Twitter Censorship Prediction Model

Cormac Madden, Liam Byrne, Nisarg Kalyani

December 3, 2022

1 Introduction

The topic of censorship and free speech on the internet has flooded the news in recent years and rules regarding what should and shouldn't be censored is often hidden behind company policy and riddled with ambiguity. Our objective is to create a machine learning model that can predict whether a certain tweet will likely be censored on twitter. This would be a useful tool to determine what tweets twitter determines as acceptable, and to enable users to gauge the likelihood that a particular tweet will be censored. The input to our algorithm are the text components from a Twitter Tweets. The text from tweets should be sufficient to predict the output, often Tweets get censored based on links that are included but accounting for this is outside the scope of our project. This project will ultimately examine how various machine learning models perform on the same input data-set of tweets.

2 Data-set and Features

2.1 Description

The Twitter API is the source of all the data for this project however, we it only returns tweets posted in the last 7 days. You need to apply for an enterprise account to gain access to the full Twitter archive. Instead we decided to use an archive called Twitter Stream Grab, which has stored 1% of all tweets from Twitter's API, each day of every month up until August of 2021. The data-set is in JSON format, and each JSON file contains around 8K tweets objects which include information such as the Tweet's text, the user, and most importantly what countries the tweet has been censored in. This is labelled as "withheld_in_countries" and the countries are stored in an array as follows: ["DE", "FR"]. There is no information available regarding Tweets that were outright blocked by Twitter such as overly explicit or graphic content, but the censorship by countries gives us an interesting insight into information that governments are trying to censor.

Once the data-set was downloaded we wrote a python script that goes through each of the sub-directories to the JSON files, and checks whether tweets are firstly withheld in any countries and secondly whether the tweet is in English. We decided to limit our data-set to English so it would be easier for us to debug. Any tweet that satisfied these conditions were then written to a new JSON file. The next step was to extract the necessary information from the tweet, for now this is just the text information, but at a later this is where we would like to experiment with separating out the censorship of different countries.

2.2 Mapping

To map the data we first created one large data-frame consisting of all the tweet text in the first column, and whether it was withheld in any country in the second column. Whether the tweet was censored was represented by a 0 or 1. Censored = 0, Non-Censored = 1.

The raw text data cannot be used as training input to our models as they expect numerical feature vectors. To extract features from the data we a bag of words model which includes used both tokenization and occurrence counting. To implement these techniques we used the "CountVectorizer" class provided by scikit-learn.

During the tokenizing process we removed all capital letters, all punctuation and extracted all words of at least two letters, which removes all words with only one letter such as "a" and "I". We preserved some context by also extracting 2-grams of words (Treating two adjacent words as one new word). Each remaining word is then given a unique integer index. We then perform occurrence counting which simply counts the number of occurrences of each word.

We removed the words that appeared too frequently or infrequently using the hyper-parameters min_diff and max_diff. The aim is to remove words that may be typos or filler words. We used cross-validation to determine the best values for each. Figure X shows the results of cross validation on both parameters. We decided to use a min_diff value of 2 and a max_diff value of 60 based on these results.

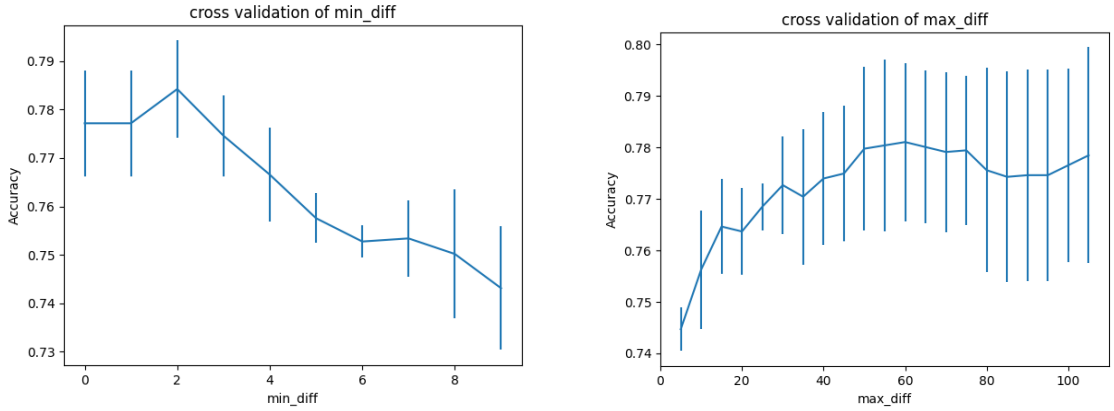

Figure 1: Cross validation of word frequency cutoff points.

