Amazon Customer Reviews

Course: AML 2304 Natural Language Processing

Assignment 01

Prepared for

Lambton College in Toronto

Prepared by

Ashvin Alex (C0880169)

Bastian Castillo (C0872274)

Darshan Ruparel (C087232)

Fadernel Bedoya (C0872455)

Marcelo Munoz (C0873813)

Professor

Vahid Hadavi

March 25, 2023

Lambton College

Table of Contents

Introduction

Through this document, different techniques of Natural Processing Language were employed on Amazon Customer Reviews. These techniques include data preprocessing, visualization, and word vectorization of each review to train various machine-learning models and classify the reviews as positive or negative.

Collecting and Saving Data

The dataset chosen to analyze corresponds to Amazon Customer Reviews. There are multiple categorizations of product reviews. In this case, it was picked the categoric of Wireless Products contains 9,002,021 instances. Due to the limited computational resources, there were selected around 20,000 instances. To obtain this subset, a python script was written to read the file, and according to the **star_rating** field, 4000 instances per category were added to a list and then saved to a CSV file. This file was saved in GitHub as a public repository and can be found in this link. This way, there are enough data for each class for the classification models. The descriptions of the columns are the following:

1. **marketplace** - 2 letter country code of the marketplace where the review was written.
2. **customer_id** - Random identifier that can be used to aggregate reviews written by a single author.
3. **review_id** - The unique ID of the review.
4. **product_id** - The unique Product ID the review pertains to. In the multilingual dataset, the reviews for the same product in different countries can be grouped by the same product_id.
5. **product_parent** - Random identifier that can be used to aggregate reviews for the same product.
6. **product_title** - Title of the product.
7. **product_category** - Broad product category that can be used to group reviews (also used to group the dataset into coherent parts).
8. **star_rating** - The 1-5 star rating of the review.

9. **helpful_votes** - Number of helpful votes.
10. **total_votes** - Number of total votes the review received.
11. **vine** - Review was written as part of the Vine program.
12. **verified_purchase** - The review is on a verified purchase.
13. **review_headline** - The title of the review.
14. **review_body** - The review text.
15. **review_date** - The date the review was written.

The columns review_body and star_rating were selected to continue with this NLP project.

Methodology

The methodology applied through this project comprised the following steps.

Missing values

After loading the dataset, the resulting number of rows was 19,942. This was due to errors in parsing the data. The percentage of missing values contained in these rows does not exceed 0.035%:

marketplace	0.000000
customer_id	0.005015
review_id	0.005015
product_id	0.005015
product_parent	0.005015
product_title	0.005015
product_category	0.005015
star_rating	0.005015
helpful_votes	0.005015
total_votes	0.005015
vine	0.005015
verified_purchase	0.005015
review_headline	0.005015
review_body	0.030087
review_date	0.005015

The total of rows with missing values is 6. Because of the low percentage of missing values, the rows with NaN values were omitted, getting 19,936.

EDA

Graphical exploration of the dataset was done before word vectorization and using the new data frames obtained by vectorizing the text from reviews using libraries like matplotlib and seaborn. These graphs and findings will be presented in the Visualizing Data section.

Text Pre-processing

Using text pre-processing techniques to remove punctuation, and numbers, fixing the lengthening of words, checking the word's spelling, word removal, POS, and lemmatization were used to prepare the data to be vectorized.

Word vectorization

Bag of Words and TF-IDF were used to vectorize the text reviews. Bag of Words counts the frequency of words in the documents, while TF-IDF also considers the context of the words by assigning a weight to each word considering the frequency on all documents. The resulting data frames obtained were used as inputs to the models and to compare the results.

