How is information disseminated on Twitter: The 2020 U.S. elections

S. Jaoua, V. Prou, C. Tissier & G. Tissier

Supervisors: Elise Gourier and Federico Baldi Lanfranchi

Table of Contents

1 Abstract

The aim of this project is to study how is information disseminated on Twitter and try to use Twitter data on the 2020 U.S. Presidential Election to forecast their results. To explain the spread of information, we first found a positive relationship between multiple variable such as the presence of a URL link or the quantity of words indicating affective dimensions in a tweet and its retweet rate[1]. Among those variables, our results indicate that the number of follower of the twitter account who posted the tweet is the most important predictor for its number of retweet. Then, we tried to predict the outcome of the 2020 U.S. presidential based on tweets emitted during the election period. We first tested few models with features obtained from TF-IDF. After consulting a lot of articles on the topic, we finally settled for a Naive Bayes classifier which correctly predicted the results of the elections in 38 out of 50 states.

Last, but not least, we extended the model by adding economic and demographic factors in order to predict presidential elections within states. The new model showed some promising results for deciding a winner and outperformed the historical votes estimation in Ohio.

2 Twitter API

Twitter is one of the most popular platform of microblogging in the world and is used a lot for information sharing. In 2019, it's audience counted approximately 290.5 million monthly active users worldwide, and was projected to keep increasing up to over 340 million users by 2024[2]. Twitter is used a lot for the spread of political messages, for instance former U.S. president Donald Trump has twitted around 25,000 times during his presidency and even if it is not a perfect representation of the public opinion, it is still interesting to see the trends occurring on this social media and how they reflect results of political events, such as the U.S. 2020 presidential elections. Data employed in this study were collected using Twitter Streaming Application Programming Interface (API)[3]. We have successfully requested an academic research access, enabling us to extract 10 million tweets per month by choosing certain criteria they had to follow (e.g. date of creation, presence of certain words...) and additionally extract details associated to the user who posted the tweet (e.g. number of followers) or the tweet itself (e.g. number of time it was "retweeted"). However, Twitter API limits the number of tweets we can get by request to five hundred every fifteen minutes. The convenient solution that we found was to divide the three month preceding the U.S 2020 elections in several time sub intervals, thus additionally ensuring that our data where homogeneously distributed in time.

3 Spread of Information

Our first research questions is: how does information spread in Twitter? Following the work of Dr.Stefan Stieglitz and Dr.Linh Dang-Xuan Stieglitz and Dang-Xuan (2012) we decided that the right measure of spread for a tweet is the number of time a tweet was

[1] If the reader isn't familiar with twitter vocabulary yet, we advise to first have a look at: `https://help.twitter.com/en/resources/glossary`

[2] `https://www.statista.com/statistics/303681/twitter-users-worldwide/`

[3] See `https://developer.twitter.com/en/docs/twitter-api` for more details

”retweeted”. Moreover we conjectured that the following variables (see Section 3.1) can explain this number. More precisely we expect that the higher their value are the higher the expected number of retweet is.

3.1 Dataset

The dataset for this section consists of 1107046 tweets containing either the word ”Trump” or ”Biden” and that were posted from August 1, 2020 to November 1, 2020. The features selected for the tweets collected are the following:

- `RT` is our dependent variable representing the number of time a tweet was ”retweeted”, i.e. shared by another user.
- `AFFECT` is the number of words in the text of the tweet describing an emotion. To establish this count, we used the NRC Emotion Intensity Lexicon Mohammad (2018),Mohammad (2020) (NRC-EIL) of Dr. Saif M. Mohammad[1].
- `HASH`: is a binary variable which takes value 1 if the text of the tweet contains a hashtag and 0 otherwise.
- `URL` is a binary variable which takes value 1 if the text of the tweet contains an URL link and 0 otherwise.
- `FOLLOWER` is the number of follower of the tweeter user who posted the tweet.
- `FOLLOWING` is the number of twitter user the account who posted the tweet follows.
- `TWEETCOUNT` is the number of tweets the account that posted the tweet tweeted since its creation.
- `ACCOUNTAGE` is the age in year of the account of the tweeter user who posted the tweet.

  We first noticed that only 145394 tweets in our sample have a positive number of retweet, 128703 contain an hashtag and 418884 an URL link. The more general descriptive statistics can be found in Table 1.

	RT	AFFECT	FOLLOWER	FOLLOWING	TWEETCOUNT	ACCOUNTAGE
mean	3.47	2.75	24179.63	2670.96	52534.76	7.15
std	129.29	2.05	564420.94	10456.04	131811.73	4.07
min	0.00	0.00	0.00	0.00	0.00	1.00
max	41418.00	50.00	58479690.00	848782.00	6199225.00	16.00

Table 1: Descriptive statistics

3.2 Poisson Regression

As our dependent variable RT is a count variable and that all the exogenous variables are either counts or categorical, the natural choice of model for the regression is the Poisson one. We first tried to keep only the same variables as in Stieglitz and Dang-Xuan (2012) (i.e. not include $FOLLOWING$ and $TWEETCOUNT$ yet) for which the model was then given by

$$\begin{aligned}\mathcal{M}_1 : log(\mathbb{E}[RT|\cdot]) = \beta_0 + \beta_1 \cdot AFFECT + \beta_2 \cdot HASH + \beta_3 \cdot URL \\ + \beta_4 \cdot FOLLOWER \quad + \beta_5 \cdot ACCOUNTAGE + \epsilon \qquad (1)\end{aligned}$$

[1] https://saifmohammad.com/WebPages/AffectIntensity.htm

The results of the Poisson regression for model $\mathcal{M}_1$ can be found in Table2. where *coeff S&L* denotes the coefficients found in Stieglitz and Dang-Xuan (2012). We first observe that all coefficients are statistically different from 0 with p-value lower than 0.01 which confirms our conjecture(see the beginning of the Section 3). Moreover, they are all positive except for $HASH$ (-8.18e-02) which indicates that according to $\mathcal{M}_1$,the higher these coefficients are, the higher the expected number of retweet gets. For $HASH$ it is the opposite, the models indicates that tweets containing URL links are expected to be less "retweeted". To be more quantitatively precise, as a Poisson regression was applied, the interpretation requires an antilog (i.e., exponential) transformation of the coefficients to interpret the odds ratio. For example, the coefficient of $AFFECT$ of 1.16e-01 means that a one-unit change in occurrence of affective-processes words will generate 1.12 more retweets in average (exp(0.116) = 1.12).

