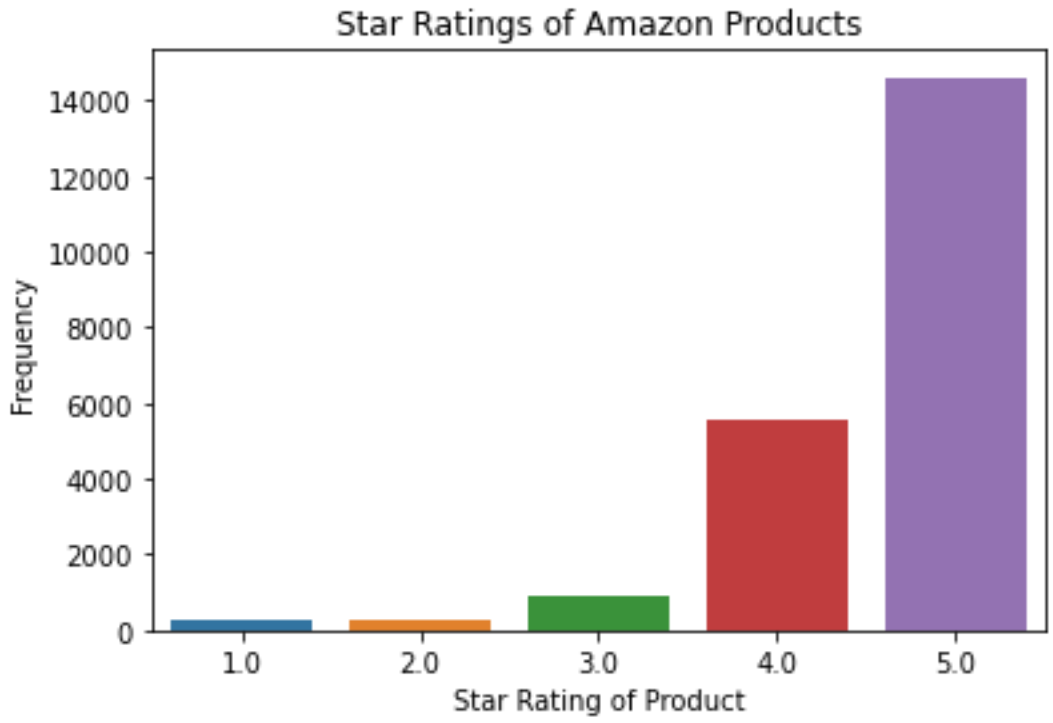


Figure 1: Amazon Electronics' Star Rating distribution

I then wanted to see how many of each product type we were observing, so I plotted a distribution of the types of products in case I wanted to use the categories as a feature for the prediction models. Most of the products reviewed were tablets.