Model Training

Four classification models were chosen: Random Forest, Gaussian Naive Base, Logistic Regression and Multinomial Naive Base. The Bag of Words data frame was used as input for the first two, and the rest used TF-IDF. The target data originally was five classes with 1–5-star ratings; after applying multiple times with different hyperparameters, like the length of the words, the models never reached an accuracy greater than 46%. So, it was decided to reduce the number of classes and experiment with two classes (Negative: 1 – Positive: 0) using Bag of Words and three classes using TF-IDF (Negative: 0, Neutral: 1, Positive: 2). This increased the accuracy of the models as will be shown in the Training Model and Testing section.

Test Model

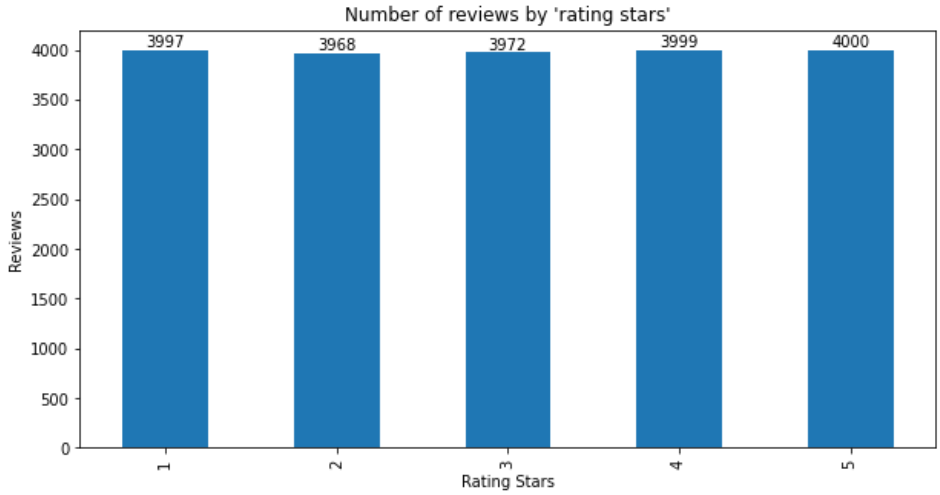
After training and saving the models, selected test data, previously reserved, was used to test the model with the best performance to evaluate how the model was generalizing against new data.

Visualizing Data

This section presents the data graphically to explore it and get some findings.

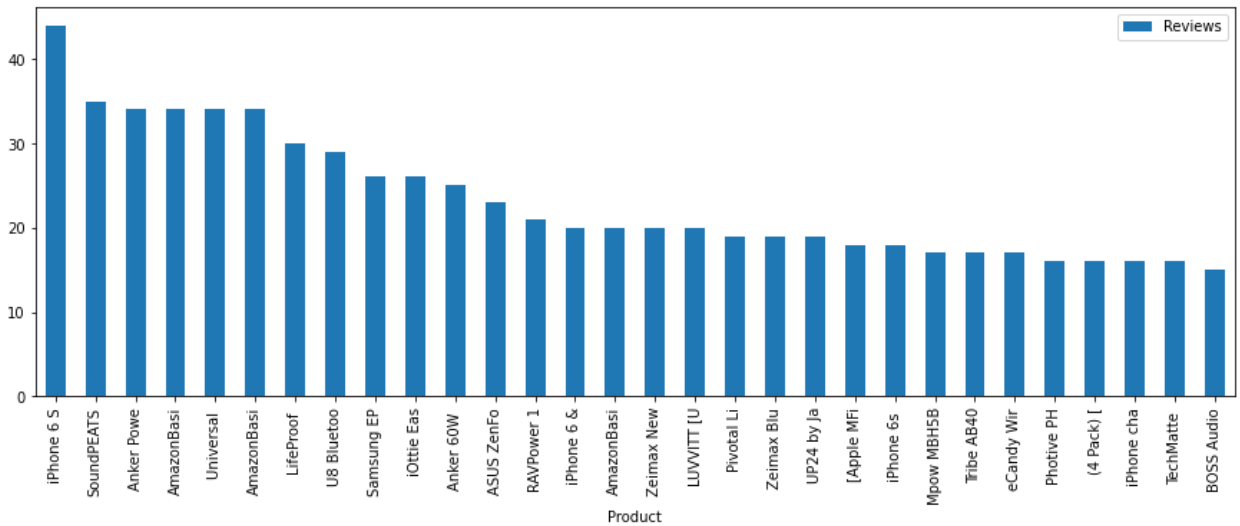
First, the graph below shows the homogeneity of the data used in the project. The five classes have around 4000 instances.