	coeff S&L	**coeff**	**p-value**	**conf-lower**	**conf-higher**
Intercept		-5.23e-01	0.000	-5.26e-01	-5.19e-01
AFFECT	0.05	1.16e-01	0.000	1.15e-01	1.16e-01
FOLLOWER	5e-04	9.17e-08	0.000	9.16e-08	9.18e-08
HASH	0.83	-8.18e-02	0.000	-8.49e-02	-7.87e-02
URL	0.27	4.40e-01	0.000	4.38e-01	4.42e-01
ACCOUNTAGE	9e-04	1.43e-01	0.000	1.43e-01	1.44e-01

Table 2: Result of the Poisson regression for model $\mathcal{M}_1$

We then tried to add the variables $FOLLOWING$ and $TWEETCOUNT$ getting our second model:

$$\begin{aligned}\mathcal{M}_2 : log(\mathbb{E}[RT|\cdot]) = \beta_0 + \beta_1 \cdot AFFECT + \beta_2 \cdot HASH + \beta_3 \cdot URL + \beta_4 \cdot FOLLOWER \\ + \beta_5 \cdot FOLLOWING + \beta_6 \cdot TWEETCOUNT \\ + \beta_7 \cdot ACCOUNTAGE + \epsilon \end{aligned} \tag{2}$$

The result of the Poisson regression for this $\mathcal{M}_2$ can be found in Table3. As before, all the coefficients have a p-value lower than 0.01 and are thus statistically different from 0. Once again they are all positive except for $HASH$ (-6.71e-02) which shows that according to $\mathcal{M}_2$,the higher these coefficients are, the higher the expected number of retweet gets. For $HASH$ it is the opposite, the models indicates that tweets containing URL links are expected to be less "retweeted".

	coeff	**p-value**	**conf-lower**	**conf-higher**
Intercept	-5.00e-01	0.000	-5.03e-01	-4.97e-01
AFFECT	1.17e-01	0.000	1.16e-01	1.17e-01
FOLLOWER	9.24e-08	0.000	9.23e-08	9.25e-08
FOLLOWING	6.54e-06	0.000	6.53e-06	6.56e-06
TWEETCOUNT	1.45e-07	0.000	1.39e-07	1.50e-07
HASH	-6.71e-02	0.000	-7.02e-02	-6.40e-02
URL	3.89e-01	0.000	3.87e-01	3.91e-01
ACCOUNTAGE	1.38e-01	0.000	1.38e-01	1.38e-01

Table 3: Result of the Poisson regression for model $\mathcal{M}_2$

The results that we have presented until now confirm mostly our conjecture (see the beginning of Section 3) except for the variable $HASH$. However despite the variables being statistically significant for both our models, they don't fit the data very well according to their Log-Likelihood and R^2 (see Table 4).

	S&L	$\mathcal{M}_1$	$\mathcal{M}_2$
Log-Likelihood		-1.99e+07	-1.97e+07
Pseudo R^2	0.22	0.07	0.08

Table 4: Goodness of fit of the Poisson regression models

3.3 Random forest approach

Because of the poor overall goodness of fit for both Poisson regression models $\mathcal{M}_1$ and $\mathcal{M}_2$ in last part, we tried to improve on the model given by Stieglitz and Dang-Xuan (2012) and use a Machine Learning algorithm: the Random Forest. We thus first tried a random forest model $\mathcal{M}_3$ with all the variables but $FOLLOWING$ and $TWEETCOUNT$, and then a random forest model $\mathcal{M}_4$ with all the variables. The result in term of goodness of fits are much better judging their R^2 coefficient, and have a decent Root Mean Squared Error, see Table 5.

	$\mathcal{M}_3$	$\mathcal{M}_4$
$\mathbf{R^2}$	0.86	0.87
RMSE	48.7	47.09

Table 5: Goodness of fit of the Random Forest models

Another improvement of this method over the Poisson regression is that it gives the importance of each predictor. We can see in Table 6 that for model $\mathcal{M}_3$ the number of follower in the most important predictor by far with 63% of importance whereas the presence of an hashtag doesn't provide much information (0.01% of importance) to estimate the number of retweets. For model $\mathcal{M}_4$, we still have that the presence of an hashtag doesn't provide much information (0.01% of importance) to estimate the number of retweets but the $FOLLOWER$ variable while staying the most important one see its importance decrease (39%) with the introduction of the $TWEETCOUNT$ and $FOLLOWING$ variable which combined account for 37% of importance. The only difference in results that we had with Stieglitz and Dang-Xuan (2012), i.e. that the we found negative coefficient for the variable $HASH$ in the Poisson models $\mathcal{M}_1$ and $\mathcal{M}_2$ can thus be explained by the low importance of the variable $HASH$, according to both random forest models $\mathcal{M}_3$ and $\mathcal{M}_4$.

4 The main approaches to election forecasting

The aim of this section is to understand how we can forecast the election of 2020. This will indicates how Twitter can represent the public opinion. Thus the aim is that given a tweet, we want to be able to predict for which candidate the author of the tweet have voted. This is a classification problem which is in this case a supervised learning.

	$\mathcal{M}_3$	$\mathcal{M}_4$
FOLLOWER	0.63	0.39
AFFECT	0.15	0.11
ACCOUNTAGE	0.15	0.06
URL	0.05	00.7
HASH	0.01	0.01
TWEETCOUNT		0.24
FOLLOWING		0.13

Table 6: Importance of the predictors in percents for models $\mathcal{M}_3$ and $\mathcal{M}_4$

4.1 Data Selection

In order to conduct the researches, a data selection is first needed. We need to train with a labeled data set. Since the vote is anonymous and most people don't reveal their vote specially on Twitter, we need to find how to label tweets. The purpose is also to label the tweets with true labels and not randomly assigning them. One idea here is to select tweets that states directly which candidates is supported. This can be done by selecting a list of hashtags that are biased, for example, the #maga can show that the author is in favor of the sayings of Donald Trump, or even, the #VoteBlue shows that a person is democrat. Now that the data is selected, we clean it before conducting futher analysis.The preprocessing of the data is going to be explained in the section 5.3 in more details.