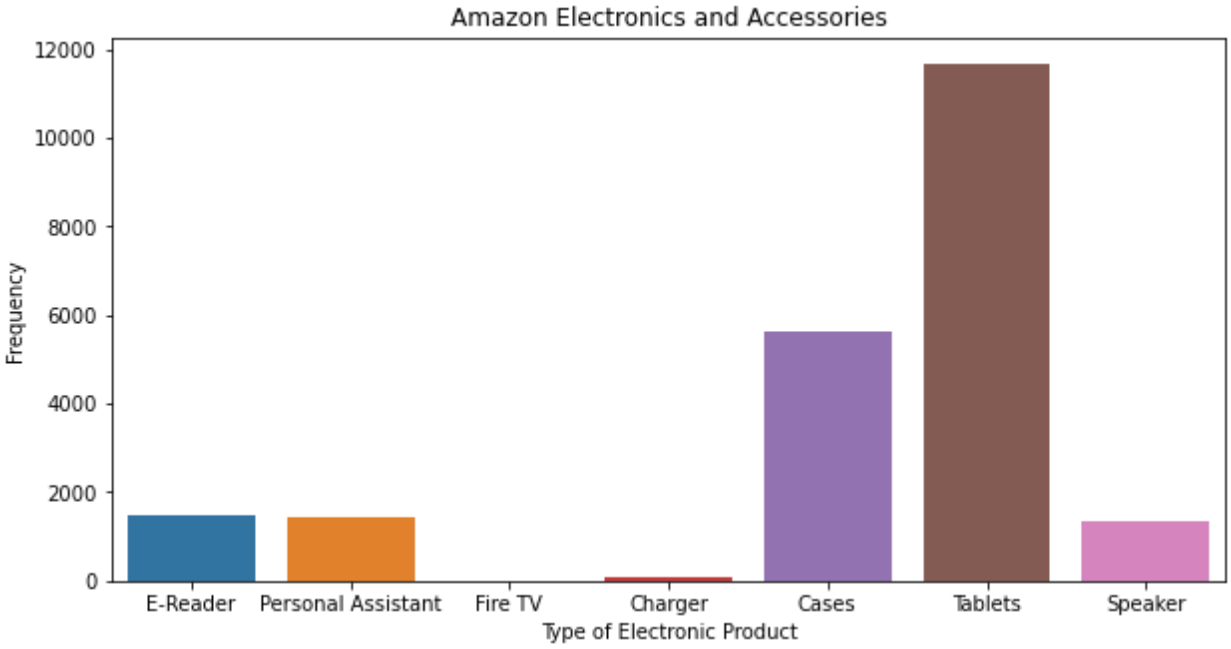


Figure 2: Types of Electronics Analyzed

After seeing the how many of each product was shown in the dataset, I wanted to see how the products were rated, so I made more plots to see the ratings for each product as seen below.

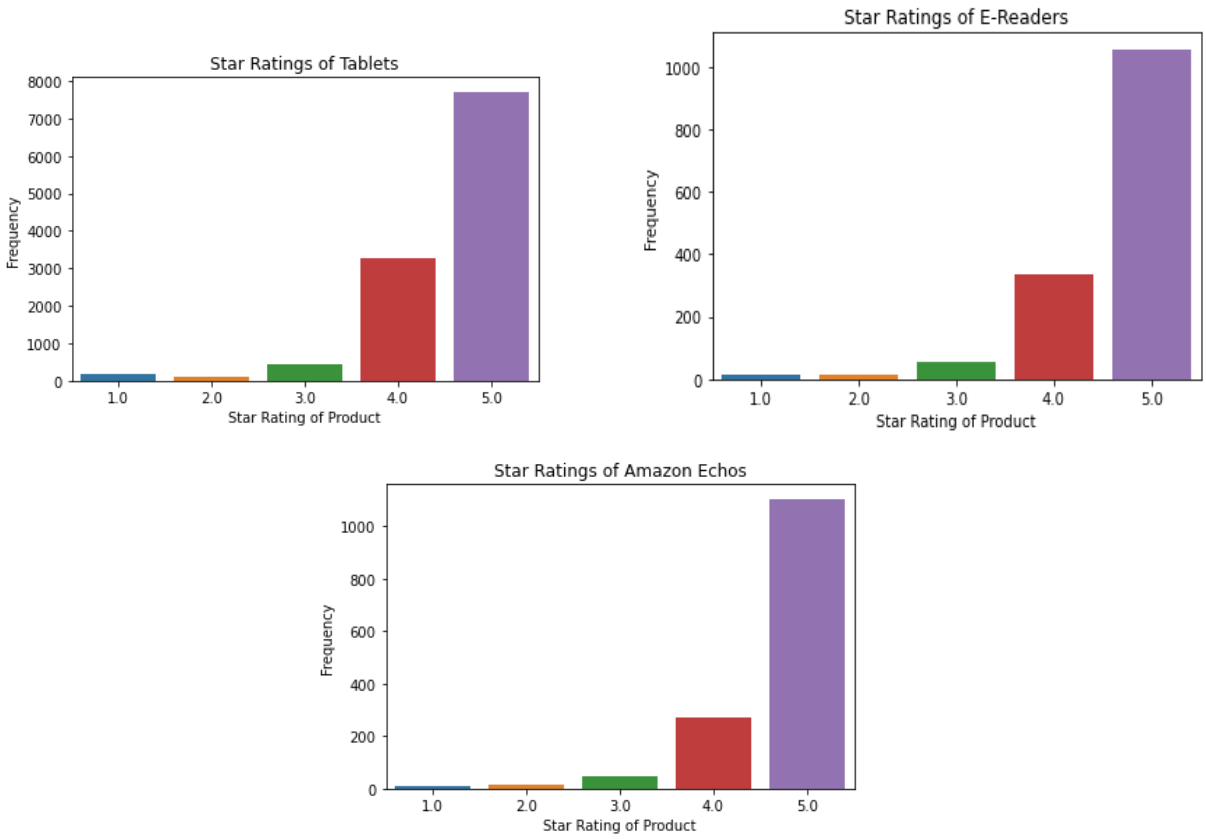


Figure 3: Reviews of Electronics

As seen in figure 3, all electronics have an extreme amount of five star ratings in comparison to the other star ratings.

