**PRESIDENTIAL ELECTION 2020 PREDICTOR**

**Tiffany A. Devlin-Drye**

CIS4914 – Final Report

Spring 2019

Advised by Cheryl Resch

cheryl.resch@ufl.edu

10 April 2019

**Abstract**

The problem this project addresses is trying to predict the outcome of the 2020 presidential election. This is important for many reasons, ranging from political strategy to personal curiosity. In politics, knowledge truly is power. Therefore, the more information obtained about the strength of certain candidates, the more power one can have in influencing the outcomes. The solution in this project focuses on creating a combination of machine learning algorithms that uses both historical data and current Twitter sentiment to predict the outcome of various candidate matchups. There was a custom dataset created to supply the historical data. This was gathered from previous presidential and senate elections, and included the following useful features: incumbency, age of candidate, home state and political party. It also included a column for the labels “winner” and “loser”. Four different types of machine learning classifiers were run on this dataset and an overall score was assigned to the potential candidates based on the outcomes. This was then combined with real time public sentiment obtained from running a sentiment analysis on the last 2000 Tweets mentioning the candidate on Twitter. This resulted in a final score for each candidate. The candidate with the higher score was the predicted winner of that specific matchup.

In regard to technical results, using the four classifiers, an accuracy of 70-80% has been achieved in classifying candidates into a “winner” or “loser” category. The Twitter sentiment has been successfully set up and is working as per the Tweepy and TextBlob documentation.

In terms of prediction results, Donald Trump is confirmed to be a very strong candidate, but this is mainly due to the historical advantage an incumbent has during an election. Bernie Sanders and Joe Biden are both weaker candidates based on historical data, but stronger candidates based on Twitter sentiment, when matched up against Trump. When all factors are considered, the model repeatedly predicts Trump as the winner, though not always by a very wide margin.

(Keywords: machine learning, algorithms, president, election, senate, Twitter, sentiment, Python, Sci-kit Learn, Tweepy, 2020 elections, POTUS, data analytics.)

**Introduction**

**1. Objective.**

Historically, 50-60% of the eligible US population participate in presidential elections [1]. The outcomes of those elections, however, affect the entire US population and have significant global ramifications. The total influence a US president has during their time in office is impossible to accurately measure, but by any metric, it is extremely high. It is therefore important to countless people, near and far, to be able to answer the question: Who will be the next president of the United States of America?

The solution this project presents is a unique assembly of various predictive tools. The idea is that an aggregation of strong analytical methods will be more accurate, meaningful and powerful than any one of these tools used alone. Future projects within machine learning will benefit from a reliable method of utilizing all workable tools available. This project is a step in that direction and stands as an example of this.

**2. Problem Domain**

This problem lies in the domain of machine learning, which by default includes statistics and data analytics. Specifically, it uses a dataset of historical election data and trains four unique machine learning algorithms. Then it tests the accuracy of these algorithms to ensure quality before they are used to predict potential presidential candidates. This information is aggregated and then combined with the candidate’s Twitter sentiment. The Twitter sentiment is also obtained using machine learning. First, a dataset of movie reviews is parsed word by word and then analyzed. The analysis creates two sets – words people use when thinking positively about something and words people use when thinking negatively about something. Once this machine learning model is trained and tested, it can be used on Tweets to determine sentiment [2]. Using the outcomes from both the set of classifiers and the Twitter sentiment analysis, we can make reliable predictions about future elections.

