**IA 645-Data Analytics for Cybersecurity (52792)**

Final Project

**Sentiment Analysis**

**A picture containing drawing, clock

Description automatically generated**

Submitted To: **Dr. Samir Tout**

Submitted By: **Ethan Smith (E01564716) &  
 Chahak Mittal (E02266200)**

**A picture containing object, clock

Description automatically generated**

**Introduction**

In today’s technology-aged world, organizations, individuals and other enterprises are facing deleterious wraths of cyber activities. To prevent the banefulness from this chaos, may it be material or non-material, anticipation of such events is a notion which should be adopted. This could be a next-to-impossible task due to the obscurity, incertitude and dearth of network data collected in the organizations. Therefore, we adopt a technique called Sentiment analysis which espies the repugnance of sentiments as positive, negative or neutral from a text provided. It excerpts the attitude of the writer about an agenda. This approach was originally developed for the department of linguistics and psychology, but it is now being related and tested to social network comments, news, reviews and dark web forums etc.

A pinnacle negative comment about a company or an organization might be a chance to target it for a cyber-attack. Due to the vast population and data on social networking sites these days, a basis of truthful dataset is difficult to find. Also, the extraction of sturdy and compelling features from the noisy and irrelevant pool of data is quite a task.

In this project, we plan to use sentiment analysis and data mining techniques on some original tweets to predict whether the sentiments behind those could be the reason a company might face a cyber-attack.

**Prior Work:**

To accomplish this task, we searched for a bunch of papers related to sentiment analysis and its application in the field of cybersecurity. We as a team divided our roles and worked collaboratively and efficiently to achieve the desired results.

Upon having enough information to begin with the project we decided to study the attitude and behaviour of people through their tweets about a company. We collected some tweets for two different organizations from twitter.

We kept weekly project team meetings and discussed the progress of every member and the next to-do lists. We even learnt the basic hands-on implementation of Python in data analytics and machine learning.

For this project we used Jupyter notebook, a platform available in the anaconda package to write the python codes. This involved a lot of practical applications, calculations and methodologies. In the beginning it was challenging with the errors in code but eventually we tried, learnt and finally succeeded with the desired output.

**Problem Statement:**

Our project aims at doing a sentiment analysis on the datasets collected for Apple Inc. and an airline company.

**Methodology:**

**Topic**

The objective of this experiment is to identify a machine learning model capable of predicting whether a set of short documents expresses a positive or negative sentiment toward an organization. Sentiment analysis on short texts is becoming more important as the wealth of information available in social media posts grows. For this reason, our model is trained and tested on two datasets of tweets that are a maximum of 240 characters in length. The first dataset is a labelled collection of tweets directed toward the Apple Corporation and the second is a labelled collection of tweets directed toward a variety of US airlines. Two different models, naïve bayes and a logarithmic regression, are trained and tested on each dataset with the objective of labelling according to a binary sentiment label and their performance compared.

Even short documents like tweets are ripe for analysis and can contain valuable insight into whether an organization is likely to be targeted by a cyberattack, as public opinion has proven to be a reliable barometer for what kind of companies and governments eventually draw the ire of hackers and activists. Indeed, the comparative ease by which a Twitter user can draft their reaction to any happening and the widespread access to the platform make it a much more reliable tool for gauging public opinion than newspaper articles, research papers, or even blogs. While granular sentiment analysis capable of identifying key topics, themes, and even implications present in the wider stream social media is useful for advertisers, counter-terrorism experts, and market researchers, cybersecurity experts are mostly concerned with the overall level of vitriol directed toward an organization. It is for this reason that the models evaluated in this experiment focus on identifying the overall proportion of ‘negative’ sentiment rather than identify certain topics or themes.

**Data**

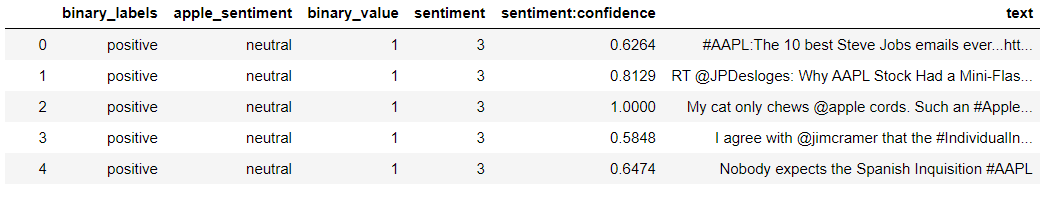
Two core datasets are used in this project. The first is a dataset produced by the crowdsourcing website Crowdflower that was hosted on the dataset repository website data.world under the name Apple Twitter Sentiment ([data.world](https://data.world/crowdflower/apple-twitter-sentiment%20), 2014). This dataset is henceforth referred to as “Apple” or “the Apple dataset.”