From there, I wanted to observe the five-star reviews and see what words were common in those reviews by creating a wordcloud from the wordcloud package in Python. It shows us the words that appear in a certain dataset, and the larger the word is, the more it appears in the set you are observing. I created a dataframe that consisted of the five-star reviews and generated my wordcloud. I opted to leave stopwords in the data as words like “not” are

crucial in letting us know if a review is positive or negative.

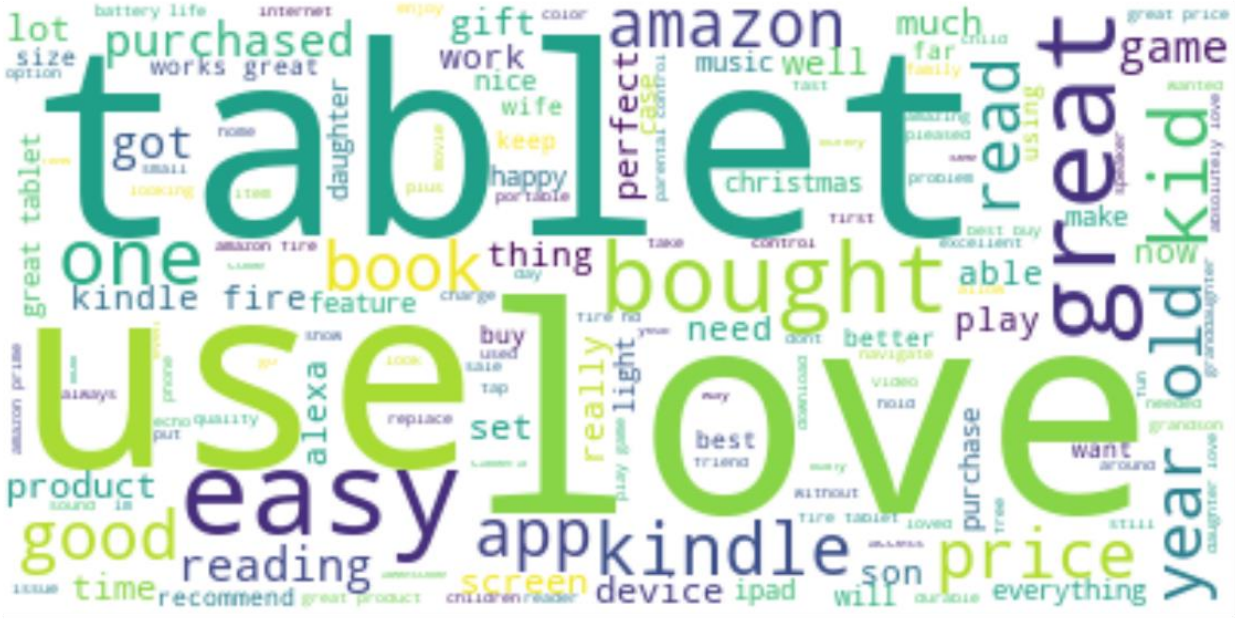


Figure 4: WordCloud for 5-star rated products

When we observe the wordcloud, we can see that the words ‘tablet’, ‘love’, and ‘use’ are the ones that stick out to us almost instantly, meaning that these were the most common in the reviews that were analyzed.

