2020 Election Sentiment Analysis

Syracuse University

IST 718 Big Data Analytics

Group 9

Ryan Ondocin: rjondoci@syr.edu

Jenny Cao: jcao21@syr.edu

Shripad Laddha: shladdha@syr.edu

Prathamesh Pradip Datar: pdatar@syr.edu

Table of Contents

1. Abstract

Social media has played an integral role in presidential elections since 2016; with the advent of Cambridge Analytica and nuanced psycho-political campaigning tactics. Twitter can be seen as a platform for candidates and users to gain substantial outreach to showcase their views to the world. Thus it is important to analyze and understand the role Twitter can play in gauging sentiment surrounding hot button issues that voters use in deciding which candidate is fit for leading the United States for the next four to eight years.

This project aimed to perform sentiment analysis on tweets about the 2020 US Presidential election for candidates Joe Biden and Donald Trump. Our objective was to unearth contrasting information in one week before and after the election results were announced. Using Principal Component Analysis (PCA) and KMeans clustering, we tried to analyze the presence of polarity or communities in the dataset. Furthermore, we performed a hashtag distribution analysis on our clusters to validate and inspect user sentiment. We applied Logistic Regression and Random Forest Classification to generate the top words, in terms of feature importance, responsible for predicting tweet sentiment and we looked for any change in the top words between these two time periods.

We also analyzed patterns of outlier users; most of which belonged to either news channels providing constant election updates, or provocative users who were supporting/slandering the two candidates. From the results of KMeans Clustering and PCA, we saw an increase in the optimal number of clusters from 2 to 3 after the election's results were announced. We feel this can sufficiently prove that before the election, the public was focused on divisive issues that characterized users into one of two political camps. However, after the election, many new topics emerged into the picture suggesting a period of reorientation in American discourse for issues such as gun reform, voter fraud, and racial inequality. Hashtag Analysis allowed us to analyze the discrepancies in the distribution of hashtags within our clusters. We discerned that Biden was popular on Twitter both before and after the election, which supports the idea that Twitter is a left-leaning platform according to the sample's insights. Based on our modeling results, we identified words such as "corrupt, stupid, idiots, win, great", etc. that had the highest feature importance for predicting the sentimental label of tweets. We found substantial congruity in the most important words for predicting sentiment before and after the election, which can attest to our model's ability to generalize. One of the underlying naive assumptions we made was to ignore the presence of sarcasm. When a user writes #Biden2020, we assume they are in favor of Biden but there is also a

slight possibility of Trump supporters using the same hashtag with undertones of sarcasm. Therefore, in the future, we hope to implement more robust models for detecting sarcasm in the hopes of improving the model's inference abilities. We used TextBlob, a third-party tool, to generate sentiment to train our machine learning algorithms. Therefore, the results obtained from our machine learning models do have a distinct possibility of being misrepresented as the generated labels cannot be considered as 'ground truth'. Hand generated labels in this context would even fall short of this assumption because there is a degree of subjectivity that goes into the interpretation of tweet sentiment. In the future, we plan on employing several methods to accurately generate sentiment labels. Important words generated by the machine learning models cannot portray an objective picture of the sentiment surrounding this election. Therefore, we will account for bigrams and trigrams in future work as well as different vectorization techniques to avoid losing contextual information surrounding tweets, allowing for better model inference.

2. Data Collection

Tweets distinguish themselves from regular textual data given their 140-character limit and ubiquitous use of slang and emojis. Furthermore, the use of embedded metadata such as hyperlinks, GIFS, and images enables users to devour a larger volume of content much more quickly than ever before. We used tweepy to extract the election data from Twitter. It scrapes the Twitter data based on certain hashtags that we provided such as "#TRUMP2020", "#BIDENHARRIS", "#ELECTION2020" along with many more variants. For our analysis, we scraped the data twice: first by taking data from one week before the election and second by using the same script for obtaining tweets from one week after the election results were announced. Before election data consists of 97,200 rows and 7 columns which include: user, id, location, date, favorites count (How many times a tweet has been Favorited), text (tweet content), and a retweet flag which will be either true or false. After election data consists of 45,394 rows.