The dataset consists of the text 3,886 Tweets from 2014 that discuss the Apple Corporation, so defined by the fact that each included tweet uses the hashtag #AAPL or tagged the company with @apple. Via crowdsourcing, each tweet in the dataset includes a sentiment label of either 1, 3, or 5 that corresponds with how the human reviewing the tweet would classify the sentiment in the tweet: positive (5), neutral (3), or negative (1). Furthermore, each human reviewer gave a score about how confident they were in assigning a sentiment to the tweet. The number of trusted human judgements on each tweet and the associated sentiment confidence scores are also averaged and included in the dataset. This information is not utilized in this analysis, but should be considered in future work. Tweets that mention @apple or use the hashtag #AAPL that are not relevant are also labeled.

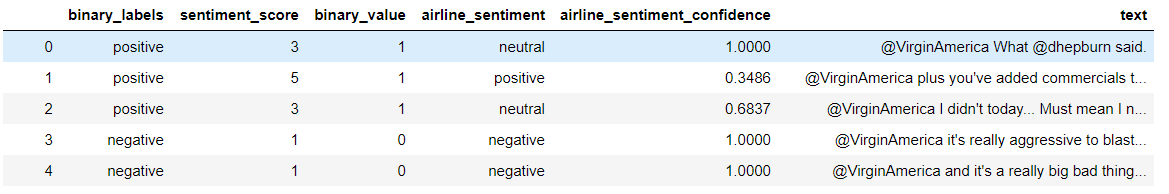
The other dataset used is a similar Crowdflower dataset outside on dataset repository website Kaggle named Twitter US Airline Sentiment ([Kaggle](https://www.kaggle.com/crowdflower/twitter-airline-sentiment), 2015), henceforth referred to as “Airline” or “the Airline dataset.” This dataset is considerably larger, containing 14,641 tweets directed toward a variety of major American airlines (i.e. the airline was tagged with an @), including Virgin America, Delta, Southwest, and United Airlines. Each tweet is labeled with its associated airline, so the data is sub settable by company, but the entire dataset is considered in this analysis. Like the Apple Dataset, it includes human-created sentiment labels of “positive”, “neutral”, and “negative” in string format instead of an integer code. Additionally, irrelevant tweets are labeled, making them easy to exclude, and a sentiment confidence is also included, although our models do not make use of this valuable information. It also includes more information about the tweeter, but this information is not relevant to the analysis in this experiment.

To get a general idea of the data that this experiment is working with, an Exploratory Data Analysis is useful. Specifically, it useful to know and visualization the breakdown of tweets that are labeled positive, negative, and neutral in each dataset, the exact attributes of the data that are utilized in the model, the most common words in subsets of the data, and keywords that give a clue to important themes that can be used to get an idea of how severe any negative opinion is. Each dataset will be analyzed in turn, beginning with the Apple dataset.

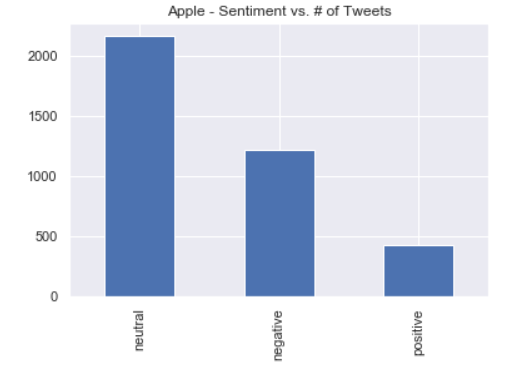
A subset of the data used in the analysis from the Apple dataset is presented below.



The columns binary\_labels, apple\_sentiment, and binary\_value were added by the authors for purposes of EDA. Binary\_value was produced by defining a function to take the 3 multi-class labels ‘positive’, ‘neutral’, and ‘negative’ and binarize them so that that ‘positive’ and ‘neutral’ tags were both given the integer tag 1 (positive) and negative tweets tagged with (0). Neutral tweets were coded as positive for two reasons. Firstly, from the perspective of a brand or corporation, any discussion about its organization or products is good as long as it isn’t actively hostile. Secondly, and more important to cybersecurity, our models are designed with the core objective of evaluating the extent to which public opinion is actively hostile, and therefore it is most useful to isolate the most negative tweets. Binary\_labels simply gives text labels to the binary\_values. A similar process was carried out on the Airline dataset, a subset of which is presented below.