4.2 Sentiment Analysis

Sentiment analysis is defined as a process that automates the mining of attitudes, opinions, and emotions from the tweets through Natural Language Processing (NLP). We did a Sentiment analysis on the tweets using VADER, a model used for text sentiment analysis that is sensitive to polarity (positive/negative) and intensity (strength) of emotion. The analysis returns a sentiment score that is a value between -1 and 1, where -1 is very negative, 0 is neutral and 1 is very positive.

"*president trump tops gallup poll of most admired man in 2020 fightfortrump maga2020* "

This particular tweet had a sentiment score of: 0.7841. This is expected as the tweets mentions Trump in a positive way. One can also use VADER to score the tweets to whether the sentences are positive or neutral or negative. To do so, we denote positive quotes if the sentiment score is greater than 0.2. We let an error margin to the sentiment analysis. If the score is between -1 and -0.2, than the sentence is negative. Otherwise, the tweets are considered neutral. From the figures below, you can see the distribution of the sentiment partitions.

A second idea to represent the subjectivity of the tweets. Since the major subject in the tweets is politic, we need a score that quantifies the amount of personal opinion and factual information. This is computed by the subjectivity score obtained using TextBlob. The higher subjectivity means that the text contains personal opinion rather than factual information. TextBlob calculates subjectivity by looking if a word modifies the next word. Now that we obtained two scores to describe the formulation and sentiment analysis of the tweets, the next step is to evaluate the importance of some words in our corpus.

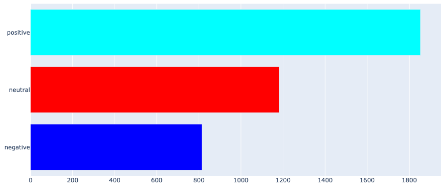


Figure 1: Sentiment categories for tweets Pro D.Trump

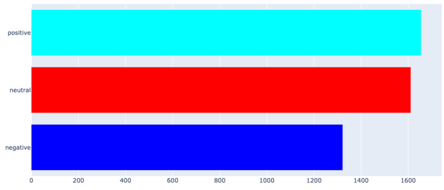


Figure 2: Sentiment categories for tweets Pro J.Biden

4.3 TF-IDF

Term Frequency & Inverse Document Frequency is a well-known technique in Natural Language Processing (NLP) for obtaining useful words and their scores from the given corpus. The useful words will then be used as features in our models. Document Frequency(DF) is the number of tweets in our case in which the word occurs as a fraction of the total number of tweets. IDF, Inverse Document Frequency is the logarithm of inverse of DF. Another significant measure of the importance of a word is the Term Frequency (TF), it represents how many times a particular word has appears in the tweets. TF-IDF corresponds then to a matrix which can be represented as a vector for each tweet with the TF-IDF, the multiplication of TF and IDF of each words in the tweet.

4.4 Models Testing

In this section, we are interested in forecasting the 2020 U.S. elections thanks to basic models available on Python. The idea is to implement different models in order to select the best one to optimize. To decide which is the best, we output the F1 score and the Prediction score. The first model chosen is a basic logistic regression which estimate the probability of voting for Trump as a function of the features created (sentiment score and TF-IDF score of the words in the tweet). The second one is Decision Tree. A decision tree is a tree where each node represents a feature, each link represents a decision and each leaf represents an outcome. The idea is to build a tree for the entire training data and when we want to predict, the features are processed in the tree which give an output. Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest makes a prediction and the class with the most votes becomes the random forest model's prediction. After, we decided to implement SVM. The Support Vector Machines used here for a binary classification finds a hyperplane in the same dimension as the input in order to separate the data points.

Table 7: Scores of the different models on Test set

	F1-score	Accuracy score
Logistic Regression	0.605	0.627
Decision Trees	0.760	0.783
Random Forest	0.97	0.778
SVM	0.67	0.544
Hard Voting Classifier	0.580	0.684
Soft Voting Classifier	0.730	0.756

In fact, the algorithm separates the training classes by a hyperplane, which is chosen to maximize the margin and uses it as decision boundaries. Finally, the two last classifier are obtained from a Voting Classifier. The main idea of a voting classifier is to train an ensemble of models an predicts an output. In our case, we decided to use decision tree, logistic regression and naive bayes gaussian models. Naive Bayes will be explained in the next section. In voting classifier, we have two different techniques that are called hard voting classifier and soft voting classifier. The hard voting predicts an output based on the majority voting for each model.The difference with the soft voting classifier is that it is based on the predicted probabilities for each classifier. Each model output a probability of chosen class as output and the soft voting predicts an output based on the highest probability. However, the results obtained in the table were not convincing enough to continue using one of these models. A deeper analysis in conducted in the next section.

5 Forecast of 2020 U.S. elections

After trying the previous classifiers with more or less success, we decided to use a different and more popular method: Naive Bayes. Its name comes from the fact that it makes the naive assumption that features are independent conditional on their class while using Bayes' rule. In our case we are interested in knowing $P(G_i|X)$ where G_i represents one of our two classes (pro Trump/pro Biden) and X would be a tweet (or more precisely a vector of words). Then according to what we just explained we get:

$$P(G_i|X) = \frac{P(X|G_i)P(G_i)}{P(X)} = \frac{\Pi P(X_j|G_i)P(G_i)}{P(X)}.$$

Despite the naive assumption this algorithm is very often used for sentiment analysis as it is rather easy to implement and tends to produce very good results.