2.3 Appropriateness of Features

To examine the appropriateness of the features being created by the models we created a "word cloud" where the significance of each feature is represented by its size in the image. See Figure 2.

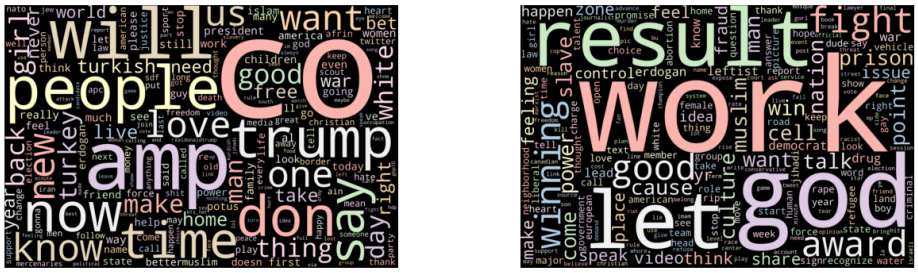

Figure 2: Word clouds of censored (left), uncensored (right)

3 Methods

3.1 Logistic Regression

Logistic regression is a generalised linear model used for classification by taking a set of input variables or features to output a binary output value. It works in a similar manner to linear regression in that it still computes the weighted sum of the input features xi and the y intercept term b, but it runs this term through an s shaped logistic function f to receive an outcome y that is compressed between 0 and 1. The logistic function is a function that gets closer to 1 as the input value increases above 0 and closer to 0 as the input value decreases below zero. As the model can produce a binary output it is suitable for classification as we can attribute a y value of 1 as belonging to the positive positive class, or in this case the censored class. L2 regularisation is enabled by default with SKlearn's implementation of Logistic regression. The hyper-parameter C inversely varies the level of regularisation on the model, with higher values of C resulting in less regularisation.

3.2 k Nearest Neighbours

KNN is an algorithm that can be used to classify data based on the new data's distance to existing data points. We used a KNN model to predict the likelihood that a given tweet will be censored on Twitter. However, we found that this type of model was quite ineffective at classifying our text. The KNN algorithm requires a large and diverse data-set to train the model. In the case of Twitter censorship, obtaining a sufficiently large data-set of censored tweets to train the model requires a lot more time and resources. Overall, while we were able to use a KNN model for predicting censorship on Twitter, we found that the other models we used were substantially more effective.

3.3 Multinomial naive Bayes

The Multinomial naive Bayes Classifier is a commonly used for text classification tasks, such as sentiment analysis or categorization. It is a classifier based on applying Bayes' theorem with strong (naive) independence assumptions between the features. This means it assumes that each feature is mutually exclusive when predicting the outcome. The Multinomial name refers to the occurrence counting part of the model. This classifier performed well on tweet censorship prediction, achieving higher accuracy results than the other two classifiers.

4 Results/Discussions

4.1 Inputs and outputs of each model

All of the models require vectorized numerical inputs as discussed in the Mapping Section 2.2. Logistic regression returns a positive or negative value that is determined by a logistic function, this is typically then bound to between 0 and 1. Binary Logistic regression is where we take the sign of the result and use it to classify our result as being one of two options. The Multinomial NB model returns a similar output but is based off several different Bayes' related algorithms. kNN returns the value of 0 or 1 depending on what training data the predicted outcome is "nearest" to.

4.2 Parameters and Hyper-Parameters

As can be seen in Figure 3,4 and 5, to find the optimal parameters for each of the 3 models, we plotted their relevant parameters against the "accuracy" metric. Accuracy is the ratio of correct predictions against total instances. It is a flawed metric in many circumstances, but here it allows us to use a single metric to see the impact of each parameter. To determine each hyper-parameter we use cross-validation 5 folds on three thousand sample tweets.

For Logistic Regression we iterated the C parameter from 0 through to 1.5 with a varying step. We thought a C value of 5 would be appropriate here as it results in a high accuracy, low standard deviation and doesn't unnecessarily over-fit the data.

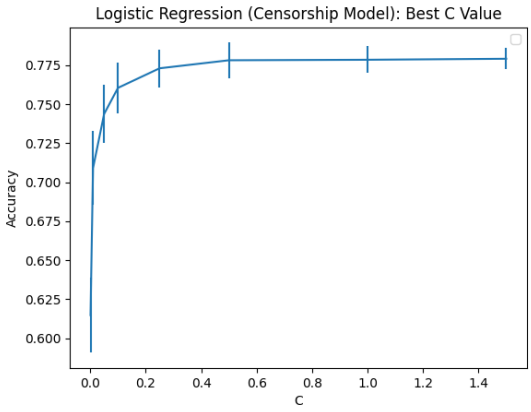

Figure 3: Graph of c values for Logistic Regression Model

Alpha is Multi-nominal NB's smoothing parameter. We iterated alpha from 0 through to 15 with a step of 1 and decided that 2 was the most suitable parameter as it resulted in the highest accuracy and lowest standard deviation.