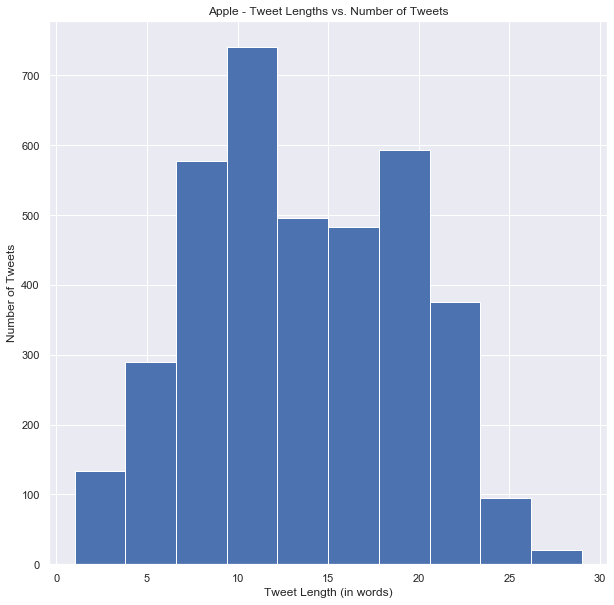
More interesting is the breakdown of positive, negative, and neutral tweets in each dataset. A bar graph plotting the sentiment against the number of tweets in the Apple dataset is presented below.



The dataset mostly contains tweets that human reviewers thought were neutral, generally consisting of memes, jokes, and general conversation about Apple and its products. However, between tweets that had a clear leaning in one direction or the other, there were considerably more negative tweets than positive tweets. The exact count works out to 2,162 neutral tweets, 1,219 negative tweets, and 423 positive tweets in the Apple dataset. For the purposes of the models built in this experiment, neutral and positive are both combined under the label “positive,” changing the graph.



When neutral tweets are counted are positive, the “positive” tweets outstrip the negative ones by 2,285 positive to 1,219 negative. A potential danger of doing sentiment analysis on tweets is that the length and complexity of each tweet varies, so that the model learns an unequal amount of each one. For that reason, a distribution of how many “long” tweets vs. “short” tweets exist in the dataset is valuable. This distribution was obtained by removing all punctuation from each tweet, then using the RegexpTokenizer from the nltk package to separate each tweet into a list of tokens and getting the length of this list to serve as a word count ([Jansma](https://harrisonjansma.com/apple" \l "Symbols-to-be-removed), 2018).

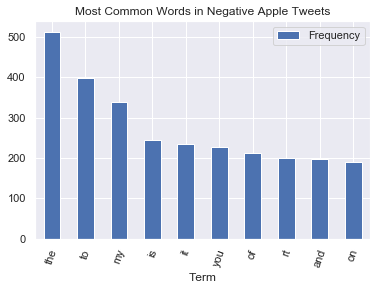


At first glance, it immediately apparent that tweet length in the Apple dataset is roughly normally distributed, which bodes well for a reliable sentiment analysis. There are relatively few outliers of extremely long or short tweets. However, there does seem to be a slight bias against tweets of the median length, with tweets slightly longer and shorter than the median being more common.

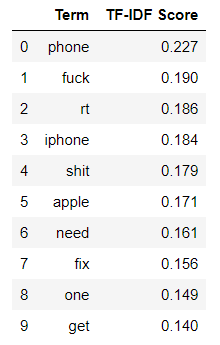
While these high-level counts are interesting, it is also useful to take a quick peek at what these tweets are talking about. The CountVectorizer method from the sklearn package is used to produce create a bag of words labelled with how often they appear in the corpus of *only the negative* Apple tweets. This is returned as a list, but with some manipulation can be converted to a dataframe that can be sorted and subset to return the most common words in tweets in the Apple database that were labelled as negative. These words and their frequencies are reported in the table and graph below.

**Apple - Most Common Terms and Frequencies in Negative Tweets**

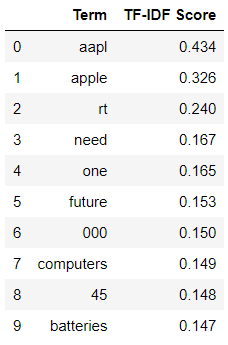
|  |  |
| --- | --- |
| **Term** | **Frequency** |
| the | 513 |
| to | 398 |
| my | 338 |
| is | 245 |
| it | 234 |
| you | 227 |
| of | 200 |
| and | 198 |
| on | 190 |



While this method does accurately return the top 10 most common words, they do not provide very much useful information. These words are simply common in general, and do not give any insight into the character of this corpus. To get a list of keywords that are more likely to be relevant, we used the CountVectorizer and TfidfTransformer methods to get a list of 1o keywords for each sentiment label (positive or negative) in each dataset ([Ganesan](https://kavita-ganesan.com/extracting-keywords-from-text-tfidf/#.XyeoJChKhPZ)). These keywords were extracted by filtering out all “stop words” (i.e. common words like the ones generated above), excluding all words that appear in 85% of the tweets in the Apple dataset, and sorting by ID-IDF score. The list of stop words was retrieved from the nltk package. The TF-IDF score measures term frequency inverse document frequency. It is calculated by dividing the Term Frequency (how often a term occurs in a document divided by the total number of terms in the document) by the Inverse Document Frequency (how important a term, measured by taking the natural logarithm of the total number of documents divided by the number of documents that include the term in question) ([tfidf.com](http://www.tfidf.com/#:~:text=Tf%2Didf%20stands%20for%20term,in%20a%20collection%20or%20corpus.)).