5.1 Literature review

Sentiment analysis to predict the outcome of an election is nothing new in Machine Learning. We found a lot of documentation on the topic but we focused on a specific article *Sentiment Analysis of before and after Elections: Twitter Data of U.S. Election 2020* Chaudhry et al. (2021). Nevertheless Ding et al. (2017) was very useful for the use of hyperparameters and the choice of distribution for the Naive Bayes model (see 5.4), while Tunggawan and Soelistio (2016) shared interesting ideas for the preprocessing and analysis of the model's accuracy (see 5.3).

The authors of Chaudhry et al. (2021) inspired us a lot to perform Naive Bayes to study the results of 2020 U.S. elections as they achieved a prediction pretty close to the official results while explaining the few outliers states by looking at the evolution of sentiment in these places.

5.2 Data Set

Using Tweeter API with a query for english tweets containing *Trump*, *Biden* or both, we retrieved no less then 4.5 billion tweets emitted between July and November 3, the final day of the elections. As we intended to get the average sentiment score per states, we had to reduce our sample to 1.5 billion tweets with location (in the U.S.A.) given by their user. In Table 8, we display the number of tweets collected for each state with respect to the population. Although the proportion varies along the different states, we get a variance of 0.027%, showing that the data set is rather balanced. However, we get an average of 0.40%, highlighting again that tweets we collected (and Twitter in general) might no be fully representative of the general sentiment towards a candidate in a given state.

Table 8: Number of tweets collected compared with populations per states (for 12 first states)

	Number of tweets	Population	Proportion (in %)
AK	3887	733391.0	0.53
AL	13167	5024279.0	0.26
AR	7812	3011524.0	0.26
AZ	40556	7151502.0	0.57
CA	203996	39538223.0	0.52
CO	30335	5773714.0	0.53
CT	13832	3605944.0	0.38
DE	4697	989948.0	0.47
FL	125954	21538187.0	0.58
GA	37803	10711908.0	0.35
HI	8641	1455271.0	0.59
IA	9049	3271616.0	0.28

5.3 Data preprocessing and feature selection

Before selecting and training our Naive Bayes classifier, we need to transform the tweets we retrieved into acceptable features. We thus go through a series of slight modifications, to turn raw texts into a list of relevant words. First we remove all information we can't easily process: url, usernames, emojis, videos... Then we transform the text, we lower the words and remove the punctuation and numbers. Finally we tokenize the tweets, that is we create a list of words rather then a string of text, before applying lemming and removing stopwords (non relevant words such as 'a', 'would' or 'your'). For instance *"Hey #Trump! Is #golfing while 50+ millions Americans are #unemployed, millions are about to lose their $600 unemployment benefits; 157,278 Americans died from your incompetence with #COVID19 your version of #MAGA? https://t.co/7k7yAC875z"* would turn into *"hey trump golf millions americans unemployed millions lose unemployment benefit americans die incompetence covid version maga"*

The second step is to get the actual features we are going to feed our algorithm with. We use a bag of word representation of our corpus (the whole set of tweets collected). This means that we represent the latter with a matrix, which entries correspond to how many time a given word (words are indexed by the columns) of the corpus appears in each tweet (tweets are indexed by the raws). Finally we can choose or not (see section 5.4) to use TF-IDF to give a better score to relevant words (see section 4.3) and Bigrams, that is using not only unigrams (tokens) but also n-grams (groups of n words), which are helpful to avoid sentiment issues linked to sarcasm.

5.4 Training the Model

Now that we have our data set and we know how to collect features, we just need to find an appropriate training set: tweets for which we already know their true label. We mimic the idea of section 4.1, but this time we use even more biased hashtags (*#VoteBlue*, *#VoteRed*, *#DumpTrump* and *#SleepyJoe*) as we manually noticed that some hashtags were not relevant enough. As an example, #Maga was too often used by people criticizing it. In total, we collected sixty thousand tweets, half of them pro Trump, the other half pro Biden. It is important to notice that we removed these hashtags from the tweets during the preprocessing, to avoid overfitting these words.
Next, we try to optimize our Naive Bayes model by searching the best hyper parameters possible, especially the use or not of TF-IDF and Ngrams (in our case all possible combination of n grams for n$\leq$ 3). In order to achieve that, we divide the previous data set in a training set (we randomly picked 2/3 of the tweets) and a validation set (the 1/3 left). Then we use a GridSearchCV function from the online library SK-learn to choose the parameters that give the best accuracy on the training set (the accuracy was calculated by taking the average accuracy after a 10 fold cross validation). We found out that the use of TF-IDF as well as both unigrams and bigrams was best. The validation set gave us the final score of the selected classifier (see Table 9 and 10) which led us to better results than our first try in section **??**. Without getting into too much details, we made this process for two different distributions: Multinomial Naive Bayes and Bernoulli Naive Bayes, but we got very similar results and decided to use the multinomial model as the prediction turned out to be better.

Table 9: Confusion Matrix

		Predicted	
		Biden	Trump
Actual	Biden	8437	1584
	Trump	1201	8578

Table 10: Score of the selected classifier

	Accuracy	Precision	Recall	F1
score	0.863706	0.875389	0.850418	0.862723

5.5 Analysis of the results

Finally, the moment of truth, we used our freshly made classifier on our date set to predict the outcome of the 2020 U.S. elections using sentiment analysis, results are available in Table **??**. Although the precise percentage given by our model is not always close to the official results, the accuracy of the general prediction is rather good with 38/50 correct states (see Tables 12, 11 and 14 for more visual graphics). Moreover if we look at the historical states in favor of democrats/republicans (see Table 15), the sentiments predicted tend to match the data except for some of the outliers already presented in Table 14, showing a clear correlation between online sentiments on twitter and real sentiments. In addition, the sentiment score shows that for states subject to controversy the general opinion (on Twitter) was very balanced with no clear winner, which is in correlation with the very close results of the elections in these states. See for instance the case of Arizona(AZ) and Georgia (GA) in Table **??**, where Donald Trump filed a number of lawsuits contesting the election process.