Tweepy's standard API only allows users to fetch tweets within the last few years or so and is limited to scraping 18,000 tweets every 15 minutes. Thus, we generated our dataset iteratively by appending the results from these web scraping parameters after each search. It can be noted that we had twice the amount of before election data compared to after the election which could lead to incongruous results; especially when discussing the optimal number of clusters k, from Kmeans.

While we acknowledge this potential shortcoming, we feel that both of our primary data sets were large enough to gauge the general pulse of Twitter discussion surrounding the 2020 election.

3. Data Cleaning and Exploration

The following shows our initial data frame after we obtained the tweets.

	user	id	location	date	favourites_count	text	retweet
0	PatsFan876	2.999235e+09	NaN	1/26/15	359685	Busting stereotypes here... I'm a truck drivin...	True
1	Yasu_Al_Masih	8.830000e+17	Nazareth	7/7/17	15736	I COULDN'T RESIST POSTING THIS ONE!😉😉😉😉😉😉😉🤦 ...	True
2	JBK11	1.503627e+07	Canada	6/7/08	43347	Busting stereotypes here... I'm a truck drivin...	True
3	SoyEIG	3.209835e+07	NaN	4/16/09	432	The dismissive hand wave is what's most disres...	False
4	cynthianatalie	2.110394e+07	Los Angeles	2/17/09	4952	Inside #DonaldTrump's "serial bad behavior" at...	True
...	...	...	...	...	...	...	...
97195	Kimberl93935319	9.230000e+17	NaN	10/24/17	14605	@Amy_Siskind #DEMFOCUS2020 We need to focus re...	True
97196	sharknadoalert	1.280000e+18	NaN	7/16/20	6375	"Missed opportunities add up to a losing strat...	True
97197	BereniceJB_	4.312218e+08	Orange County	12/8/11	29087	The Gun Lobby is calling on voters to vote for...	True
97198	1989Mckoy	8.440000e+17	NaN	3/22/17	135684	Please vote #registertovote if you have yet to...	False
97199	davescar1	5.778776e+07	Spanish Fork, UT	7/17/09	74464	@njlang4 @ACTBrigitte The president was amazin...	True

97200 rows × 7 columns

Figure 1: Original Look at before election dataframe

The above figure shows that tweets can be rife with emojis, special characters, and hyperlinks. These will be removed in the preprocessing step. Also, note the prevalence of retweeted data. This speaks a lot to the ecosystem of Twitter's platform. At a single click of a button, users can garner more attention than original tweets. Since locational data was collected from user bios, we saw a lot of variants for the same locations. If we wanted to perform geolocational sentiment analysis, we could purchase a premium version of tweepy to access the latitude/longitude of each tweet. Additionally, to ensure that we were not assigning disproportionate scores to our vectorized features, we removed retweeted data from our analysis.

3.1. Preprocessing and Feature Engineering

Through our research, we discovered numerous publications that refer to boilerplate python functions when it comes to cleaning tweets (Ganesan, 2019). After combining the use of regular expressions to address common formatting styles of election-based Twitter data, we have developed a function that helped us in removing hyperlinks, tickers, @mentions, special punctuation, whitespace, stopwords, emojis/special characters, lowercase our data and extract hashtags. We also removed retweets along with several other unnecessary columns such as ID, Date, favourites_count, location from the dataframe since we had no plans on integrating them into our analysis.

3.2. Hashtags

Hashtags allow users to distill the main points they are trying to ascertain. Although many empty fields are found following the extraction, they provided a manual way for our group to intuitively label the gist of each tweet. For example, if a tweet contains [#ElectionFraud] vs [#Trump2020] then we have two starkly different viewpoints (most likely) that we can distinguish with #hashtags over the content of the tweet itself. Also, it was observed that there were many more variants of election hashtags than we could have searched for.

	hashtags	index		hashtags	index
0	#election2020	6496	0	#vote2020	3189
1	#trump	3082	1	#uselection2020	2903
2	#donaldtrump	2306	2	#election2020	1956
3	#biden2020	2157	3	#biden2020	1772
4	#vote	1911	4	#trump	1769
5	#biden	1824	5	#joebiden	1580
6	#kamalaharris	1667	6	#biden	1521
7	#debate2020	1535	7	#donaldtrump	1463
8	#joebiden	1011	8	#uselection	969
9	#harris	966	9	#trump2020	693

Figure 2: Hashtag Distribution before and after the election

3.3. Tweet Sentiment via Text Blob

Instead of using a Naive Bayes Classifier to identify the sentiment of tweets on a more granular scale (disgust, anxious, positive…) which would require vast amounts of hand-labeled data and hours of work), we decided to use TextBlob to scale tweets based on polarity and subjectivity between 0 and 1. This helped in our investigation of clustered data which allowed us to gain a high-level sentimental snapshot of tweets.