Figure 1 - Number of reviews by Rating



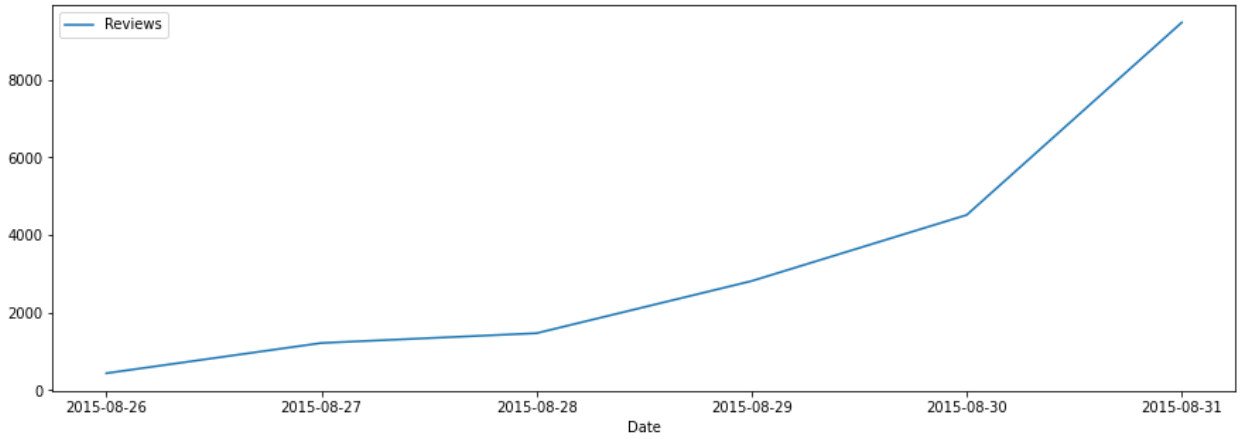
Counting the number of reviews per product, iPhone 6 concentrates the maximum number of reviews. Also, can be seen the next top 29 products with more reviews.

Figure 2 - Top 30 Products with more reviews



The dataset contains the reviews between 08-26-2015 to 08-31-2015; most of the reviews correspond to days 30th and 31st, according to the following graph:

Figure 3 - Distribution of reviews by date



This cloud of words shows the frequency of the words in all the documents (reviews) based on the Bag of Word data frame after word vectorization and previous applying feature selection:

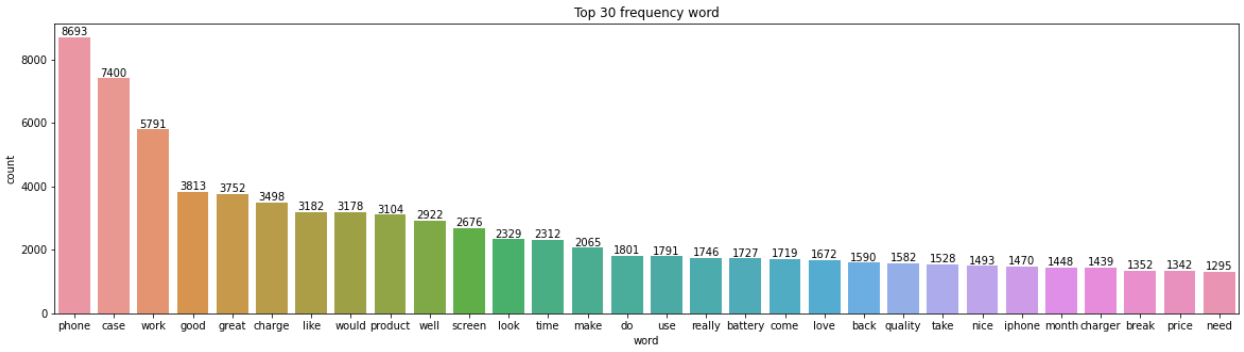
Figure 4 - Word Frequency (previous feature selection)