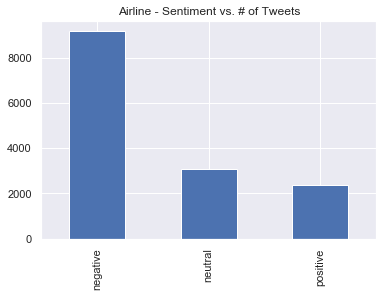
ID-IDF is designed to highlight only *important* frequently occurring words and suppress common but unimportant words like ‘and.’ The top 10 most important words by TF-IDF score that also do not appear in 85% of negative Apple tweets are as follows.

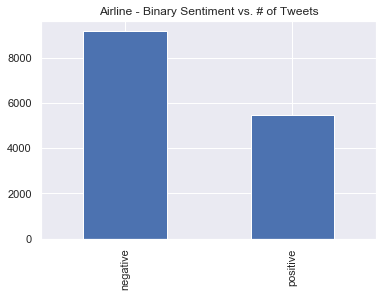
These results are considerably more interesting. For one thing, they appear to indicate that a lot of the negative tweets about Apple are related to the iPhone, and also supports the obvious conjecture that profanity may be a useful proxy for negative sentiment. Monitoring for the incidence of these keywords could be a lightweight solution for measuring overall sentiment changes over time. Now we turn to keywords in the positive Apple tweet corpus extracted with the same method.



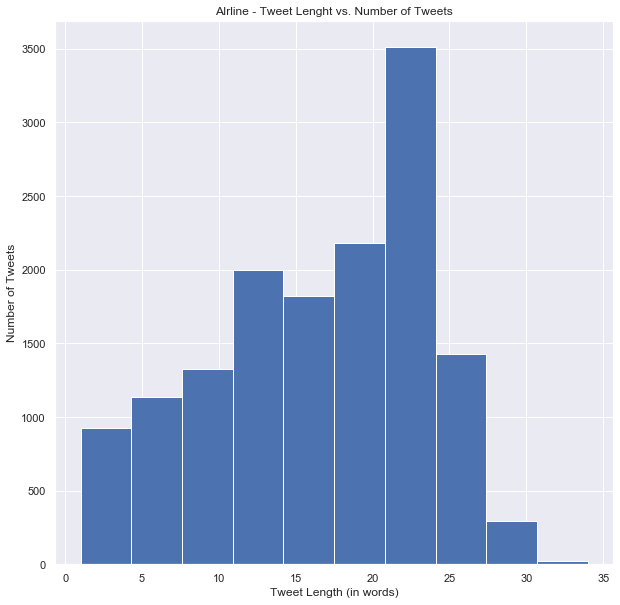
While more analysis is needed to draw solid conclusions, a quick look at these keywords might indicate that positive sentiment toward Apple includes themes of futurism, consumer enthusiasm (“need one”), and positive opinions toward Apple computers, as opposed to the iPhones that appeared to draw more negative sentiment.

Next, an EDA of the Airline dataset is presented. All data is retrieved with the same techniques used on the Apple dataset.

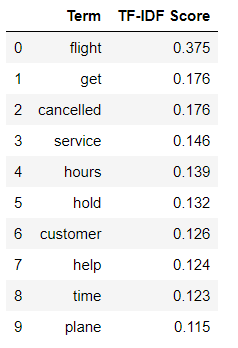


 The Airline dataset is overwhelmingly compromised of negative tweets, dwarfing the neutral and negative tweets. There are 9,178 negative tweets to 3,099 and 2,2363 neutral and positive tweets respectively. This alone may suggest that airlines are a generally more unpopular class of company than tech companies like Apple, and may be evidence that airlines should be especially concerned about potential cyberattacks.

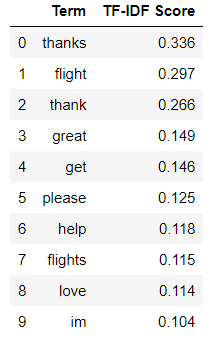
Combining the neutral and positive tweets balanced the picture somewhat, but negative tweets still outweigh positively classified tweets by 9,178 to 5,462 in the Airline dataset.



The Airline dataset is slightly positively skewed, with the majority of tweets being between 5 and 20 words and comparatively few longer than 25 words. The single highest word count range incidence was 20-25 words, with approximately 3,500 tweets containing that number of words. Ultimately, this is not an overly concerning distribution. There are not enough tweets with extremely low or high word counts to skew the sentiment analysis.

Because simply retrieving the most common words was not fruitful, that part of the analysis was not repeated on the Airline data. However, the top 10 keywords by ID-IDF score for negatively classified tweets in the Airline data are presented below.

This may be the most interesting result yet. A relatively clear picture of why travellers get upset with their airlines is presented. They don’t like it when their “flight” “get”s “cancelled”, or when the “customer” “service” is poor, and when “hours” of their “time” is wasted being put on “hold.” The top keywords in the positive Airline tweets don’t tell a story that is quite so complete.