However, as we pointed out, the predicted sentiment percentage can be quite far from the official results and even the prediction per state is not perfect. This can be explained by two majors factors. First we only collected an average of 0.4% of the population per states of tweets, which might not be enough to get a perfect prediction. However increasing the number of tweets failed to give a better prediction as we got similar results using a data set of 150.000 tweets. Similarly increasing the number of tweets in the training set failed to increase the performance of the model after getting past a certain threshold (around 30.000 tweets). To really improve the model, we would need a more diversified manually labeled training set. An other reason could be that the online sentiment is not the only parameter explaining a vote for a given candidate. We surely observed a logical correlation between the two, but we might need to add other factors to our model to better explain the official results, which is the topic of next section.

Table 11: Official results[1]

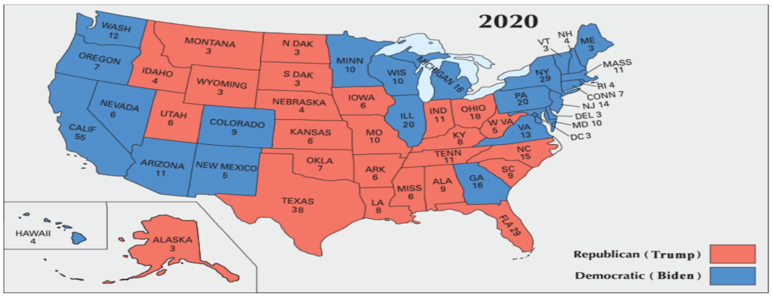


Table 12: Map of predicted Results

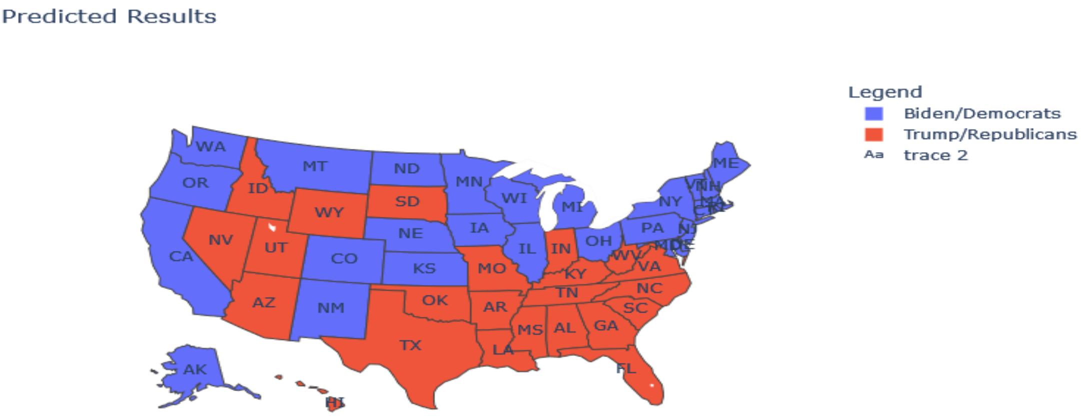
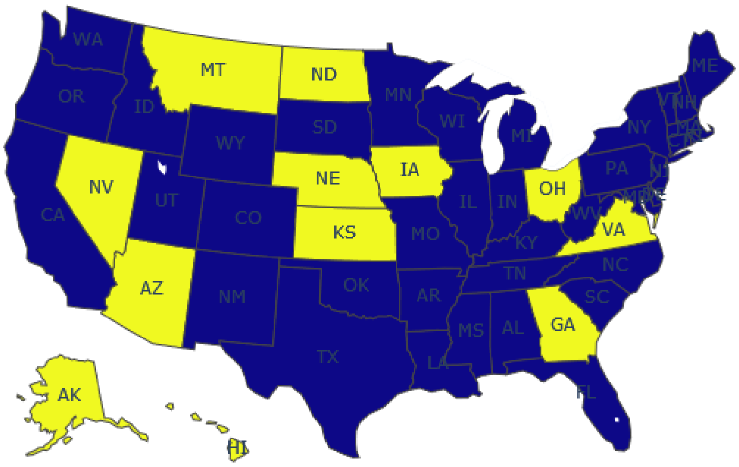


Table 13: Comparison of official results and sentiment prediction

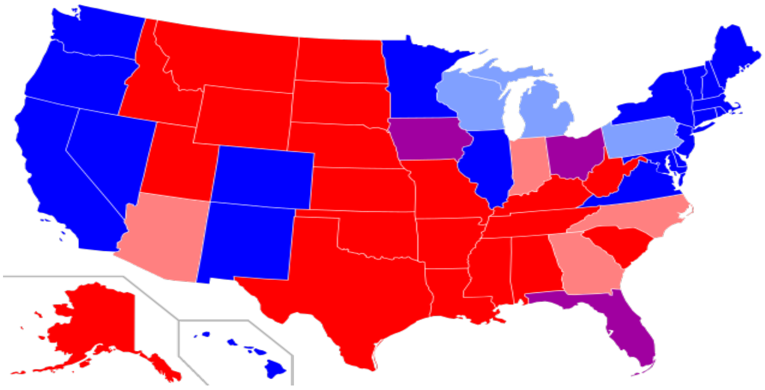
	True Biden	True Trump	Predicted Biden	Predicted Trump
AK	42.77%	52.83%	52.48%	47.52%
AL	36.57%	62.03%	46.04%	53.96%
AR	34.78%	62.40%	47.53%	52.47%
AZ	49.36%	49.06%	48.96%	51.04%
CA	63.48%	34.32%	52.79%	47.21%
CO	55.40%	41.90%	54.09%	45.91%
CT	59.26%	39.19%	51.76%	48.24%
DE	58.74%	39.77%	52.25%	47.75%
FL	47.86%	51.22%	47.87%	52.13%
GA	49.47%	49.24%	48.55%	51.45%
HI	63.73%	34.27%	49.79%	50.21%
IA	44.89%	53.09%	52.83%	47.17%
ID	33.07%	63.84%	45.61%	54.39%
IL	57.54%	40.55%	51.46%	48.54%
IN	40.96%	57.02%	47.84%	52.16%
KS	41.56%	56.21%	50.33%	49.67%
KY	36.15%	62.09%	47.85%	52.15%
LA	39.85%	58.46%	46.99%	53.01%
MA	65.60%	32.14%	53.44%	46.56%
MD	65.36%	32.15%	52.18%	47.82%
ME	53.09%	44.02%	55.7%	44.3%
MI	50.62%	47.84%	50.14%	49.86%
MN	52.40%	45.28%	51.37%	48.63%
MO	41.41%	56.80%	48.43%	51.57%
MS	41.06%	57.60%	45.71%	54.29%
MT	40.55%	56.92%	51.36%	48.64%
NC	48.59%	49.93%	49.82%	50.18%
ND	31.76%	65.11%	51.31%	48.69%
NE	39.17%	58.22%	52.88%	47.12%
NH	52.71%	45.36%	52.25%	47.75%
NJ	57.33%	41.40%	50.65%	49.35%
NM	54.29%	43.50%	55.33%	44.67%
NV	50.06%	47.67%	46.91%	53.09%
NY	60.86%	37.75%	52.04%	47.96%
OH	45.24%	53.27%	50.53%	49.47%
OK	32.29%	65.37%	45.08%	54.92%
OR	56.45%	40.37%	53.99%	46.01%
PA	50.01%	48.84%	50.29%	49.71%
RI	59.39%	38.61%	52.56%	47.44%
SC	43.43%	55.11%	46.56%	53.44%
SD	35.61%	61.77%	48.96%	51.04%
TN	37.45%	60.66%	44.83%	55.17%
TX	46.48%	52.06%	47.02%	52.98%
UT	37.65%	58.13%	48.57%	51.43%
VA	54.11%	44.00%	49.45%	50.55%
VT	66.09%	30.67%	58.52%	41.48%
WA	57.97%	38.77%	53.07%	46.93%
WI	49.45%	48.82%	50.54%	49.46%
WV	29.69%	68.62%	42.11%	57.89%
WY	26.55%	69.94%	44.44%	55.56%