IV. Pre-Processing

In natural language processing text preprocessing is extremely important because we can remove characters and words that are unnecessary from text data and we can feed it into models that can interpret language in the way we want it to. Text pre-processing usually consists of making all the data lowercase, removing stop words (filler words like “the”, “and”, “did”, etc.), expanding contractions, removing spaces, and lemmatizing or stemming (reducing words to their root form). In this case, I made my text lowercase, removed whitespaces, and removed special characters and punctuation. I opted to leave in the stop words, as words like “not” are important in reviews. I also did not lemmatize or stem the

words because the methods to do so can cause errors. After finishing text preprocessing, I established my independent and dependent variables for my models, which were the product reviews and ratings respectively. I ran the review text into a TF-IDF (term frequency-inverse document frequency) vectorizer, which turns the text into numeric vectors that can easily be read by the program, it does so by finding out the originality of a term by comparing the number of times it occurs in the document with the number of documents the term shows up in other documents. The dependent variable was set to the integer value for product ratings. I then scaled the data using the standard scaler method and then split my data into training and test sets. I set the testing data size to 25% and the training set accounts for 75%. From there, I moved on to modeling.

V. Modeling

I started the modeling process with the dataframe by using a LinearSVC model on all of the data, this was the only model I was able to apply to the data as it was too large for Jupyter to handle. I then undersampled the data so that each star rating had the same number of data points for a better model, for this, I utilized LinearSVC, Logistic Regression, KNearest Neighbors, Gradient Boost Classifier, and Random Forest Classifier. After that, I further undersampled by turning the problem into a binary classification problem where, instead of trying to predict the star rating, the model would predict whether or not the review was positive. For binary classification, I completed the same models as the undersampled class, but did not complete KNN. I will detail the classification reports below

Confusion Matrix and Classification report for LinearSVC on all data:

```
[[  59    4    0    1    5]
 [   6   51    0    5    7]
 [   3    3  162   21   32]
 [  10   11   29 1054  290]
 [  34   28   40  379 3173]]
```

	precision	recall	f1-score	support
1	0.53	0.86	0.65	69
2	0.53	0.74	0.61	69
3	0.70	0.73	0.72	221
4	0.72	0.76	0.74	1394
5	0.90	0.87	0.89	3654
accuracy			0.83	5407
macro avg	0.68	0.79	0.72	5407
weighted avg	0.84	0.83	0.83	5407

Figure 5: Classification Report and Confusion Matrix for Linear SVC on all Data

On first glance, the metrics for linear SVC look pretty decent, with a 0.83 accuracy and high precision/recall for products with 4 and 5-star ratings, however, upon further inspection, we can see that the precision for one and two star models just is not the best. Also, for imbalanced models, such as the one we started with, accuracy is not a good metric to base our model off of.

Next, I undersampled the data so that there was the same number of products for each rating, so that there could be less error in prediction models as there will not be a major imbalance.

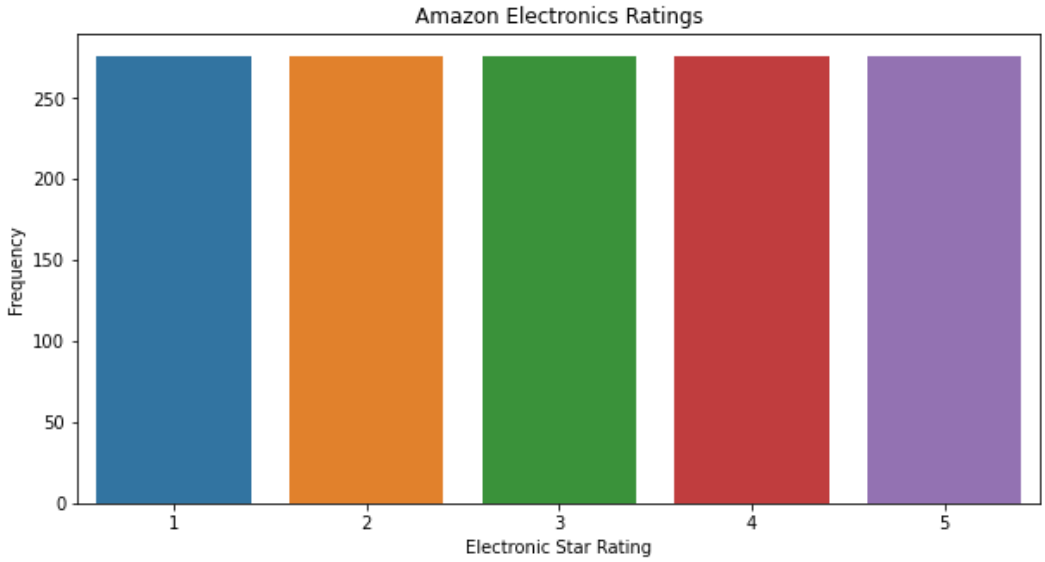


Figure 6: Star Ratings after Resampling

```
[[59  2  1  3  4]
 [ 6 51  3  4  5]
 [ 6  3 30 17 13]
 [ 6  4 10 25 24]
 [ 2  8  7 17 35]]
```