So, the words with the most frequency are: activate, active, adhere, absorb, adult, acute, advertise, and adjust, among others. After applying a feature selection using a frequency dictionary, the resulting Bag of Words can be represented like this:

Figure 5 - Word Frequency (after feature selection)

Now the words with the most frequency are phone, charger, look, time, work, break, etc. The graph below represents the frequency in another way:

Figure 6 - Word frequency (Bar)

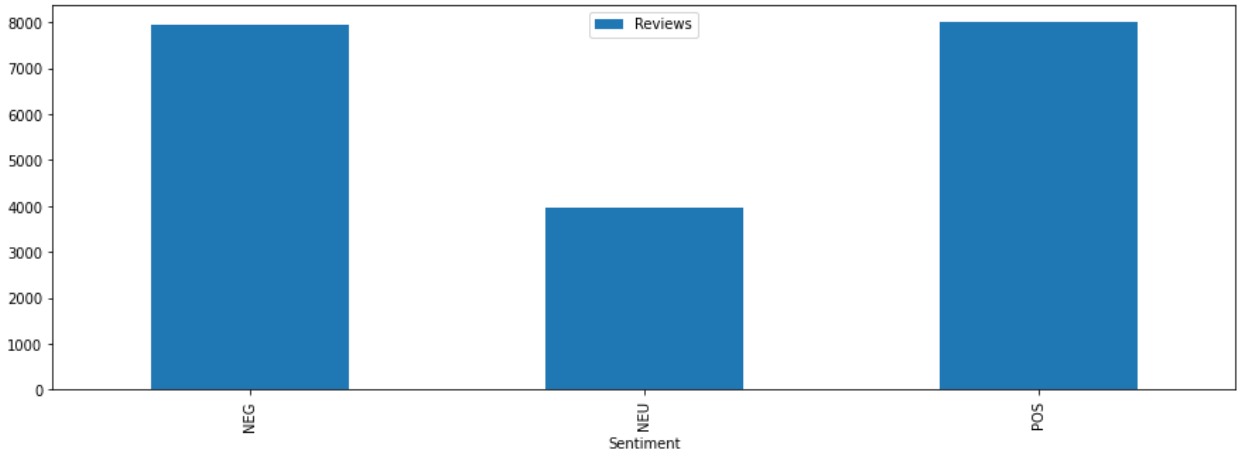


The classes were reduced by discretizing their values according to these rules:

- rating < 3: 0 – Negative
- rating = 3: 1 – Neutral
- rating > 3: 2 – Positive

As a result of applying this discretization, the Positive and Negative reviews from the dataset seem to be even, but there are less records of Neutral values. This can impact the performance of the models in terms of Neutral predictions.

Figure 7 - Sentimental Trends



Word Vectorization

Before vectorizing words, the text from each review had to be prepared using these techniques:

1. **Removing punctuation and numbers**: punctuation characters and numbers were replaced by empty strings using regular expression patterns.
2. **Fixing lengthening**: some words were fixed by removing extra letters and using a regular expression pattern and replace function.
3. **Spell checker**: the words were fixed in term of spelling based on a frequency dictionary and SymSpell library.
4. **Tokenization and stop-words**: the resulting documents, after being cleaned, were tokenized, and the words that did not provide significant meaning were removed from the documents using the NLTK library.