4. Methodology

4.1 Tokenization and Vectorization

The following diagram represents our Vectorization pipeline: Tokenizer, Stop Words Filter, CountVectorizer, and Inverse Document Frequency:

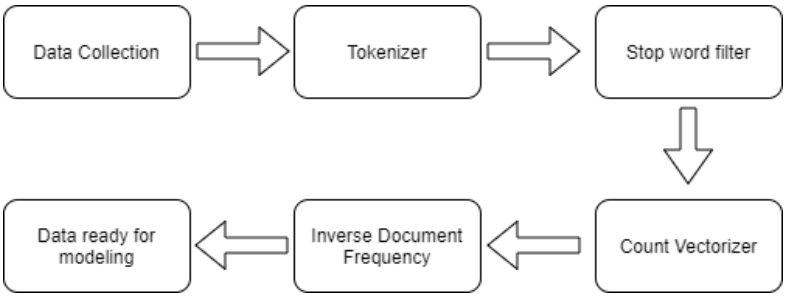

Figure 3: Vectorization Flow Chart

Text vectorization is the process of converting text into numerical representations. This module in the project pipeline answers how we vectorized our data in preparation for modeling. The first step in the pipeline "Tokenizer" converts full text into individual words. Since typical words like "a" and "the", are used a lot of times in almost any sentence, it is better to remove them as they do not contain any informational value. The "Stop Words Filter" stage handles the removal of frequent words in the English vocabulary. "CountVectorizer" counts the occurrence of every word in the vocabulary within a document. As the same words could be used again and again in multiple documents, Term frequency is not a real indicator of information value for a word. "Inverse document frequency" assigns a lower weight to words that occur frequently across many documents and vice-a-versa.

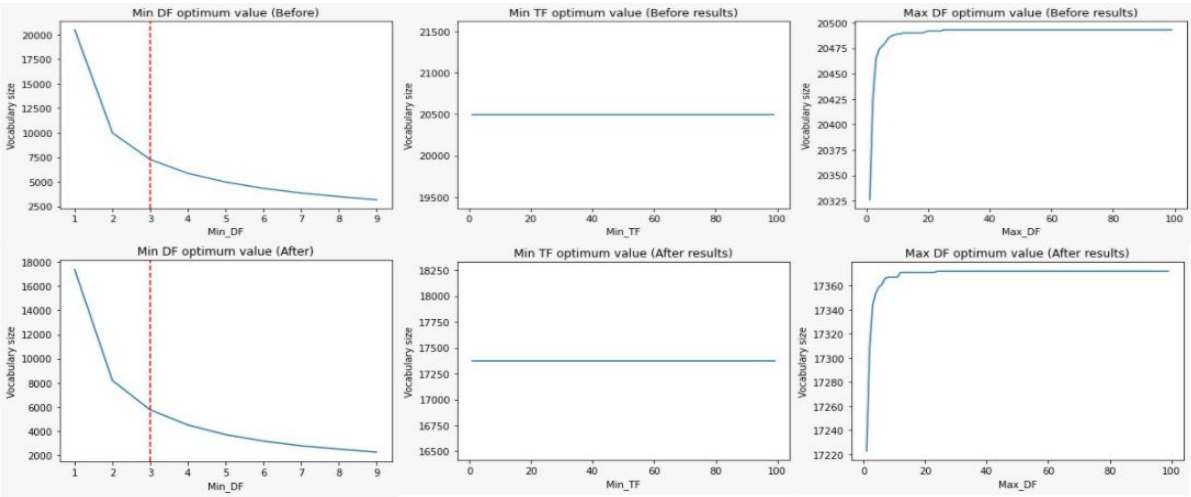

Figure 4: Countvectorizer Hyperparameter Tuning

The sparsity of the vectorized matrix is directly proportional to the vocabulary size. If we have a highly sparse matrix, it is difficult for the machine learning algorithm to optimize and find an accurate solution. CountVectorizer has several hyperparameters that can be tuned to better suit our dataset. MinDF is the minimum document frequency that must be satisfied for a word to exist