**3. Literature Sources**

In researching this project, I looked into what features have been found to be reliable predictors of presidential elections and how these have been used in the past. I discovered a book called *Keys to The Whitehouse* [3] which was helpful in picking features. The modern-day leaders in the field of election predictive analysis are Nate Silver and his team at 538. Their resources [4, 5] were useful in gathering information about how this problem had previously been approached. Both of these sources were influential in my approach to developing this machine learning model. The models themselves were written in Python using SK Learn. Tutorials and documentation for both of these [6, 7, 8] guided me throughout the project. I learned about the various algorithms and how they work from videos on YouTube [9, 10] and online resources [11, 12, 13, 14]. The Twitter sentiment was researched in various places [2, 15, 16, 17] and in the end, a Python library called Tweepy, along with a Tweepy tutorial [2] and its documentation [18], was instrumental in building the Twitter sentiment part of the project. General information about elections was referred to [1] and the historical election information was obtained from several sources [19, 20, 21]. The dataset used was primarily custom made, but the names and winner/loser fields were obtained from datasets available from MIT and data.gov [22, 23].

**Solution – Technical Approach**

1. **Planning**

The starting point for this project was asking the question: What meaningful data is available? The data used in the machine learning algorithms had to be 1) a possible predictor of a presidential election, 2) available prior to the election, and 3) able to be represented by a number. I searched for datasets that may be helpful at data.org, MIT, Princeton, etc., but there were none that contained all of the information I needed. I then went to work in creating my own custom dataset to solve the problem. Data ruled out due to a low probability of being a predictor were features such as: eye color, height, weight, etc. Primary election results were unable to be used due to the primaries not having happened yet, and thus their results not being available prior to the election. Stances on specific issues weren’t considered as there was no reliable way to translate these into numbers. What I was left with were the following features for a candidate: home state, age, party affiliation and incumbency status. Finally, once these features had been identified, data was collected for the top two candidates in each presidential election since 1916 and the top two senate candidates in each 2016 senate election.

Elections of every type are determined not by any concrete or scientific method, but simply and only by public opinion. Polls determining this public opinion prior to a particular election are known to be accurate by 94.1% [5]. Therefore, I would be remiss in carrying out this project without a meaningful attempt to capture and utilize current public opinion. Traditional polls could not be used due to time and man power restrictions, but in a world of social media, public opinion is available openly and powerful tools utilizing Twitter are available. In combination with the historical data attained as above, I decided to incorporate live Twitter sentiment in producing a full analysis of the strength of a certain candidate.

1. **Twitter sentiment analysis implementation**

After extensive research I was able to find a reliable way to attain sentiment from Twitter – Tweepy and TextBlob. Tweepy is a Python library that interacts with Twitter’s API to carry out the various steps of retrieving Tweets. TextBlob is a text processor which includes a machine learning algorithm to carry out sentiment analysis. My implementation was heavily guided by a tutorial from Geeks for Geeks [2], but customized based on what was most appropriate for this particular project. Tweepy and TextBlob work together to create the Twitter sentiment analysis in the following way. TextBlob has a specific, large dataset of movie reviews. These reviews are already labeled as “Positive” or “Negative”. Each review has been parsed word by word and processed. The meaningless words (known as stopwords) were thrown away (words like “I”, “have”, “on”, etc.) and the rest were processed through a Naive Bayes classifier (type of machine learning algorithm) and categorized as either words people use when they are thinking positively about something or words people use when thinking negatively about something [2]. Having been trained on this data, the classifier is then ready for any textual data. Using Tweepy, any Tweets can be fed into TextBlob to analyze sentiment. For this project, Tweepy was used to fetch the most recent two thousand Tweets mentioning the candidate under consideration. For example, if we were interested in Donald Trump, Tweepy would be used to retrieve the most recent two thousand Tweets containing the words “Donald Trump”. TextBlob was then used to analyze whether these Tweets were positive, negative or neutral. For our purposes, neutral Tweets were thrown out and a percentage of positive and negative Tweets were calculated. A score was then given to the candidate between -0.5 and 0.5 which correlated with their sentiment analysis. For example, if a candidate had 67% positive Tweets, their score would be 0.67\*.5 = 0.335, whereas a candidate with 80% negative Tweets would be given a score of -0.8\*0.5 = -0.4.