These keywords seemed to suggest that a lot of the positive sentiment directed toward the airlines are people tweeting their appreciation for a great flight, and thanking the flight staff and/or company.

**Approach**

Two models to predict the sentiment of tweets are specified, one built with a Naïve Bayes classifier (the Naïve Bayes Model, or Model 1) and the other built using a logistic regression (the Regression Model, or Model 2). Naïve Bayes assumes that all attributes of a dataset are independent (which is a strong assumption), and uses the attributes of an instance I to calculate the highest posterior probability of a given class J being associated with those attributes, so that class J is predicted for class I. Traditionally, Naïve Bayes performs best with non-normally distributed multi-class data ([Ray](https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/), 2017).



Naïve Bayes equation ([Ray](https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/), 2017)

Logistic Regression on the other hand assumes a normal distribution, but is generally a good fit for binary classification problems ([Stojiljkovic](https://realpython.com/logistic-regression-python), 2020) We regress the class labels on the TF-IFD vectorized corpus to generate classification predictions for each of the datasets.

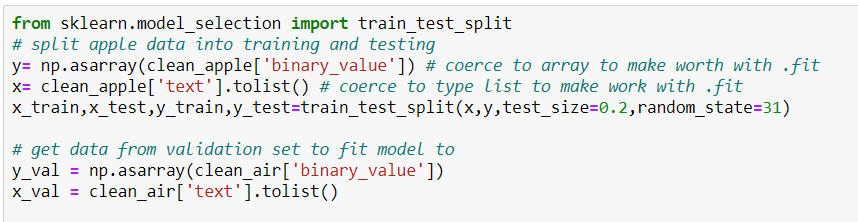
Each model is trained, tested, and validated on each dataset. That is, the Apple dataset is first used to train and test the Naïve Bayes Model, which is then validated using the Airline dataset. Then, the roles of the datasets switch so that Model 1 is trained and tested on the Airline dataset and validated using the Apple dataset. The process is then repeated with Model 2, and the performance of the model in each permutation is compared using confusion matrices, accuracy, precision, and recall. This approach is taken to assist with diagnosing issues with poorly performing models, and parse out whether poor model performance is due to lack of data or overfitting.

**Experimental Setup**

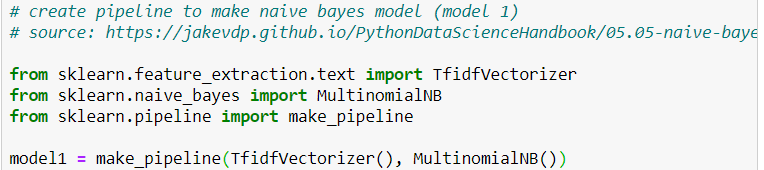
To avoid confusing our models with extraneous punctation and symbols, all text was converted to lowercase and standardized such that all remained was a list of words by defining a function to replace all punctuation with nothing, deleting it ([Jansma](https://harrisonjansma.com/apple" \l "Symbols-to-be-removed), 2018).

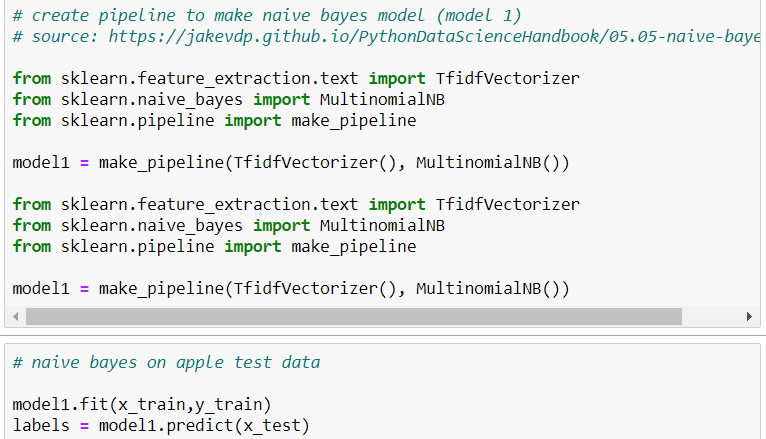


Then, sklearn’s train\_test\_split method is called to segregate the Apple dataset into testing and training data, and the relevant Airlines data for validation is retrieved and coerced to the array and list datatypes needed for the .fit function to work properly. The process is repeated so that the Apple dataset is treated as the validation data and the Airline data is split into training and testing. The segregation of Apple and the coercion of Airline is shown below.



Finally, sklearn’s TfidVectorize and canned naïve bayes algorithm MultinomialNB are combined using sklearn’s make\_pipleline to produce model1, which is first fitted to the Apple dataset training data and then the Airline dataset training data ([VanderPlas](https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html)). This is repeated, replacing MultinomialNB with LogisticRegressionCV(). The process for the Apple data and Model 1 is shown below.