in the vocabulary. Including all the words in the vocabulary would not make sense as the vocabulary size correspondingly increases. Therefore, to strike the right balance for MinDF, we plotted various values of the same and tracked the change in vocabulary size. Looking at the figure, we could see that Min_DF = 3 would be optimal for unwanted words in the dataset, thus retaining the uniqueness of words. Highly frequent words lose their ability to uniquely represent a tweet. Therefore, we carried out the same process to select optimal Minimum Term Frequency and Maximum Document Frequency values.

4.2 User Aggregation

We wanted to perform data exploration in the form of analyzing users instead of tweets. As we were interested in identifying patterns that separate users from others, we had to perform user aggregation. After performing user aggregation, every username became a primary key and all tweets that the person had written were collected in the clean_text column. This technique seemed to be effective when we were representing a person on the PC1 and PC2 dimensional space where every scatter point illustrated a person. We used the original cleaned dataframe for other modeling purposes.

5. Models

5.1 Unsupervised Models

5.1.1 PCA

We used Principal Component Analysis (PCA) as a means of dimensionality reduction to understand how principal components contribute to the cumulative sum of variance explained. Looking at both the graphs, we see that around 500 Principal components are required to explain more than 75% of the variance in the before election data and 90% variance after the election. This shows that we have enough variance in the dataset that needs a collective effort of more than 500 Principal components to explain.

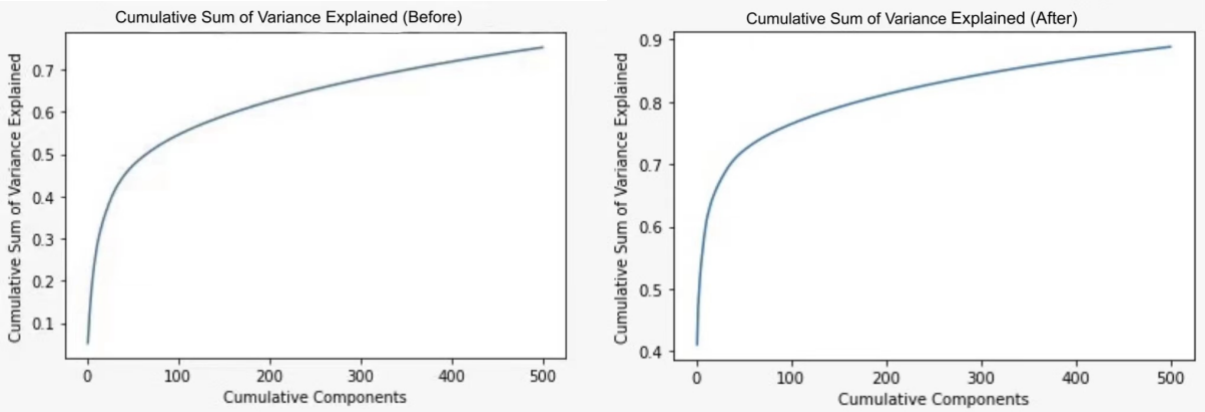

Figure 5: PCA Cumulative Sum of Variance Explained

5.1.2 KMeans

We used the silhouette scoring approach to select the optimal clusters from the before and after election datasets. Silhouette scoring techniques use a distance-based approach to minimize intra cluster variation. Looking at the change in the optimal number of clusters from K = 2 to K = 4 suggests that there could be more topics of interest after the election. This phenomenon will be further analyzed in the upcoming sections.