1. **Classifier implementation**

In order to process the historical data obtained, I researched various machine learning algorithms and libraries that could be useful. In doing this research, I also discovered the Python library Scikit-Learn. Scikit-Learn is an extremely popular and extensive library for a wide range of machine learning and data processing methods [6]. Using Scikit-Learn, the custom dataset was processed on several different classifiers, all using supervised learning. Supervised learning means the dataset includes labels when training the model (as opposed to unsupervised learning, where the machine is left to categorize data on its own) [10]. In our case, the dataset consisted of features (age, home state, incumbency and political party) and labels (winner or loser). Here is a brief explanation of the type of classifiers used in this project:

Decision tree classifier – This classifier works by analyzing the strength of the various features and then building a tree with the most important features at the top. The data point (in our case, candidate) is taken through this decision tree feature by feature, with its path determined by the Boolean value at each feature node, until it arrives at a leaf labeled as either “winner” or “loser” [12].

Naive Bayes classifier – This classifier works by analyzing the features given to it and determining a series of probabilities based on comparing the features with the labels. In the end, two probabilities are calculated – the probability of “winner” and the probability of “loser” with the higher probability being the prediction of the model. Because it calculates a probability, a confidence of prediction can also be calculated. That is, when a candidate is determined to be a winner with a probability of 51% or when a candidate is determined to be a winner with a probability of 99%, both cases will provide a prediction of “winner”. However, the prediction with a 99% probability is obviously stronger, and that strength is something we can measure in this model [13].

Random Forest classifier – This is a type of ensemble classifier, which means that it computes several models using the same dataset and uses information from all of the models to compute a final result. In the case of the Random Forest classifier, each model used is itself a decision tree. Many decisions trees are created, and a randomly assigned piece of the data set is used on each individual tree. Once all of the trees have come up with a label, each tree in the ‘forest’ is granted a vote in the overall prediction. The most popular label based on the votes is considered the prediction for the entire forest [11].

Quadratic Discriminant Analysis classifier – This classifier attempts to create a mathematical function by which new predictions can be made. By “quadratic” we simply mean the function can be of a quadratic type; that is, to the second degree. This is as opposed to always being a linear function. Once the data is analyzed and a quadratic function can be found to reliably separate the dataset into classifications, it can be used on new data to make a prediction on its class [14].

Other classifiers were tested, but their accuracy was low and was not significantly more reliable than a 50/50 guess. The classifiers chosen for this project all had an accuracy of about 70% or higher. A custom ensemble was created in order to implement all four of the chosen classifiers. An ensemble of classifiers simply means any method by which the individual decisions of various classifiers are combined. Again, the idea being that if we use an aggregation of multiple reliable methods of prediction, our overall results will be stronger. For each classifier, the dataset was the same and the models were individually trained and tested before predictions were made. This step was done 1000 times on each model every time the program ran, and an accuracy percentage was calculated from this testing phase. Then the candidate we were interested in was run through each model and given a score in the range of

-0.125 to 0.125 so that the final result from all four classifiers would be in the range of -0.5 to 0.5. A negative value represents a classification as a “loser” and a positive value represents a “winner”. Each classifier prediction (1 for winner or -1 for loser) was weighted based on the accuracy of the classifier and the confidence of the prediction and therefore the total classifier score was determined by this function:

*(Decision Tree prediction)(accuracy of classifier)(confidence of prediction)(0.125) +*

*(Naive Bayes prediction)(accuracy of classifier)(confidence of prediction)(0.125) +*

*(Random Forest prediction)(accuracy of classifier)(confidence of prediction)(0.125) +*

*(Quadratic Discriminant prediction)(accuracy of classifier)(confidence of prediction)(0.125) =*