Figure 8 - Tokenization

```
Sample: first 10 tokenized rows:
(['embarrassed', 'admit', 'recently', 'negative', 'opinion', 'self', 'sticks', 'pod
(['fits', 'iphone', 'well'], 5)
(['great', 'charger', 'easily', 'charges', 'samsung', 'galaxy', 'works', 'perfectly
(['great', 'price'], 5)
(['great', 'case', 'better', 'customer', 'service'], 5)
(['easy', 'great', 'functions', 'price'], 4)
(['works', 'great'], 5)
(['good', 'essentially', 'identical', 'replaced', 'another', 'company', 'stopped',
(['seems', 'durable', 'always', 'harder', 'right', 'people', 'make', 'also', 'send'
(['super', 'thin', 'lightweight', 'intrusive', 'feels', 'amazing', 'protective', 'ca
(['good', 'samsung', 'galaxies', 'opinion'], 5)
```

5. **Part of the Speech**: the type or classification of each word about the speech was attached to each word. This part is essential previous applying a lemmatization technique.

Figure 9 - POS

```
Sample: first 10 part-of-speech (POS) classified rows:
([('embarrassed', 'JJ'), ('admit', 'NN'), ('recently', 'RB'), ('negative', 'JJ'), (
([('fits', 'NNS'), ('iphone', 'VBP'), ('well', 'RB')], 5)
([('great', 'JJ'), ('charger', 'NN'), ('easily', 'RB'), ('charges', 'VBZ'), ('samsu
([('great', 'JJ'), ('price', 'NN')], 5)
([('great', 'JJ'), ('case', 'NN'), ('better', 'RBR'), ('customer', 'NN'), ('service
([('easy', 'JJ'), ('great', 'JJ'), ('functions', 'NNS'), ('price', 'NN')], 4)
([('works', 'NNS'), ('great', 'JJ')], 5)
([('good', 'JJ'), ('essentially', 'RB'), ('identical', 'JJ'), ('replaced', 'VBD'),
([('seems', 'VBZ'), ('durable', 'JJ'), ('always', 'RB'), ('harder', 'VB'), ('right'
([('super', 'NN'), ('thin', 'JJ'), ('lightweight', 'JJ'), ('intrusive', 'JJ'), ('fe
([('good', 'JJ'), ('samsung', 'NN'), ('galaxies', 'NNS'), ('opinion', 'NN')], 5)
```

6. **Lemmatization**: this technique was used to obtain the base of each word using the POS obtained in the previous step. This was done with the NLTK library, and these are some samples:

Figure 10 - Lemmatization

```
Sample: first 10 lemmatized rows:
(['embarrassed', 'admit', 'recently', 'negative', 'opinion', 'self', 'stick', 'pod',
(['fit', 'iphone', 'well'], 5)
(['great', 'charger', 'easily', 'charge', 'samsung', 'galaxy', 'work', 'perfectly',
(['great', 'price'], 5)
(['great', 'case', 'well', 'customer', 'service'], 5)
(['easy', 'great', 'function', 'price'], 4)
(['work', 'great'], 5)
(['good', 'essentially', 'identical', 'replace', 'another', 'company', 'stop', 'work
(['seem', 'durable', 'always', 'harder', 'right', 'people', 'make', 'also', 'send',
(['super', 'thin', 'lightweight', 'intrusive', 'feel', 'amaze', 'protective', 'case'
(['good', 'samsung', 'galaxy', 'opinion'], 5)
```

Bag of Words and TF-IDF were the techniques employed to vectorize the lemmatized list of words. As was mentioned in the Methodology section, it was decided to reduce the number of classes and experiment with two classes (Negative: 1 – Positive: 0) using Bag of Words and three classes using TF-IDF (Negative: 0, Neutral: 1, Positive: 2). These are some samples of both:

Figure 11 - Bag of Words

	phone	case	work	good	great	charge	like	would	product	well	...	administrator	aficionado	poof	cleaned	reacts	soybean	skipping	intermixed	invitation	TARGET
0	3	0	0	1	0	3	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	1	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
4	0	1	0	0	1	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19931	0	0	1	0	0	2	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
19932	1	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
19933	1	3	0	0	1	0	3	0	0	0	...	0	0	0	0	0	0	0	0	0	0
19934	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
19935	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

19936 rows × 7001 columns

Figure 12 - TF-IDF

	abandon	abbr	ability	able	abler	abroad	absolute	absolutely	absorb	absorbent	...	zillion	zip	ziploc	zipper	zippered	zippy	zombie	zone	zoom	TARGET
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19931	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
19932	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
19933	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
19934	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
19935	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

19936 rows × 7001 columns

Seven thousand features were selected using a frequency dictionary after multiple iterations and because of the limit of hardware resources.

Lambton College

Training Model and Testing

As was mentioned, four models were used. The following table summarizes the dataset, type of classification, classes, and models:

Dataset	Classification	Classes	Model
Bag of Words	Binary	0, 1	Random Forest Classifier
			Gaussian Naive Base
TF-IDF	Multiclass	0, 1, 2	Logistic Regression Multiclass
			Multinomial Naive Base

80% of the data was used for training and 20% for validation. After training, the models this was the accuracy for each of them:

Dataset	Model	Accuracy
Bag of Words	Random Forest Classifier	77.6%
	Gaussian Naive Base	45.1%
TF-IDF	Logistic Regression Multiclass	69.8%
	Multinomial Naive Base	69.0%

Here can be seen the confusion matrices of each model:

Figure 13 - CM RFC

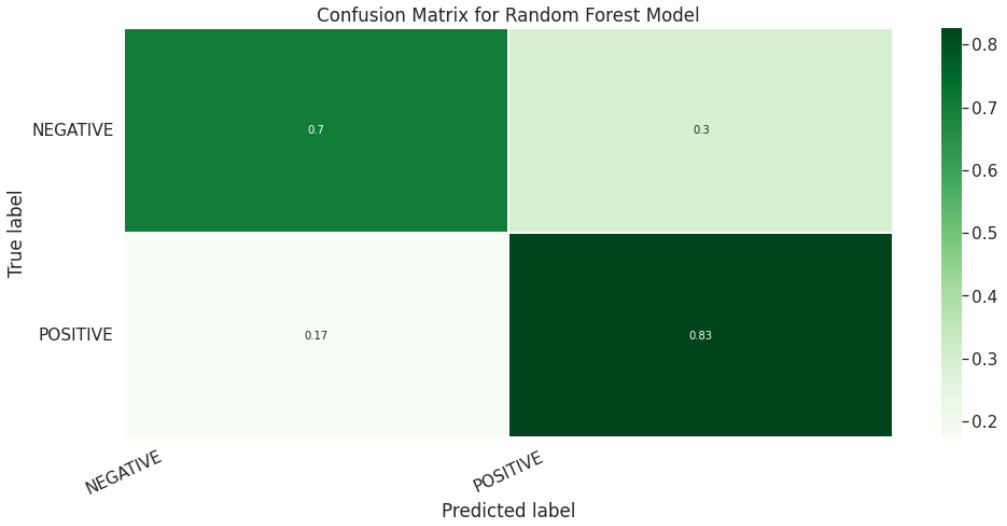


Figure 14 - CM GNB

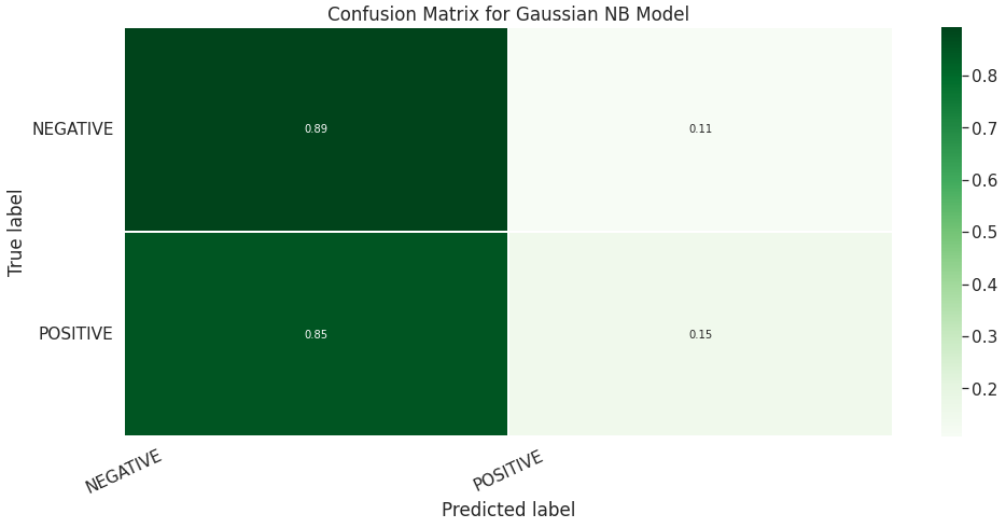


Figure 15 - CM LRMC

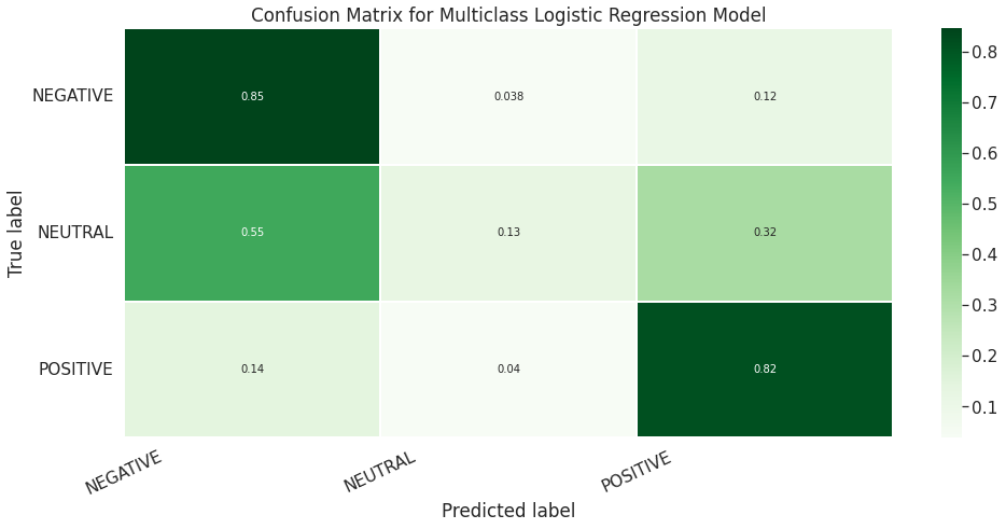
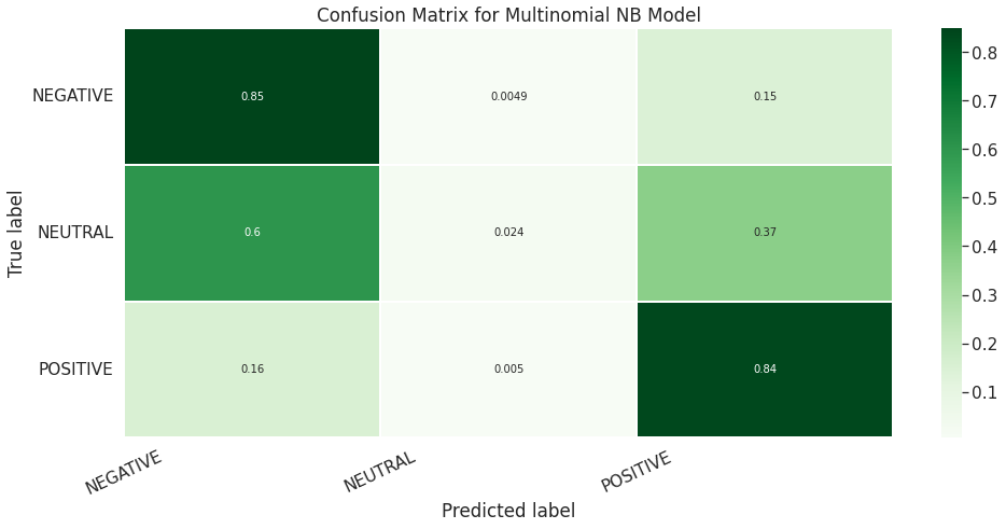
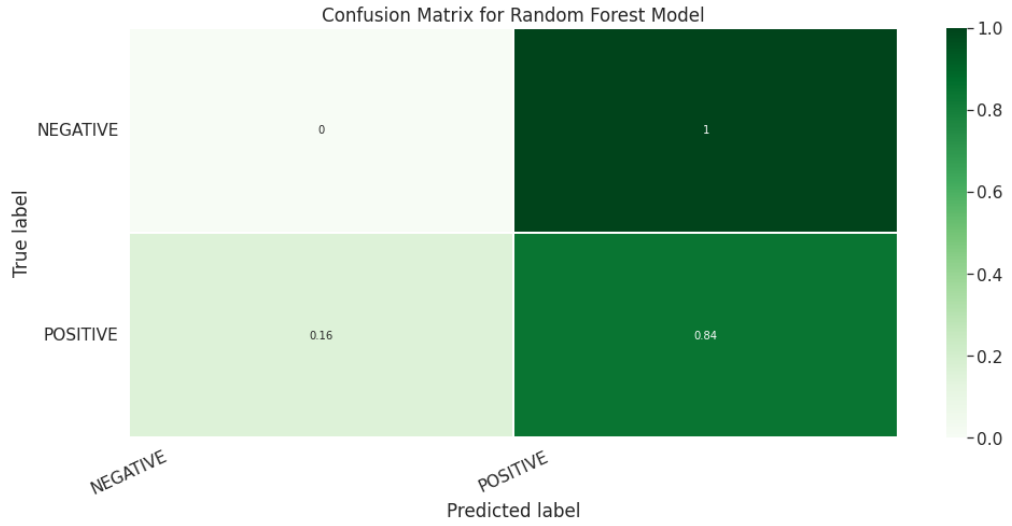


Figure 16 - CM MNNB



The model with the best performance was the Random Forest Classifier over the Bag of Words dataset. This model was selected to evaluate the performance against new data. The model was loaded from a file, and 20 reviews were used to test it after applying the previous text preprocessing steps. The accuracy obtained over this new data reached 80.0%.

Figure 17 - CM RFC Test Data



Conclusion

The Random Forest Classification model had even performance in classifying true positive and true negative labels with 83% and 70%, respectively. In the case of Gaussian Naive Based, the true positive label was not precise, reaching only 15%. That is why the performance of this model was the worst, meaning the model classifies almost all reviews as negative. In the case of multiclass models have nearly the same performance, but looking at the confusion matrices of both, the worst accuracy was about neutral labels, and it is probably due to the fewer number of neutral reviews compared to positives and negatives (see Figure 7 - Sentimental Trends).
So, the model that performed better was Random Forest Classifier and obtained 80% accuracy on test data. Still, additional tests with more negative reviews should be done, but in general terms, it behaves similarly to the training data.

References

Amazon. (n.d.). Amazon Customer Reviews Dataset. [Data set] https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Wireless_v1_00.tsv.gz.