	precision	recall	f1-score	support
1	0.75	0.86	0.80	69
2	0.75	0.74	0.74	69
3	0.59	0.43	0.50	69
4	0.38	0.36	0.37	69
5	0.43	0.51	0.47	69
accuracy			0.58	345
macro avg	0.58	0.58	0.58	345
weighted avg	0.58	0.58	0.58	345

Figure 6: Confusion Matrix and Classification report for LinearSVC on resampled data

Though the accuracy decreased here, it seems that the precision and recall have gotten better for 1 and 2-star reviews, but they have gotten significantly worse for 3, 4, and 5-star reviews. These results are similar to the other resampled models as seen below.

```
[[58  3  3  3  2]
 [ 2 53  6  3  5]
 [ 7  8 37 10  7]
 [ 5  7 13 22 22]
 [ 4  6  8 13 38]]
```

	precision	recall	f1-score	support
1	0.76	0.84	0.80	69
2	0.69	0.77	0.73	69
3	0.55	0.54	0.54	69
4	0.43	0.32	0.37	69
5	0.51	0.55	0.53	69
accuracy			0.60	345
macro avg	0.59	0.60	0.59	345
weighted avg	0.59	0.60	0.59	345

Figure 7: Confusion Matrix and Classification report for Random Forest Classifier on resampled data

```
[[62  1  4  2  0]
 [ 6 54  3  3  3]
 [ 9  9 32 12  7]
 [ 6  7  8 25 23]
 [ 3  8  8 13 37]]
```

	precision	recall	f1-score	support
1	0.72	0.90	0.80	69
2	0.68	0.78	0.73	69
3	0.58	0.46	0.52	69
4	0.45	0.36	0.40	69
5	0.53	0.54	0.53	69
accuracy			0.61	345
macro avg	0.59	0.61	0.60	345
weighted avg	0.59	0.61	0.60	345

Figure 8: Confusion Matrix and Classification report for Gradient Boost Classifier on resampled data

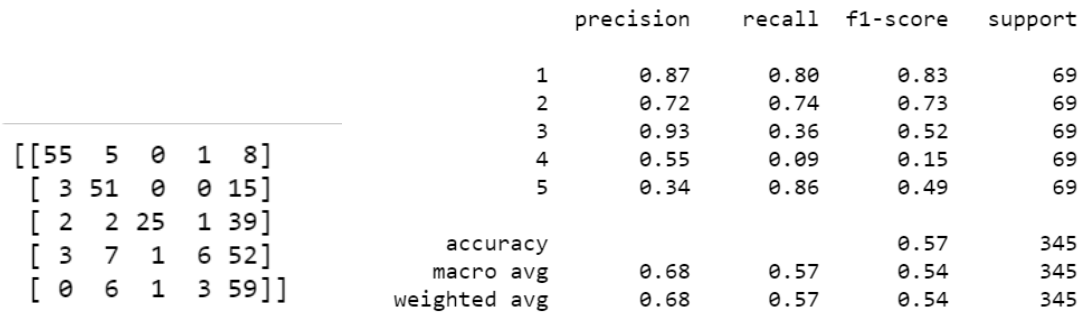
Gradient Boost Classifier, Random Forest both are extremely similar in their metrics, with Gradient Boost slightly edging out over random forest because its precision is higher, the model accuracies are nearly the same as well. Choosing either of these models would be best if we needed to create a model that could predict a star rating.

```
[[61  1  2  3  2]
 [ 5 51  4  2  7]
 [ 4  3 33 16 13]
 [ 5  3 10 28 23]
 [ 2  9  6 17 35]]
```

	precision	recall	f1-score	support
1	0.79	0.88	0.84	69
2	0.76	0.74	0.75	69
3	0.60	0.48	0.53	69
4	0.42	0.41	0.41	69
5	0.44	0.51	0.47	69
accuracy			0.60	345
macro avg	0.60	0.60	0.60	345
weighted avg	0.60	0.60	0.60	345

Figure 9: Confusion Matrix and Classification report for Logistic Regression on resampled data

Logistic Regression also boasts very similar results to our previous models, however the precision and recall for higher rated products is significantly lower than those for random forest and gradient boost.