*Classifier ensemble score*

1. **Final prediction**

In the final analysis of a particular candidate, the Twitter sentiment score and the total classifier score are combined to give us a number that can be compared to that of another candidate. The predictions from both methods (classifier method and Twitter sentiment method), however, are not equally accurate. Both scores are therefore weighted based on how reliable their predictions are. For the Twitter sentiment, the reliability is hypothesized to be equivalent to traditional public opinion polls. This is based on the fact that these polls have been the accepted method by which public sentiment has been measured. Public polling has a 5.9% failure rate [5], and therefore we multiply the Twitter score by 0.941 to give it its proper importance. The classifier score is weighted based on the average accuracy of all the models. This is usually around 75% but is calculated at runtime. Therefore, our final score given to a candidate falls in the range of -1 to 1 and follows this formula:

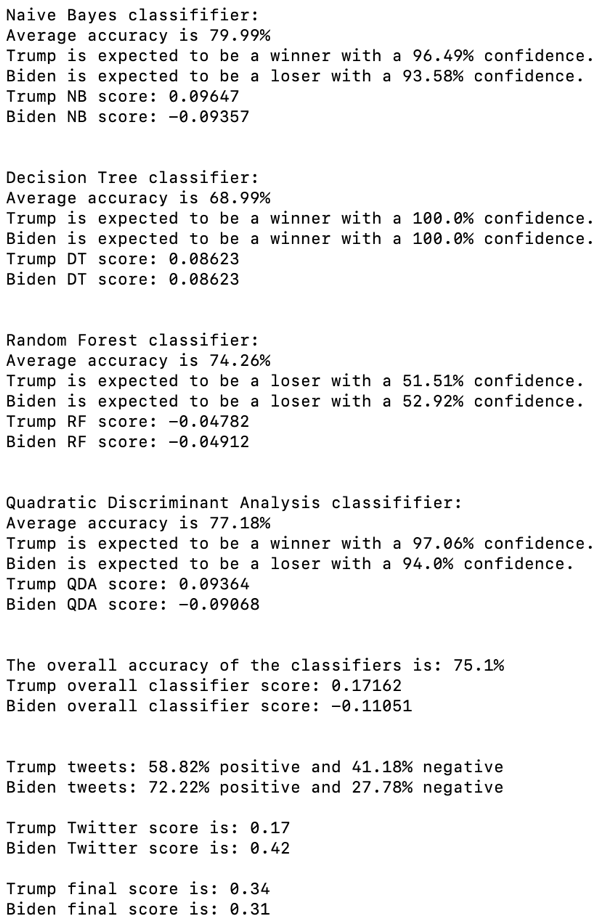
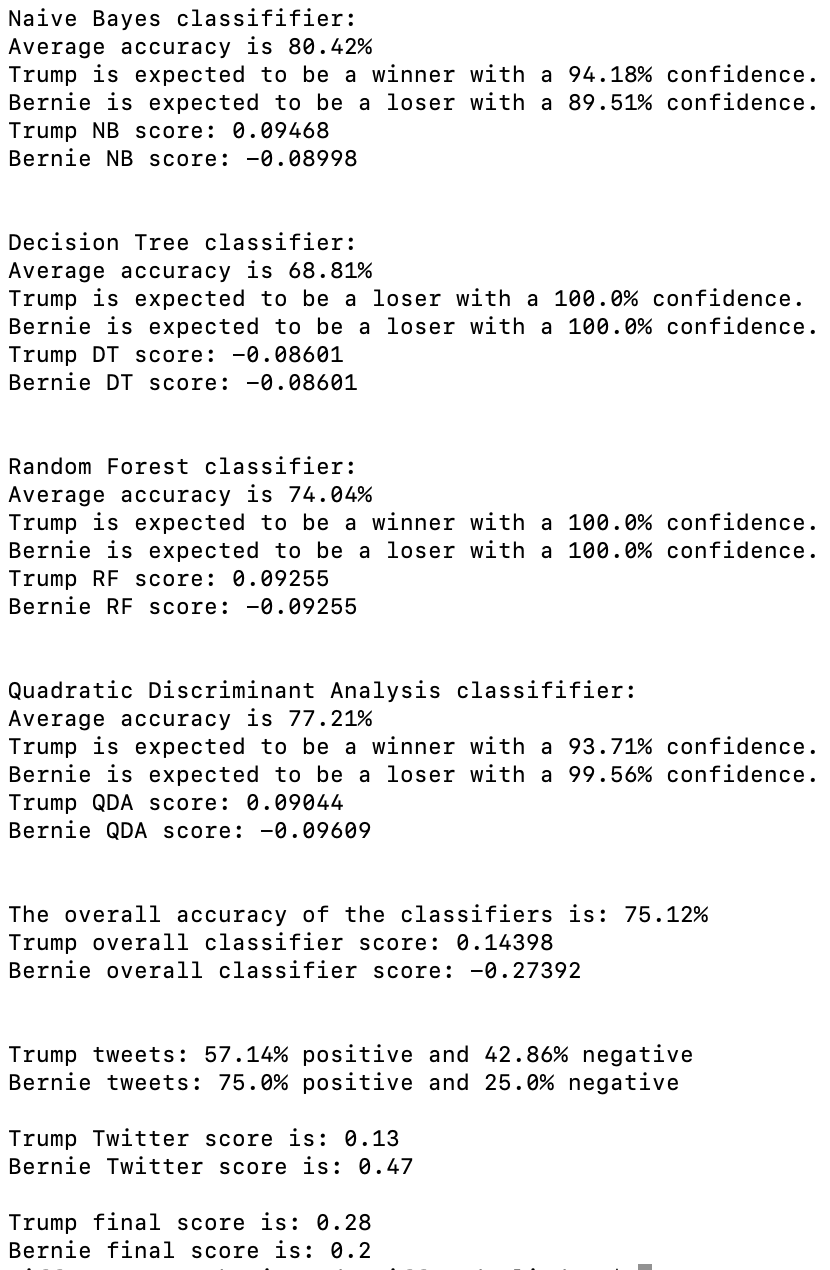
*(Twitter sentiment score)(.941) + (Classifier ensemble score)(Ensemble accuracy) =*