Table 14: Map of wrongly predicted results



1

Table 15: Summary of results of the 2008, 2012, 2016, and 2020, presidential elections by state. With x-y representing the score for republican vs democrats, In red 4-0, in light red 3-1, in purple 2-2, in light blue 1-3 and in blue 0-4



6 Adding Factors: The case of Georgia

In political science, the most of the models used to predict election results feature economic data in addition to the support rate of the candidates usually estimated by poll surveys. However the model in the previous section predict solely based on the twitter sentiment for each candidates which is a proxy for the support rate. The goal of this section is to extend the previous model by adding factors.

6.1 Literature review

The framework for this section comes from Liu et al. (2021). The goal of the authors of this article is to propose a prediction model, the motivation for this new model is to substitute the support rate polls by the twitter sentiment analysis. The model is trained for the 2016 US presidential election, and on the counties of the state of Georgia. The choice of working on county level election instead of country or state level, is justified by the fact that presidential election is run county by county and it avoid misspecification due to geographical context.
The choice of the state is due to practical reason, Georgia has the second highest number of counties right behind Texas. But contrary to Texas, the election results are tight 13 and the counties won by the candidates are better distributed. Thus the state of Georgia constitute a fairly good training set.

They decide to use explanatory variables related to economic growth (GDP growth rate, per capita personal income growth rate and unemployment rate growth rate) and twitter support rate calculated as the number of users with positive sentiment to Clinton over the the total number of users in the county.
Independent variables:

- `GDP GR` $= \frac{GDP_{2016} - GDP_{2015}}{GDP_{2015}}$
- `PCI GR` $= \frac{PCI_{2016} - PCI_{2015}}{PCI_{2015}}$
- `unemploy GR` $= \frac{unemploy_{2016} - unemploy_{2015}}{unemploy_{2015}}$
- `Twitter support rate` $= \frac{\text{Number of users with positive sentiment to Clinton}}{\text{Total number of users}}$

For regression is the results of Hillary Clinton in the county in %. For classification, the binary variables that is decided on the winner of the county.
Dependent variable:

- Regression: Y = Clinton%
- Classification: Y = $\mathbb{1}$[County won by Clinton]

The twitter support rate used in their paper is calculated differently from the one we use in this project. Our twitter sentiment analysis is more straightforward and will not take into account the users or the intensity of the sentiment.

6.2 Model and data set

The model we established is based on Liu et al. (2021), for the 2020 US presidential election the dependent variable is related to the results of Joe Biden (in % for regression, binary for classification). We use the same economic growth variables for years 2019-2020. And using the Naive Bayes presented in the previous section we defined the Twitter support rate as the proportion of tweets labelled pro Biden in the county among the tweets related to the election, or in equation:
`Twitter support rate` $=\frac{\#\text{ tweets pro Biden}}{\text{Total }\#\text{ of tweets}}$.

Furthermore, due to empirical findings, we decided to add the population in the county as independent variable as it improves significantly the results for classification 25.

Features	Outcome
GDP GR PCI GR unemploy GR Twitter support rate Population	Biden election results

Table 16: 2020 prediction model

The economic data are downloaded from the Federal Reserve Economic Data website, the GDP and Per capita incomes are annual, the unemployment rates is averaged over the year. Due to difficulty to access the data, we use the population registered in 2022 instead of 2020.
We first train the model on the counties of Georgia and then use the trained methods to predict the results on the state of Ohio which is also a state that failed the classification of the previous section.
The state of Georgia constitute a good training set due to the justification above. Ohio is contested state, in 2020 7 counties out of the 88 were won by Biden. Compared to 2016, only 3 of the counties switched side, namely Lorain and Mahoning flipped from democrat to republican while Montgomery did the opposite.

The tweets are collected using twitter API, for each counties we search for all tweets within 20mi (∼32,2 km) of the center of the county. This approach is not the most precise but it is an easy way to get the tweets around a specific location.
The key words for the queries are:
Trump, republican, #Trump, @DonaldTrump, @realdonaldtrump, GOP, biden, @JoeBiden, #Biden, MAGA, #MAGA, POTUS, #POTUS, democrats, democratic, joebiden, election, president
The time interval considered range from 2020/08/01 to 2020/11/03. Also we make sure to not pick the retweets.
After gathering the tweets, we see that the amount of tweets collected varies greatly across counties, from 1 to 11036 in Georgia and from 0 to 8535 in Ohio. This is why we remove the counties with less than 20 tweets from the sample. Another odd observation is the county of Wilkinson in Georgia, despite having less than 9000 inhabitants, we collect 11015 tweets. This is due to the the fact that the center geopoint of Georgia is located in Wilkinson, hence twitter assign the tweets with no precise location to this area. The same

effect is less noticeable in Ohio. Thus we remove Wilkinson from the sample
Finally the training set includes of 163171 tweets shared among 144 counties of Georgia. And the test set includes 67684 tweets from 76 counties of Ohio.