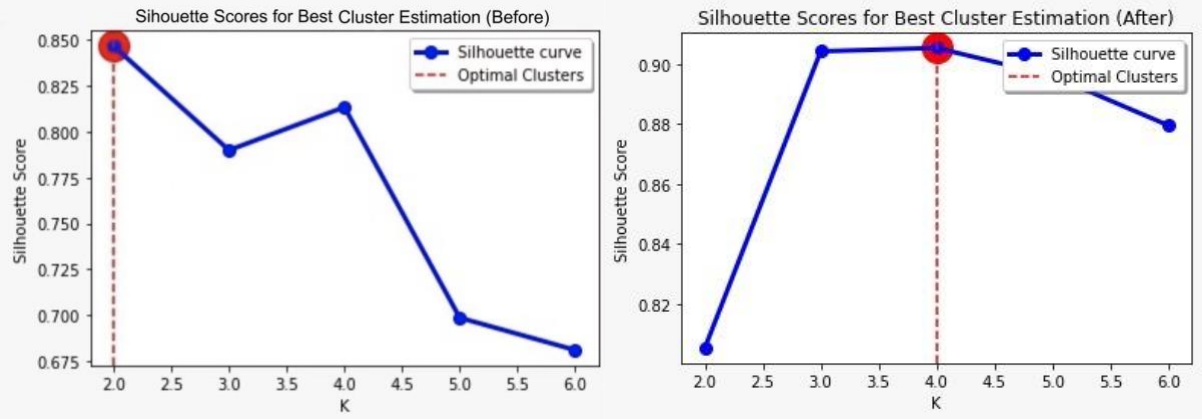

Figure 6: Silhouette Score for Best Cluster Estimation

5.1.3 KMeans and PCA

After selecting the best clusters that represented our dataset, we combined both the PCA and KMeans analysis for inference. We represented every user on the two-dimensional space formed by the first two principal components and colored them based on the cluster assigned by the KMeans algorithm. We were able to infer interesting points about the outliers in the two-dimensional space. Amongst the outliers were the journalism channels like “ElecCollPolls”, “electionRobot”, “StatesPoll” which were giving constant updates to users regarding the election.

Some users posted repetitive content in most of their tweets mentioning different people in each one. E.g., a user @Krishdangal1 tweeted "*Why is Joe Biden now promising to do so many things for us and he did nothing last 8 years when he was VP, simply beside making his son- Hunter Biden Rich?*" 12 times mentioning different people in each tweet. We expected to find Pro-Biden or Pro-Trump outliers in distinct clusters. Although, every cluster had polar users in its outliers we were not able to conclude satisfactorily that each cluster distinctly represents polar users supporting different candidates. Therefore, we need more historical data to check for any communities in the dataset which could be handled more efficiently by community detection techniques such as the Clique Percolation Method.