For K Nearest Neighbours we iterated k from 0 through to 100 with a step of 5. The graph clearly shows that with a low k value the model was quite unstable and varied a lot with each fold. We thought a k value of around 20 would be a relatively good choice here, but it would still result in a very low accuracy of approximately 0.54.

The importance of using the optimal hyper-parameters for these models is particularly relevant when using data-sets with a large number of feature variables such as with text classification. It's difficult to know whether that data has been over-fit, but considering the limited input data, a closely fit model may not be so bad. We believe the hyper-parameters we have chosen are appropriate given the graphs produced by our cross-validation.

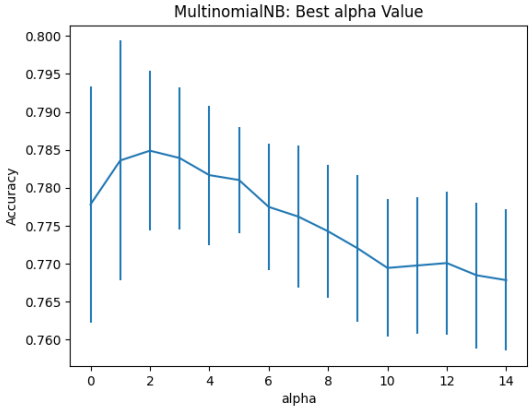

Figure 4: Graph of alpha values for Multinomial NB Model

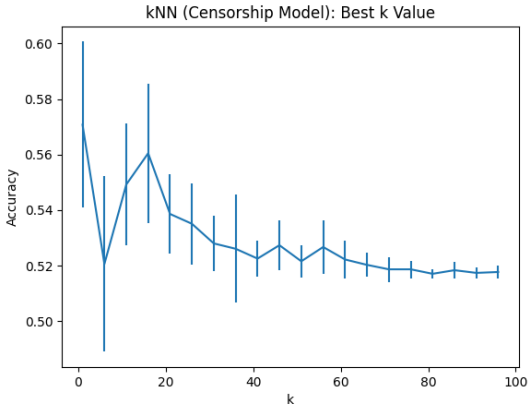

Figure 5: Graph of k values for k Nearest Neighbours

4.3 Optimization Algorithms

For the logistic regression model, we used gradient descent as our solver ("sag"). This produced marginally better results on the ROC curves when compared with Sklearns default L-BFGS algorithm. For the k Nearest Neighbours model we chose the KD tree algorithm as we were familiar with it and it performed well compared to the alternative brute force algorithm. The Multinomial NB model uses a combination of several optimization algorithms so that was left unchanged.

Logistic Regression			
	Precision	Recall	f1-score
Not-Censored	0.78	0.71	0.75
Censored	0.75	0.81	0.78
Average	0.77	0.76	0.76
Accuracy			0.76
Multi Nomial NB			
	Precision	Recall	f1-score
Not-Censored	0.77	0.80	0.79
Censored	0.80	0.77	0.79
Average	0.79	0.79	0.79
Accuracy			0.79
k Nearest Neighbours			
	Precision	Recall	f1-score
Not-Censored	0.67	0.36	0.47
Censored	0.57	0.83	0.68
Average	0.62	0.60	0.57
Accuracy			0.60
Dummy Classifier			
	Precision	Recall	f1-score
Not-Censored	0.00	0.00	0.00
Censored	0.51	1.00	0.68
Average	0.25	0.50	0.34
Accuracy			0.51

Table 1: A Table of the metrics from each classification model.

4.4 Primary Metrics

To measure the performance of each of the models we used the following metrics:

- Precision: true positives / (true positives + false positives)
- Recall: true positives / (true positives + false negatives)
- f1-score: 2 * (precision * recall) / (precision + recall)
- Accuracy: correct predictions / total predictions

For performance evaluation we separate our data into a test-train split of 0.2. This means 20% of our data is held out for testing.

4.5 Discussion

We found that the logisitc regression and multinomialNB models gave much better results than the KNN model. In particular, the multinomialNB model achieved a precision score of 0.79, a recall score of 0.79, and an F1 score of 0.79, while the logistic regression model achieved a precision score of 0.77, a recall score of 0.76, and an F1 score of 0.76. In comparison, the KNN model achieved a precision score of 0.62, a recall score of 0.60, and an F1 score of 0.57. The multinomialNB model was able to better identify the relevant features in the dataset and make accurate predictions based on those features. In contrast, the KNN model struggled to accurately predict censoring due to its reliance on distance measures to identify patterns in the data. Overall, these results indicate that the logistic regression and multinomialNB models are more effective for predicting censoring of tweets than KNN models.