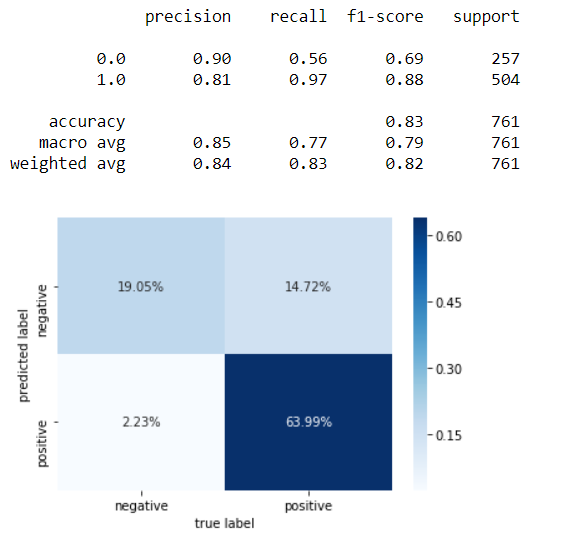
**Experimental Results & Analysis**

The experimental design described in the previous section produces 8 distinct results, summarized in the table below. The performance of each model on the test data (from the same dataset on which it is trained) is compared to its performance on the validation data to detect overfitting, and the role of testing and validation is switched for repeatability, experimental rigor, and verification of whether sample size appears to be playing a role.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 1 | Naïve Bayes | Apple | Airline | Test |
| 2 | Naïve Bayes | Apple | Airline | Validation |
| 3 | Naïve Bayes | Airline | Apple | Test |
| 4 | Naïve Bayes | Airline | Apple | Validation |
| 5 | Log Regression | Apple | Airline | Test |
| 6 | Log Regression | Apple | Airline | Validation |
| 7 | Log Regression | Airline | Apple | Test |
| 8 | Log Regression | Airline | Apple | Validation |

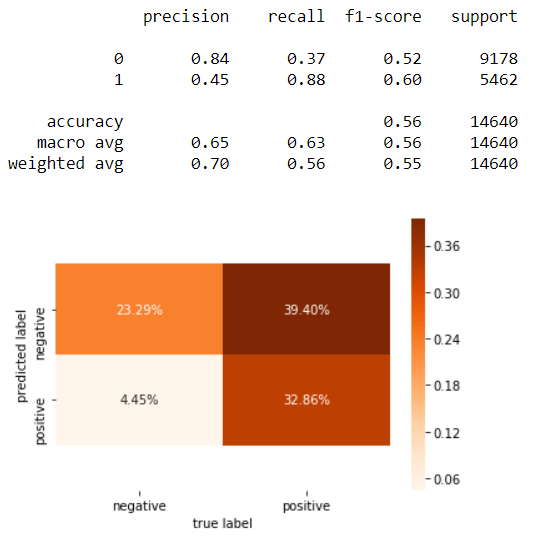
Each “result” is presented in the order indicated in the table above, and each model’s performance in each permutation will be reported with a confusion matrix, precision, recall, and accuracy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 1 | Naïve Bayes | Apple | Airline | Test |



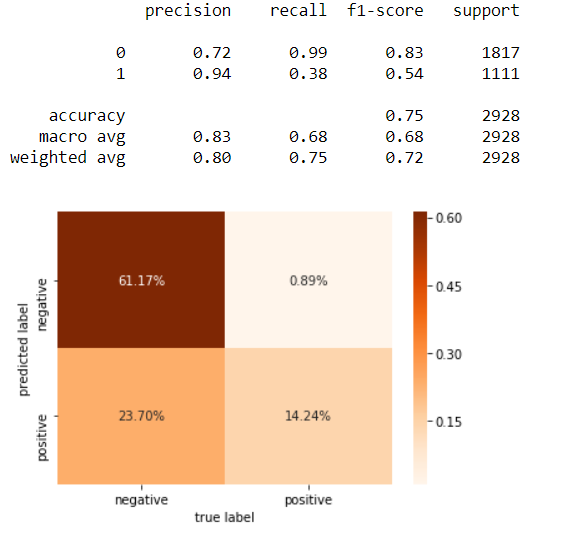
The Naïve Bayes classifier performed rather well when trained and tested on the Apple dataset. It was able to accurately classify 83% of the tweets. However, it did have a tendency to classify positive tweets as negative, which could lead an organization to overrate its risk of a cyberattack and spend unnecessary resources on securing its systems. Ultimately, the real test of the model 1 is not how it performs on the data it was trained on, but the validation data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 2 | Naïve Bayes | Apple | Airline | Validation |