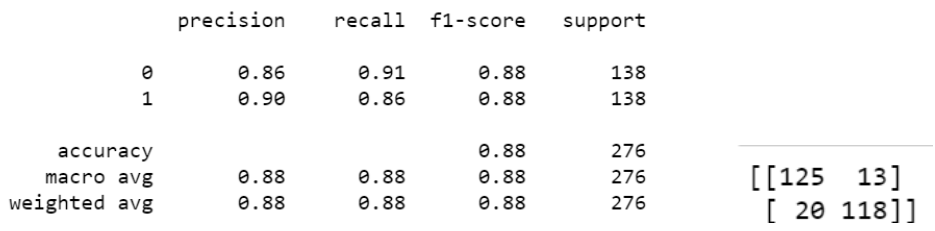
```
[[55  5  0  1  8]
 [ 3 51  0  0 15]
 [ 2  2 25  1 39]
 [ 3  7  1  6 52]
 [ 0  6  1  3 59]]
```

	precision	recall	f1-score	support
1	0.87	0.80	0.83	69
2	0.72	0.74	0.73	69
3	0.93	0.36	0.52	69
4	0.55	0.09	0.15	69
5	0.34	0.86	0.49	69
accuracy			0.57	345
macro avg	0.68	0.57	0.54	345
weighted avg	0.68	0.57	0.54	345

Figure 10: Confusion Matrix and Classification report for K-Nearest Neighbors on resampled data

K-Nearest Neighbors was a wildcard, its metrics are extremely scattered and radical especially between its precision and recall for products with 3, 4, or 5-star ratings. This model would not be even considered to be put out into production.

After working through the more even models, I then wanted to try turning this into a binary classification problem. I did this by copying my undersampled dataframe, then I removed the three star rating column altogether and assigned the number 1 to positive reviews (those with either 4 or 5 stars) and the number 0 to negative reviews (those with 1 or 2 stars).



	precision	recall	f1-score	support
0	0.86	0.91	0.88	138
1	0.90	0.86	0.88	138
accuracy			0.88	276
macro avg	0.88	0.88	0.88	276
weighted avg	0.88	0.88	0.88	276

```
[[125  13]
 [ 20 118]]
```

Figure 11: Confusion Matrix and Classification report for Gradient Boost (Binary Classification)

	precision	recall	f1-score	support
0	0.90	0.89	0.90	138
1	0.89	0.91	0.90	138
accuracy			0.90	276
macro avg	0.90	0.90	0.90	276
weighted avg	0.90	0.90	0.90	276

```
[[123  15]
 [ 13 125]]
```

Figure 12: Confusion Matrix and Classification report for Random Forest (Binary Classification)

	precision	recall	f1-score	support
0	0.81	0.85	0.83	138
1	0.84	0.80	0.82	138
accuracy			0.83	276
macro avg	0.83	0.83	0.83	276
weighted avg	0.83	0.83	0.83	276

```
[[117  21]
 [ 27 111]]
```

Figure 13: Confusion Matrix and Classification report for Logistic Regression (Binary Classification)

	precision	recall	f1-score	support
0	0.87	0.89	0.88	138
1	0.89	0.87	0.88	138
accuracy			0.88	276
macro avg	0.88	0.88	0.88	276
weighted avg	0.88	0.88	0.88	276

```
[[123  15]
 [ 18 120]]
```

Figure 14: Confusion Matrix and Classification report for Linear SVC (Binary Classification)

Overall, there was a large increase in prediction power once the data was interpreted in a binary problem, every model has high accuracy, precision, and recall. All their confusion matrices show a low number of false positives and negatives. The gradient boost and random forest models showed the best metrics and that would be the one I would proceed with as a prediction model for predicting Amazon reviews. I then wanted to use TF-IDF and Normalized Count Occurrence, I opted to leave stopwords in these visuals as well to keep consistent with my models. The word importance figures are shown below.