*Final candidate score*

Again, a negative score represents an overall prediction of “loser”, while a positive score represents an overall prediction of “winner”. In cases where the candidates are predicted to both be winners or both be losers, the score itself can determine a winner or loser of that matchup. In other words, we may have two strong candidates that are predicted to be winners based on historical data and Twitter sentiment. The numerical value of the score would then be able to distinguish the stronger winner and therefore the predicted winner if those candidates were to run against each other in an election.

**Solution – Technical Results**

Overall, it was found that President Trump has a high likelihood of being re-elected. This is largely based on the fact that historically an incumbent has a much higher probability of being re-elected. That said, other candidates could be strong contenders based on their Twitter sentiment. Take these example outputs between Trump and Bernie Sanders, and Trump and Joe Biden, respectively:



As can be seen, Bernie and Biden are both expected to lose based on all classifier models, but due to the sharp difference in Twitter sentiment, there ends up being a narrow margin between each of their matchups against Trump. In summary, Trump has an advantage based on historical elections, but does not have as much positive public opinion as some of the potential Democratic nominees.

**Conclusions**

1. **Summary.**

Starting with a simple desire to know the possible outcomes of the 2020 presidential election, I was able to delve into the world of machine learning and come out with some strong predictions. The first step taken was to access public sentiment using Twitter, as public opinion was thought to be one of the most reliable predictors. This was accomplished successfully in Python with help from the Tweepy and TextBlob libraries. Next was to create a custom dataset and custom ensemble of machine learning classifiers powered by data from Wikipedia and utilizing the Scikit-Learn library. In the end, reasonable weights were given to the outcomes and a strong prediction could be made for specific matchups.

1. **What was learned.**

Based on the initial predictions we can conclude that both Joe Biden and Bernie Sanders are strong candidates against Trump, specifically when reviewing the Twitter sentiment data alone. However, historically speaking, a sitting president has a significant advantage during an election and based on this, Trump is likely going to be re-elected. As for technical aspects, we can conclude that given a meaningful data set, there are several options for reliable machine learning classifiers. In our case the reliability was roughly as follows: Decision Tree classifier – 68%, Naive Bayes classifier – 80%, Random Forest classifier – 74% and Quadratic Discriminant classifier – 76%.