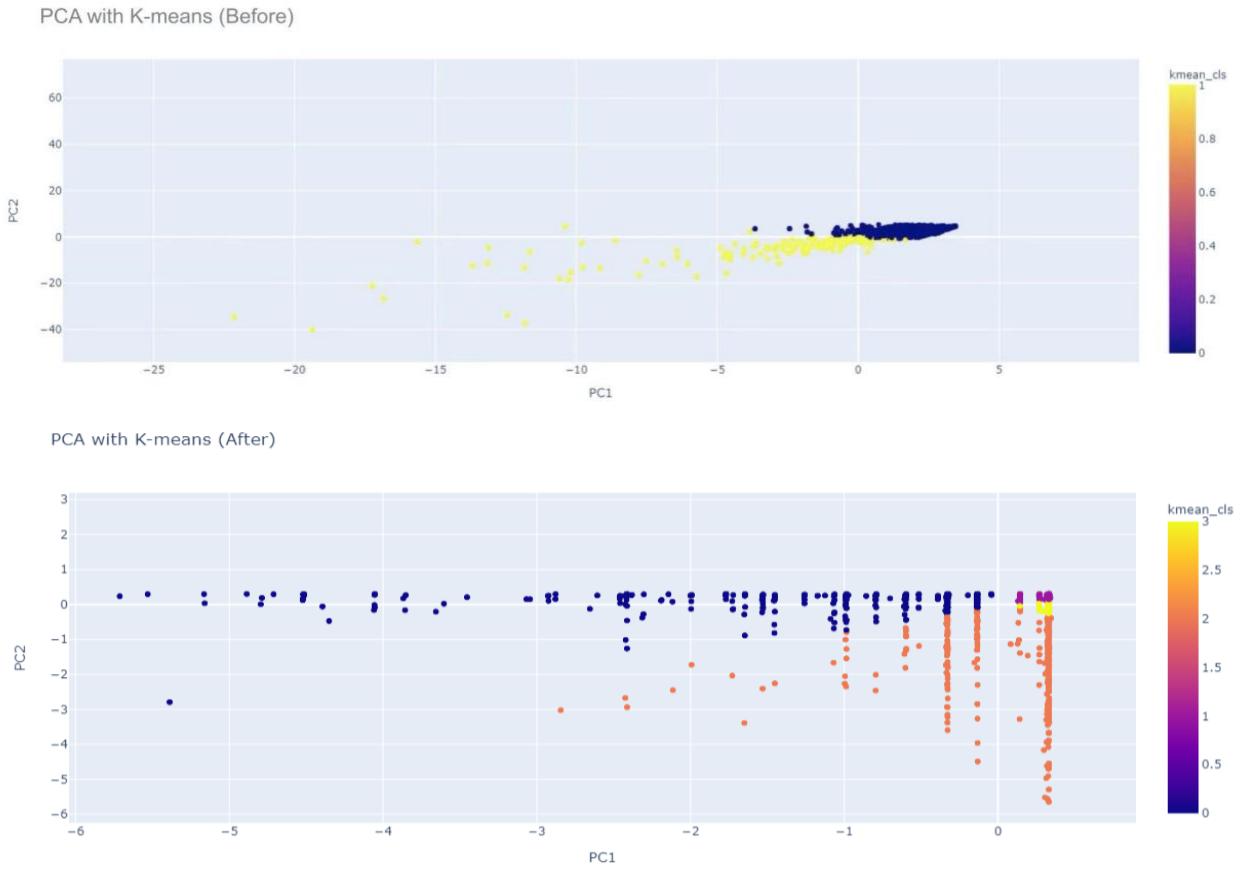

Figure 7: PCA with K-means for before and after election data

5.1.4 Hashtag Analysis

We divided the data into several chunks based on the cluster assigned by KMeans clustering and looked at the distribution of hashtags within the assigned clusters to validate the presence of communities. Based on analyzing the distribution of hashtags, we saw a trend that Joe Biden was more popular and had more positive support than Trump both, before and after the

election. We did not see any clear distinction amongst the clusters in the usage of hashtags which points to the conclusion that there are no evident distinct communities within our dataset.

	hashtags	index
0	#election2020	1244
1	#biden2020	552
2	#trump	545
3	#biden	505
4	#kamalaharris	436
5	#vote	402
6	#donaldtrump	331
7	#joebiden	307
8	#debate2020	301
9	#harris	289

	hashtags	index
0	#election2020	5252
1	#trump	2537
2	#donaldtrump	1975
3	#biden2020	1605
4	#vote	1509
5	#biden	1319
6	#debate2020	1234
7	#kamalaharris	1231
8	#vote2020	740
9	#joebiden	704

Figure 8. Before Election Hashtag Distribution

	hashtags	index
0	#vote2020	394
1	#votelikeyourlifedependsonit	191
2	#music	191
3	#countrymusic	190
4	#song	190
5	#behappy	169
6	#election2020	148
7	#florida	147
8	#thanks	145
9	#biden2020	144

	hashtags	index
0	#uselection2020	213
1	#vote2020	195
2	#election2020	154
3	#joebiden	150
4	#trump	150
5	#biden2020	134
6	#biden	132
7	#donaldtrump	129
8	#uselection	65
9	#bidenharris2020	54

	hashtags	index
0	#uselection2020	1189
1	#vote2020	1154
2	#trump	809
3	#election2020	744
4	#biden2020	665
5	#joebiden	654
6	#biden	634
7	#donaldtrump	624
8	#uselection	440
9	#trump2020	280

	hashtags	index
0	#vote2020	1446
1	#uselection2020	1374
2	#election2020	910
3	#biden2020	829
4	#trump	720
5	#joebiden	660
6	#biden	632
7	#donaldtrump	625
8	#uselection	428
9	#trump2020	338

Figure 9. Hashtag Distribution after the election

5.2 Supervised Models

5.2.1 Logistic Regression

We used logistic regression to find the most important words for predicting sentiment labels for both of our data sets. To make it easier to train the logistic regression model, we removed the neutral tweets, which do not have much influence on the sentiment of the tweets.