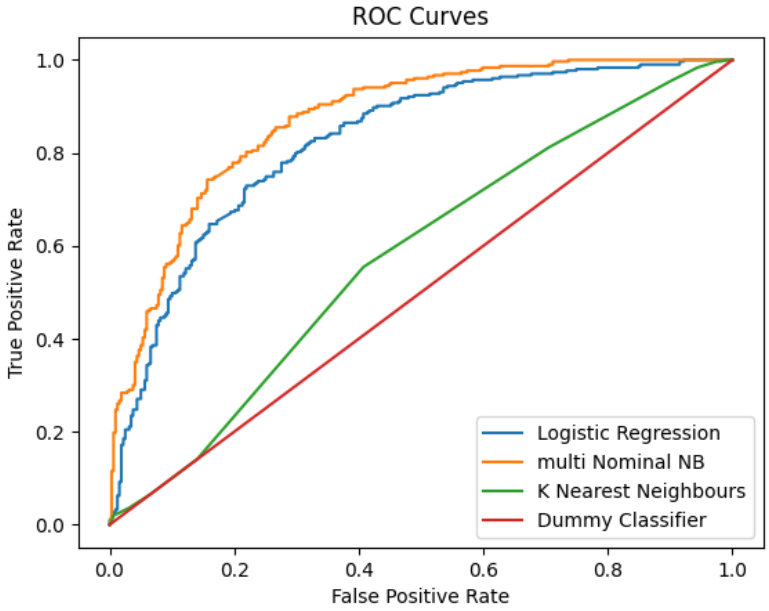

Figure 6: ROC Curves for Logistic Regression, multiNominalNB, kNN and a Dummy Classifier

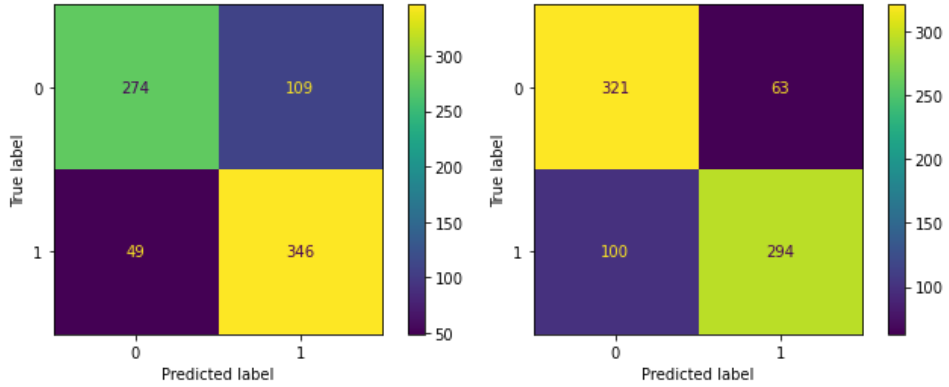

Figure 7: Confusion Matrix Logistic Regression (left) and Multinominal Naive Bayes (right)

5 Summary

Our Twitter Censorship Prediction Model mostly achieved the our objective of predicting the likelihood that a tweet will be censored on Twitter based on its text content. The final model we created is a Multinomial naive Bayes model, that uses the text components of tweets and whether the tweet has been censored by any country as inputs. The model uses the bag of words approach, with tokenization and occurrence counting, to extract features from the text data. The model preserves contextualized information in the tweet by extracting 2-grams of words. The model uses the hyper-parameter values we calculated of min_diff=2 and max_diff = 90 to remove words that appear to often or too rarely, and model uses the alpha value of 2 that we calculated using cross validation. We are satisfied with the performance of our model and are confident that with the addition of more training data our model would only improve further.

6 Contributions

Cormac:

- Code for scraping Twitter API,
- Code for filtering withheld English Tweets
- Code for creating figures for Hyper-Parameters/Cross-Validation
- Code for creating ROC Curves Figure
- Wrote the Data-set and Features section of the report
- Assisted in writing of other sections of the report

Liam:

- Code for creating Logistic Regression, kNN, multinomialNB classifiers
- Code for creating Word Clouds figures
- Code for creating Word Confusion Matricies Figure
- Wrote Methods section of the report
- Worked on Testing Google Sentiment Analysis API
- Worked on using custom Bag of Words Algorithm
- Assisted in writing of other sections of the report

Nisarg:

- Code for original Twitter API calls
- Worked on Testing the ML models
- Wrote Discussions section of the report
- Wrote Summary section of the report
- Assisted in writing of other sections of the report

7 GitHub

Link to GitHub: https://github.com/cormacmadden/Twitter_Censorship_Prediction.git