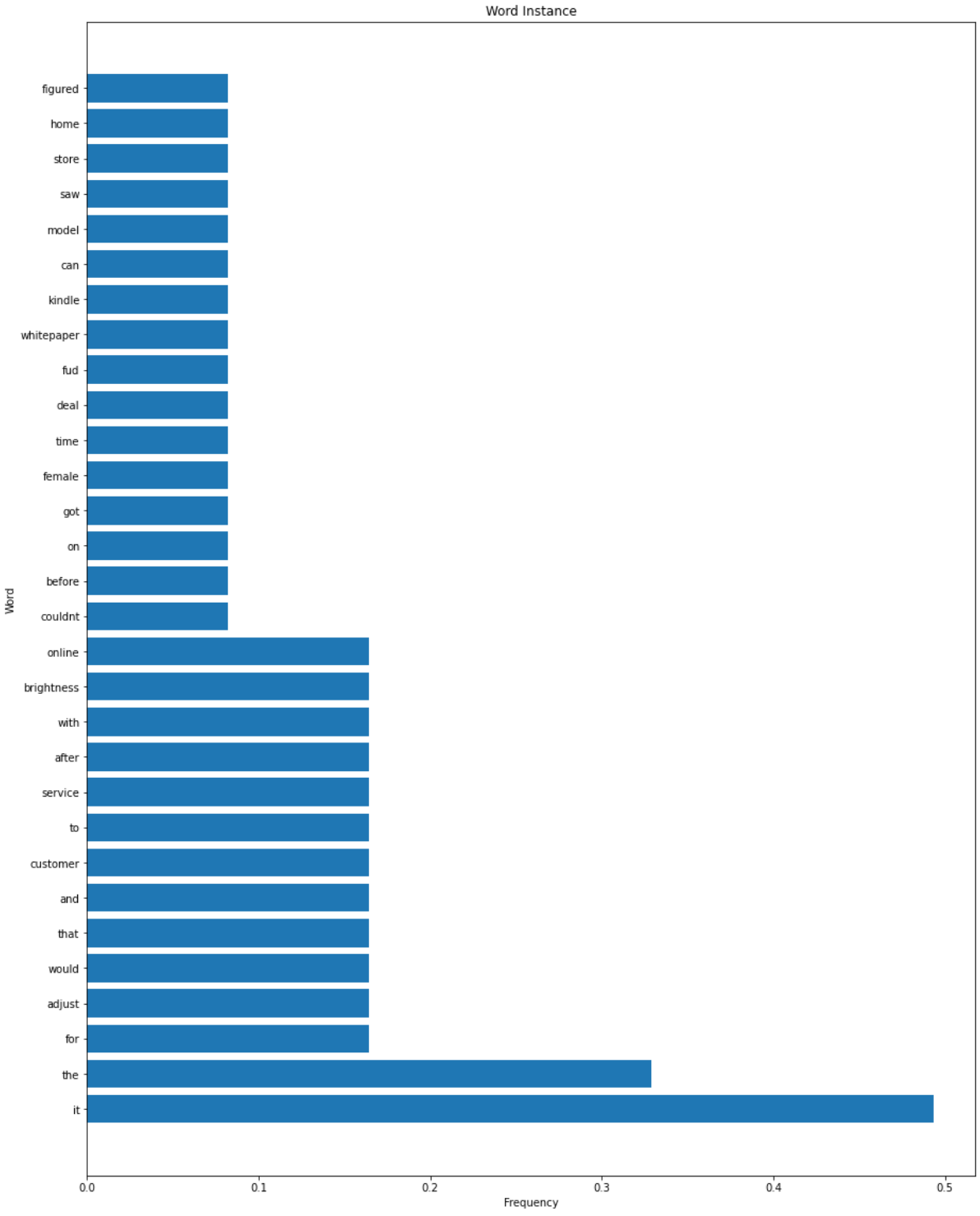


Figure 15: Feature Importance using Normalized Count Occurrence

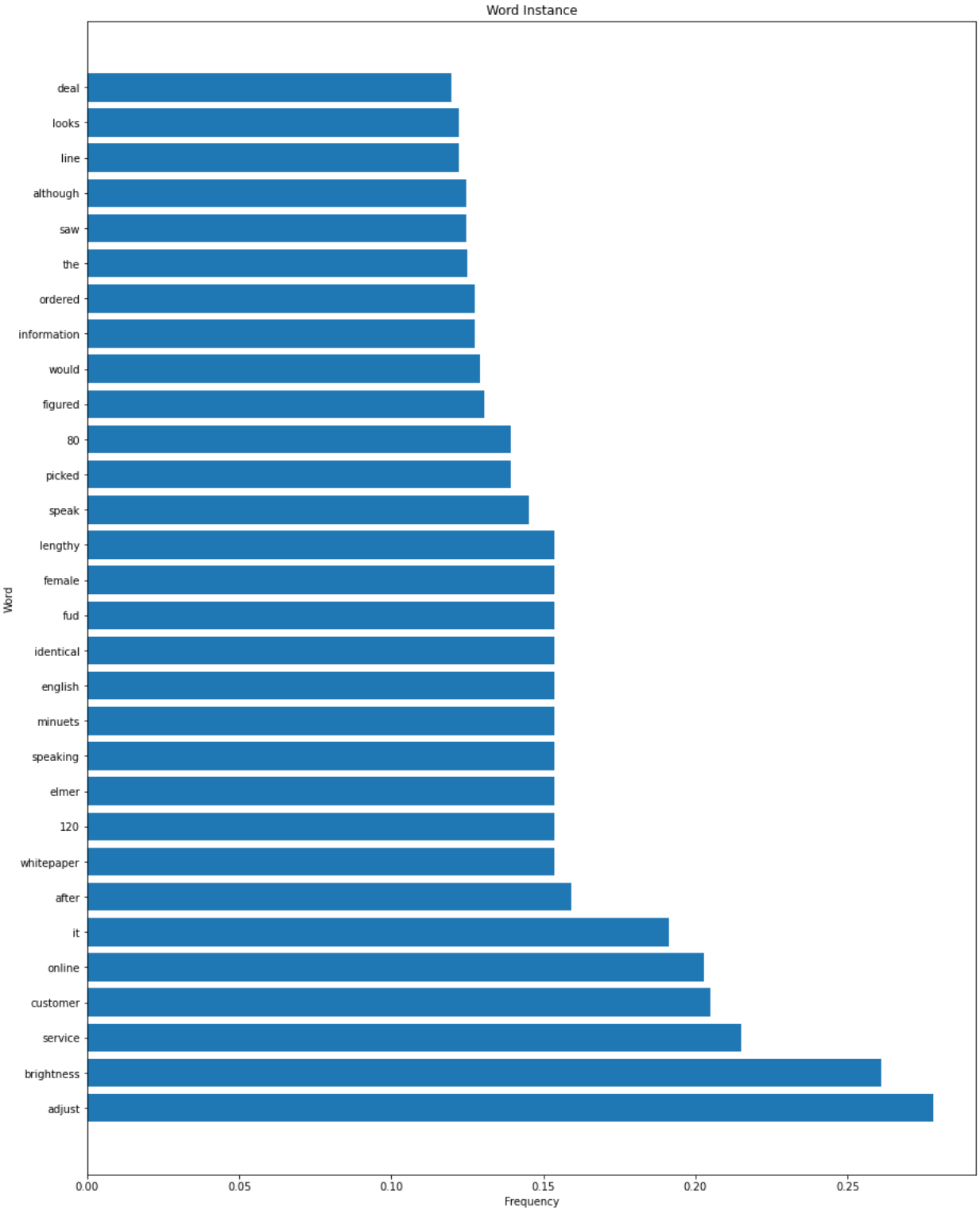


Figure 16: Feature Importance using TF-IDF

The most important text features as shown in our TF-IDF representation are adjustments, brightness, and service.

VI. Conclusion and Future improvements

The work I have completed used data from Amazon and I utilized machine learning methods, natural language processing, clustering, and classification methods. As with any other machine learning model, this model is not truly complete, there can always be future improvements and additions to the models. For example, I could create a recommender system based on these reviews to help consumers in choosing a product to purchase. In the realm of natural language processing, I could have added some more steps to the text pre-processing method like removing stopwords and numbers to see how the models change. I also could have tested more models by using another service with higher computing power than Jupyter Notebook. I also could have performed sentiment analysis to see a deeper tone in reviews that goes past just positive or negative. I could also implement some deep learning models in the future.

Overall, the best model was the binary random forest with 90% accuracy, average precision and recall for positive and negative reviews. That model will be the one used to create an app that can predict the rating of the model.