For model tuning, we decided to use areaunderROC as the evaluation technique. The reason is that areaunderROC is a measure of how well a parameter can distinguish between two diagnostic groups, which is the same as our goal: we want to ensure that the two target labels are separable. We used a grid search with 3-Fold cross-validation to optimize elasticNetParam (alpha)

and regParam (lambda). Based on the tuning result, we used L2 (Ridge Regression) along with lambda = 0.5 to regularize our model.

	Negative word	score
0	worst	-0.144012
1	corrupt	-0.132141
2	outrageous	-0.128522
3	boring	-0.127508
4	insane	-0.126952
5	pathetic	-0.124976
6	terrible	-0.122350
7	disgusting	-0.122205
8	disappointed	-0.121327
9	crap	-0.121073

	Positive word	score
0	proud	0.072794
1	right	0.072888
2	nice	0.074059
3	lol	0.076070
4	good	0.078195
5	better	0.081684
6	win	0.087409
7	love	0.089324
8	best	0.096129
9	great	0.101759

	Negative word	score
0	fucking	-0.140101
1	moron	-0.138700
2	violent	-0.133689
3	insane	-0.130664
4	fake	-0.129026
5	sorry	-0.127712
6	false	-0.126255
7	crap	-0.126070
8	bad	-0.125830
9	stupidity	-0.125397

	Positive word	score
0	free	0.069341
1	interesting	0.070580
2	proud	0.072433
3	happy	0.072708
4	nice	0.074199
5	great	0.085121
6	good	0.088222
7	best	0.088897
8	elect	0.089673
9	win	0.093586

Figure 10: Word Importance of Logistic Regression: Before (left 2), After (right 2)

It's interesting to find that the most important word for predicting a positive sentiment label before the election is 'proud'. There are many cases in which we could argue that this word does carry a positive label but the most common context this word appears in throughout the data set is 'proud boys', which on partisan platforms such as Twitter doesn't always have the best reception. The word 'right' is also an interesting observation because it can have three distinct meanings, either indicating a political party, direction, or moral justification. The political context makes the most sense but we don't have any way of ensuring that the context of words is maintained. The negative words for both datasets make perfect sense which indicates that our model is good at generalizing.

5.2.2 Random Forest

The optimal parameters following a grid search for our Random Forest were maxdepth=9 and numTrees=30 for 3-fold cross-validation. We used the same metric evaluator: areaunderROC as we wanted maximum separability between classes.