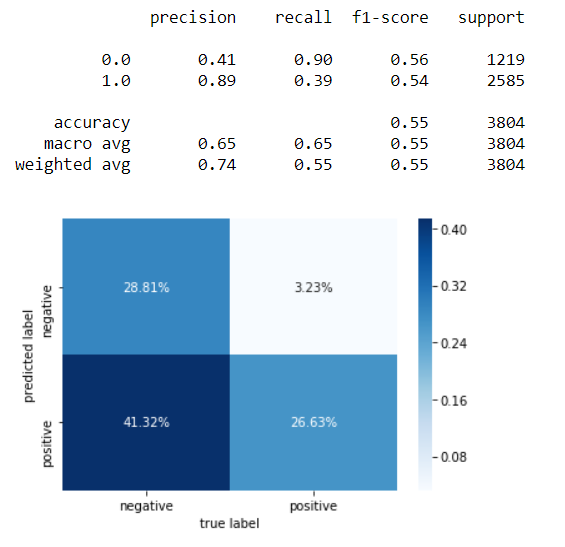
When applied to the validation dataset, in this case Airline, model 1 performs dramatically worse, which is clear evidence of overfitting. The model was only able to accurately classify 56% of the tweets. Even worse, it actually produced more false “negatives” (as in a negative classification) than it accurately identified negative tweets. Since that it is our primary concern in a cybersecurity application, it is safe to the say that the Naïve Bayes classifier does not perform adequately. While overfitting is the most likely explanation for the models poor performance, it may also be the case that the model was not able to learn enough from the relatively small Apple dataset, and it may fare better if tested on the Airline dataset instead. This scenario is examined next.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 3 | Naïve Bayes | Airline | Apple | Test |



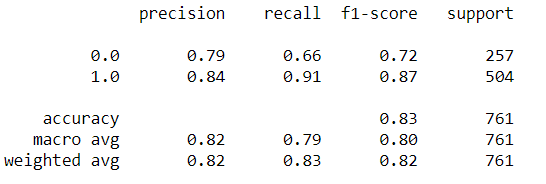
When the Naïve Bayes classifier is trained and tested on the Airlines data, it performs moderately well, but worse than when it was tested and trained on the Apple data. It accurately classified 75% of the tweets, but it had a tendency to classify negative tweets as positive. However, it almost never classified positive tweets as negative, opposite of the result in the Apple dataset. This is likely due to the fact that Airline has more negative tweets and Apple has more positive tweets.

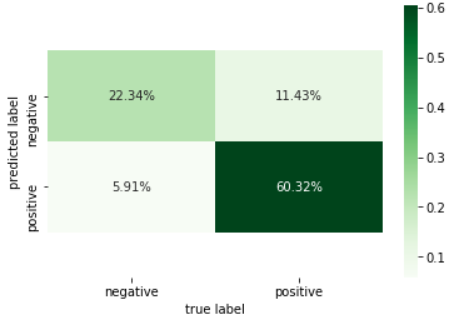
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 4 | Naïve Bayes | Airline | Apple | Validation |



When applying the Naïve Bayes classifier that was trained on the Airline dataset to the Apple dataset, we see the other side of the same coin. The accuracy is almost exactly equally poor, 55%, with a heavy tendency to produce false “positive” classifications. This is irrefutable evidence of overfitting, and points to Naïve Bayes being a poor model for classification in this application. Perhaps Logistic Regression will fare better.

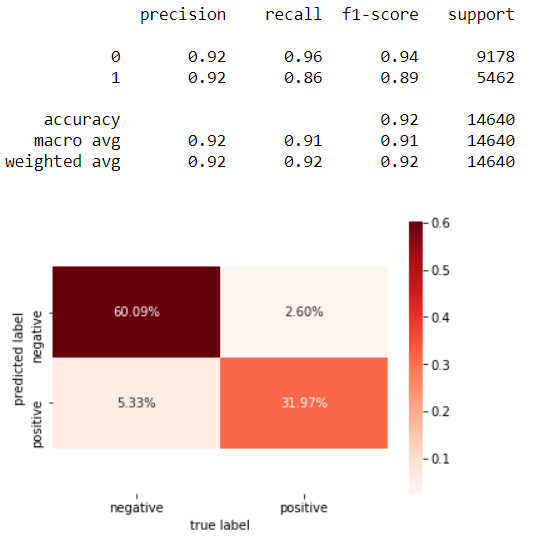
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 5 | Log Regression | Apple | Airline | Test |





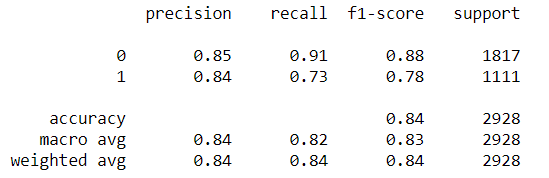
Coincidentally, the regression model was exactly as accurate as the Naïve Bayes when trained and tested on the Apple dataset, correctly classifying 83% of the tweets. It tended to err on the side of assigning false “negative” classifications, which is better than missing negatives and calling them positives for our application. Overall, it performed relatively well, but may still be overfit. Applying the trained model to the validation data will reveal whether this is the case.