We provide below some summary statistic for the state of Georgia and among its counties containing the most amount tweets.

	All counties		Most tweets	
	Mean	Std	Mean	Std
GDP GR	-0.02	0.05	-0.03	0.04
PCI GR	-0.03	0.30	-0.03	0.30
unemploy GR	0.51	0.28	0.68	0.29
TwitterSentiment	0.46	0.10	0.45	0.03
Population	74533	155700	198022	259447
Biden%	34.79	15.98	43.57	18.77

Table 17: Summary statistics for Georgia

The economic variables might induce a forward looking bias in our model, because we average the index over the whole year 2020, while the elections day is on the 3rd of November.
Moreover the economic variables indicates a drop in economy due to the COVID crisis. This may affect our model if we want to extend to future elections.

6.3 Results

6.3.1 In sample OLS

Table 18: In sample OLS

	All counties	Pro Biden	Pro Trump	Most tweets
const	13.547**	38.259***	21.108***	31.709
	(6.149)	(10.351)	(3.210)	(36.262)
GDP GR	11.361	-28.091	13.164	-79.155
	(25.275)	(35.831)	(15.101)	(59.467)
PCI GR	2.576	1.730	-2.829	-2.137
	(3.946)	(5.575)	(1.955)	(8.056)
unemploy GR	18.625***	12.520*	-6.586*	32.389***
	(5.112)	(7.259)	(3.750)	(11.614)
Support rate	21.795*	20.709	-5.552	-35.135
	(11.675)	(18.146)	(6.134)	(83.566)
Population	2.85e-05***	6.13e-06	7.50e-05**	1.71e-05
	(8.79e-06)	(6.93e-06)	(3.05e-05)	(1.27e-05)
Observations	144	34	38	39
R^2	0.283	0.384	0.236	0.499
Adjusted R^2	0.257	0.274	0.117	0.423
Residual Std. Error	13.773	8.948	3.711	14.260
F Statistic	10.890***	3.488**	1.979	6.568***

Note: *p<0.1; **p<0.05; ***p<0.01

First we look at the in sample results of the OLS to determine the importance of the variables. From 18 we observe that the most significant features are the Unemployment growth rate, Population and then Twitter sentiment. The other two economic growth related variables seems to be less significant, although it seems that the model indicates colinearity with a condition number of 3.79e+06.
A interesting observation comes when we look at the quarter of counties that vote the most for Biden and the other quarter for Trump, coefficient associated to the twitter support rate swap sign and is less significant for the pro Trump counties. Also the R^2 indicates that our models tends to work better for counties that vote more democrats.
The performance of the OLS is also correlated with the number of tweets collected but not due to the support rate calculated. The last column of 18 show the results for the counties with the most tweets collected. The R^2 is bigger in this case although the twitter support rate is less significant, this may be due to less variance in the sentiment 17.

	squared residual	residual difference	F-stat	Pr(>F)
Twitter sentiment	35885.82			
Twitter sentiment+economic	28175.47	7710.35	13.55	0.00
Final Model	26179.02	1996.45	10.52	0.00

Table 19: Anova of the models

We also perform an ANOVA table (19) to make sure that adding the economic variables and population to the model of the previous section is relevant.

6.3.2 Regression results

The methods trained for the regression use 20% of the counties of Georgia as validation set. We use the R^2 and the root mean squared error as metrics to our methods.
The neural network considered are simple feed forward with 16 neurons on each layers. The parameters for the regularized least square are $\alpha_{Ridge} = 20$, $\alpha_{Lasso} = 0.1$, $\alpha_{Elastic} = 0.072$ and $L1_{Elastic} = 0.5$.
We present the results in sample in table 20 and out of sample 21, also we present some predictions including the counties that have flipped in table 22. The 2016 results show the percentage of vote obtained by Hillary Clinton for the previous presidential election.
Overall, the results for the test set are not satisfying, they do not match with the results obtained by the historical percentage. The only methods that give some information are the regularized least square while the other ones completely fail to fit the test data.

	OLS	Ridge	Lasso	Elastic net	Random forest	Gradient boost	NN1	NN2	NN3	NN4
R^2	0.28	0.28	0.28	0.28	0.90	0.95	0.40	0.37	0.57	0.58
RMSE	13.48	13.51	13.49	13.49	4.94	3.60	12.31	12.68	10.47	10.32

Table 20: In sample regression results

	OLS	Ridge	Lasso	Elastic net	Random forest	Gradient boost	NN1	NN2	NN3	NN4	**2016 results**
R^2	-0.07	0.38	0.36	0.37	-0.77	-0.99	-0.83	-1.36	-2.88	-3.62	**0.96**
RMSE	15.01	8.91	8.99	8.98	15.01	15.92	15.25	17.34	22.20	24.24	**2.20**

Table 21: Ohio regression results

	OLS	Ridge	Lasso	Elastic net	Random forest	Gradient	NN1	NN2	NN3	NN4	2016 results	Realized
Lorain	70.31	46.57	47.60	47.39	70.31	72.48	46.36	39.19	41.20	22.05	47.63	47.96
Mahoning	58.30	35.37	35.34	35.32	58.30	56.51	25.69	25.49	26.90	13.95	49.48	48.36
Montgomery	60.71	46.43	47.28	47.10	60.71	59.04	41.38	38.32	41.95	22.69	46.95	50.18
Medina	58.05	40.57	41.14	40.98	58.05	63.96	34.09	31.30	14.00	32.70	34.92	37.53
Lake	62.95	42.82	43.56	43.39	62.95	65.35	32.61	30.73	15.54	13.98	39.59	42.45
Crawford	47.36	37.80	38.18	38.19	47.36	41.11	56.57	65.63	91.34	102.24	23.93	23.73
Ottawa	33.55	30.10	29.71	29.77	33.55	36.54	19.19	26.79	21.61	27.07	37.01	37.46
Allen	39.91	38.34	38.74	38.67	39.91	36.14	33.24	32.18	5.01	23.18	28.75	29.42
Washington	28.00	28.02	27.42	27.51	28.00	26.60	19.33	18.35	13.17	22.89	26.63	28.81