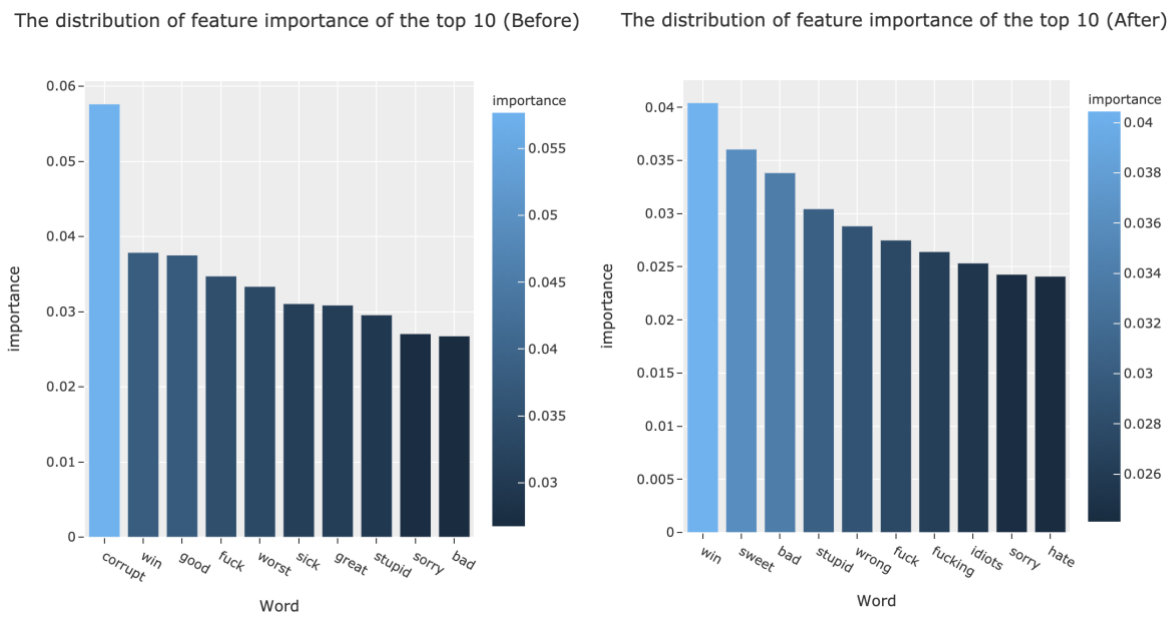

Figure 11: Feature Importance for Random Forest

Random forest showed the most important words for predicting the sentimental label of a tweet; however, it didn't indicate whether the word was positive or negative which is why these two algorithms can be seen as complementary to one another. We saw a fair amount of overlap of important words from the random forest model with the top words from the logistic regression model. Also, we did not find any striking difference in the most important words before and after the election suggesting the overall usage of words has stayed the same. "Win" is the most important feature for the after-election data which correlates to the fact that there would be a lot of discussion on Joe Biden winning the election race. Similarly, "corrupt" is the most important feature before the election result, which might convey a lot of speculation regarding the candidates.

6. Conclusion

In conclusion, this project served as an adequate exercise in analyzing the sentiment of tweets surrounding the 2020 US Presidential election. By scraping twitter's API for data containing search words about the candidates of the election, we obtained enough tweets for clustering users and figuring out the most important words for predicting the sentiment. We settled on TextBlob to assign polarity and subjectivity scores to our observations which aided in our investigation of clustered data as a means of gauging a high-level sentimental overview of tweets within the platform's ecosystem. We also chose to remove retweeted data because we felt it could lead to

disproportionate weights being assigned to the IDF score, leading us to make illogical inferences and conclusions concerning label sentiment.

We then performed PCA as a means of dimensionality reduction. Proactive users who were either Pro-Trump or Pro-Biden accounted for most of the time typical outliers observed. Since we performed user aggregation and count vectorizer, its term frequency increased which pushed them apart in the PC1 and PC2 dimensions of space. Similar outliers were detected on the after election results data. Following PCA, we applied KMeans to make inferences about the nature of our clustered data. We found an increase in the optimal number of clusters from 2, to 4 which suggests that more topics were being used to characterize clusters and discussions after the election was called, whereas before it was simply left vs. right. It seems that users reoriented and opened up to other topics such as Georgia, fake elections, voter fraud, etc.

Our findings from the hashtag analysis show that Biden was leading before the election was announced but took an even larger lead in terms of frequency distributions after the election was announced which is perfectly logical given the outcome. At any rate, this was a fascinating exercise for analyzing one of the most important Presidential elections to date and we look forward to improving our code in the future. And our results, following Logistic Regression and Random Forest showed significant overlap in the most important words for determining sentiment labels some of which were win, stupid, and corrupt, suggesting the high ability for model generalization. At any rate, we enjoyed this exploration and look forward to improving our methodology to try this code on future events that are discussed on Twitter's platform.

7. References

Ganesan, K. (2019, April). All you need to know about text preprocessing for NLP and Machine Learning. Retrieved November 30, 2020, from: https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html

8. Appendices

Appendix A: Original Look at before election dataframe (Figure 1)

Appendix B: Hashtag Distribution before and after the election (Figure 2)

Appendix C: Vectorization Flow Chart (Figure 3)

Appendix D: Countvectorizer Hyperparameter Tuning (Figure 4)

Appendix E: PCA Cumulative Sum of Variance Explained (Figure 5)

Appendix F: Silhouette Score for Best Cluster Estimation (Figure 6)

Appendix G: PCA with K-means for before and after election data (Figure 7)

Appendix H: Before Election Hashtag Distribution (Figure 8)

Appendix I: Hashtag Distribution after the election (Figure 9)

Appendix J: Word Importance of Logistic Regression: Before (left 2), After (right 2) (Figure 10)

Appendix K: Feature Importance for Random Forest (Figure 11)