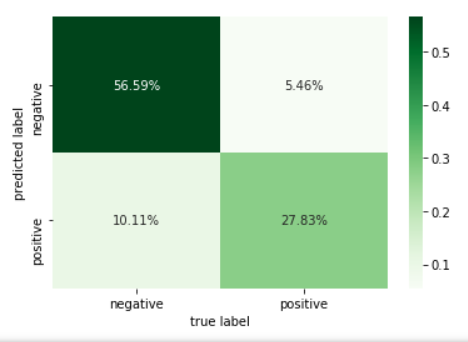
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 6 | Log Regression | Apple | Airline | Validation |



This time, we get an astounding result. Not only does the Logistic Regression outperform Naïve Bayes when applied to the validation data, it actually does better than it did on the test data! This is an encouraging but unusual result, so switching the roles of the datasets can help reveal if the 92% accuracy is a fluke.

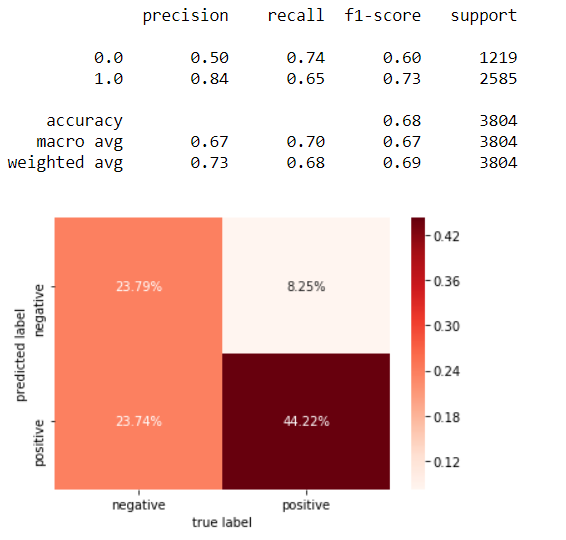
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 7 | Log Regression | Airline | Apple | Test |





Again we see the accuracy that we have come to expect from applying our models to the test data, 84%. False “positive” classifications are somewhat more common than false “negatives,” but the model performs relatively well on the test data. Now it is time to see if we can replicate the impressive results of the Log Regression’s first outing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result #** | **Model** | **Test/Train Data** | **Validation Data** | **Result Reported** |
| 8 | Log Regression | Airline | Apple | Validation |



When applying the Log Regression model to the (Apple) validation data, we get a very mixed result. The model unquestionably performs better than Naïve Bayes, scoring a 68% classification accuracy over 55-56%, but has a heavy bias toward predicting false “positive” classifications, which could cause an organization to substantially underrate their risk of cyberattack. Nonetheless, there are more true positives than false positives and more true negatives than false negatives.

**Conclusion**

The ability of two models, the naïve bayes classifier and the logistic regression, to accurately predict the overall sentiment of a tweet directed by an organization was tested and compared. The naïve bayes classifier was shown to severely overfit the data when trained on each of the datasets used in the analysis and did not appear able to generate significant results without extensive modification.

The logistic regression model performed very well on validation data, correctly classifying 92% of tweets, suggesting that it may be well suited to this application. However, this result was not very robust to switching the datasets used for training/testing and validation, so further testing and modification is needed to create a truly reliable predictor. The basic logistic regression model tested in this experiment could be dangerous if used in its imagined cybersecurity application, measuring the level of vitriol toward a company. If this vitriol is underrated, as this experiment suggests the model may lead to, an organization may decide not to adequately invest in protecting against a cyberattack.

**Github Repository Overview**

The Github repository on which this project is housed contains two ipynb jupyter notebooks and four .csv data files. The first notebook, Cleaning Data and EDA.ipynb, contains the code used to drop unnecessary columns, add labels and binarize the data, plot the tweet lengths, plot the frequency of each tweet class, and generate lists of common words and keywords. It uses the data from ‘AirlineTweets.csv’ and ‘Apple-Twitter-Sentiment.csv.’ The clean data produced in Cleaning Data and EDA.ipynb converted to comma separated values format (as a string) in Spyder and then manually fed into Excel and converted to clean\_air.csv and clean\_apple.csv.

These two datasets are used in Testing Models.ipynb, which contains the code used to fit each of the models and produce the confusion matrices and classification reports presented in this report.

References

1. <https://www.kaggle.com/crowdflower/twitter-airline-sentiment>
2. <https://data.world/crowdflower/apple-twitter-sentiment>
3. <https://harrisonjansma.com/apple#Symbols-to-be-removed>
4. <https://kavita-ganesan.com/extracting-keywords-from-text-tfidf/#.XyeoJChKhPZ>
5. <http://www.tfidf.com/#:~:text=Tf%2Didf%20stands%20for%20term,in%20a%20collection%20or%20corpus>.
6. <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>
7. <https://realpython.com/logistic-regression-python>
8. <https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html>