Table 22: Some regression predictions

6.3.3 Classification results

As before the neural network features 16 neurons on each layers. The results in sample are reported in table 23, out of sample in table 24. Note that we associated the positive to a win of Biden and the negative to a win of Trump. To compare to the historical prediction, we show the scores for the method that naively classify by the previous election results. Our model seems to perform well for classification, the K-means and the Neural network with 3 hidden layers provides fairly good results. In particular table 26 shows that K-means succeed in classifying the counties that have changed side, thus outperform slightly the historical prediction.

We also provide the results out of sample of the original model which does not feature the population of the county in table 25, as we mentioned earlier the results are worse which motivate addition the independent variable population.

	accuracy	precision	recall	F1
Decision tree	1.00	1.00	1.00	1.00
Random Forest	1.00	1.00	1.00	1.00
K-means	0.85	1.00	0.15	0.27
Gradient boosting	1.00	1.00	1.00	1.00
NN1	0.98	1.00	0.88	0.94
NN2	0.96	0.92	0.85	0.88
NN3	0.97	1.00	0.85	0.92
NN4	0.95	0.88	0.85	0.86

Table 23: In sample classification results

	accuracy	precision	recall	F1
Random Forest	0.80	0.30	0.86	0.44
K-means	**0.97**	**1.00**	0.71	**0.83**
Gradient boosting	0.72	0.23	0.86	0.36
NN1	0.84	0.31	0.57	0.40
NN2	0.66	0.12	0.43	0.19
NN3	0.92	0.55	0.86	0.67
NN4	0.66	0.12	0.43	0.19
2016 results	0.96	0.75	**0.86**	0.80

Table 24: Ohio classification results

	accuracy	precision	recall	F1
Random Forest	0.63	0.16	0.71	0.26
Kmeans	0.91	0.00	0.00	0.00
Gradient boosting	0.54	0.11	0.57	0.19
NN1	0.72	0.21	0.71	0.32
NN2	0.76	0.13	0.29	0.18
NN3	0.58	0.04	0.14	0.06
NN4	0.83	0.29	0.57	0.38
2016 results	**0.96**	**0.75**	**0.86**	**0.80**

Table 25: Ohio classification without population

	Random forest	K-means	Gradient boost	NN1	NN2	NN3	NN4	2016 results	Realized
Lorain	Biden	Trump	Biden	Biden	Biden	Biden	Trump	Biden	Trump
Mahoning	Biden	Trump	Biden	Trump	Trump	Trump	Trump	Biden	Trump
Montgomery	Biden	Biden	Biden	Biden	Trump	Biden	Trump	Trump	Biden
Franklin	Biden	Biden	Biden	Biden	Biden	Biden	Biden	Biden	Biden
Ashland	Trump	Trump	Trump	Trump	Trump	Trump	Trump	Trump	Trump
Lawrence	Trump	Trump	Trump	Trump	Trump	Trump	Trump	Trump	Trump
Hocking	Trump	Trump	Trump	Trump	Trump	Trump	Trump	Trump	Trump
Clinton	Biden	Trump	Biden	Biden	Biden	Trump	Biden	Trump	Trump

Table 26: Some classification predictions

7 Conclusion & improvements

We first identified in Section 3 several factors statistically significant to explain the spread of the information contained in a tweet, i.e. its number of retweet. The most important predictor among them is without a surprise the number of follower, followed by the number of tweet posted.

Then with our Naive Bayes classifier we managed to accurately forecast the results of the elections in 38 out of 50 states, showing a correlation between online trend on Twitter and official results. As we suspected that sentiment wasn't the only relevant factor for the forecast we finally added economic data and twitter support rate, yielding a very accurate classification (although the regression gave bad results) and highlighting the relevance of taking into account other factors for our model.

One can think of some improvements that we can do for this project such as testing other factors to explain the spread of information. About the forecasting, we could improve the accuracy of the model by using a better training set, that is getting more diversified labeled tweets. We can also search for other features like a variable related to historical vote to improve the prediction. One can also try the model on others state to validate its performance.

8 References

Stefan Stieglitz and Linh Dang-Xuan. Political communication and influence through microblogging–an empirical analysis of sentiment in twitter messages and retweet behavior. In *2012 45th Hawaii international conference on system sciences*, pages 3500–3509. IEEE, 2012.

Saif M. Mohammad. Word affect intensities. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan, 2018.

Saif M. Mohammad. Practical and ethical considerations in the effective use of emotion and sentiment lexicons, 2020.

Hassan Nazeer Chaudhry, Yasir Javed, Farzana Kulsoom, Zahid Mehmood, Zafar Iqbal Khan, Umar Shoaib, and Sadaf Hussain Janjua. Sentiment analysis of before and after elections: Twitter data of us election 2020. *Electronics*, 10(17):2082, 2021.

Tianyu Ding, Junyi Deng, Jingting Li, and Yu-Ru Lin. Sentiment analysis and political party classification in 2016 us president debates in twitter. 2017.

Elvyna Tunggawan and Yustinus Eko Soelistio. And the winner is…: Bayesian twitter-based prediction on 2016 us presidential election. In *2016 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pages 33–37. IEEE, 2016.

Ruowei Liu, Xiaobai Yao, Chenxiao Guo, and Xuebin Wei. Can we forecast presidential election using twitter data? an integrative modelling approach. *Annals of GIS*, 27(1): 43–56, 2021. doi: 10.1080/19475683.2020.1829704. URL `https://doi.org/10.1080/19475683.2020.1829704`.