1. **Future work.**

There are several aspects of this project that could be improved upon with more time and man power. These fall under the following general categories: increasing the dataset, building a front end, improving the Twitter sentiment reliability, and applying the model to other areas.

Currently, the dataset consists of about 150 data points. In the end, only elections in which an incumbent was running were considered and the number of presidential elections with an incumbent is very small. Therefore, senate elections were a big part of the dataset, but this may have skewed the results in favor of an incumbent due to the fact that an incumbent senate member has an even higher advantage than a presidential incumbent (this is because states are generally more loyal along party lines than the country is as a whole). Taking all that into account, the dataset could be improved by both 1) adding more data points, and 2) correcting for factors that are present in senate races but are not as important in presidential races (i.e. incumbency and any other factors that can be found to be statistically different).

As for a front end, the ideas and predictions generated by this model would be conveyed more effectively if graphs, charts and an aesthetically pleasing UI were built. A starting place would be 538’s website (fivethirtyeight.com).

The reliability of the Twitter sentiment could be improved upon by choosing and testing other methods of retrieving the sentiment. One method that would be particularly interesting to test is a sentiment analysis based on emoticons. In that method, Tweets are processed in a similar way to the Tweets in this project, except the way the model differentiates “positive” or “negative” sentiment is by analyzing the emoticons used [17]. New and unique ways to analyze sentiment could also be invented and tested. Creating a way to distinguish other words people use to refer the same person would also be an improvement in the Twitter analysis. For example, the current iteration requires “Donald Trump” and will not produce correct results if only “Trump” is used. That’s because if “Trump” were used, we would be given sentiment for *everything* “Trump” – Trump hotels, Trump businesses, Melania Trump, etc. If, however, a method was developed to extract the Tweets referring to Donald Trump, using the term “Trump”, we would have a more accurate sentiment analysis.

A major way in which this work could be expanded upon is by applying the same models to other datasets and other elections. Adjusting this project to create predictive models for the primary elections, for example, would be fairly straightforward and not require much time or work. It could also be expanded into other areas such as sports, races, stock markets, etc., by changing the dataset and targets for Twitter sentiment. The possibilities are only bound by one’s imagination.

**Acknowledgements**

I would like to thank my advisor, Cheryl Resch, who made a true impression on me many semesters ago when we first had an opportunity to work together. Her focus on caring for students and a real desire to impart knowledge to them has been more impactful than she knows. I would also like to thank my husband who has always made it easy for me to make learning a priority and who helped in proofreading this report.

**References**

[1] FairVote.org. “Voter Turnout 101.” FairVote, www.fairvote.org/voter\_turnout\_101.

[2] “Twitter Sentiment Analysis Using Python.” GeeksforGeeks, 7 Feb. 2018, www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/.

[3] Lichtman, Allan J. The Keys to the White House, 1996: a Surefire Guide to Predicting the next President. Madison Books, 1996.

[4] NateSilver538. “2016 Election Forecast.” FiveThirtyEight, 8 Nov. 2016, projects.fivethirtyeight.com/2016-election-forecast/.

[5] NateSilver538. “The Polls Are All Right.” FiveThirtyEight, FiveThirtyEight, 30 May 2018, fivethirtyeight.com/features/the-polls-are-all-right/.

[6] “Learn.” Scikit, scikit-learn.org/stable/.

[7] Robinson, Scott. “Decision Trees in Python with Scikit-Learn.” Stack Abuse, Stack Abuse, 28 Feb. 2018, stackabuse.com/decision-trees-in-python-with-scikit-learn/.

[8] Scikit-Learn. “Scikit-Learn/Scikit-Learn.” GitHub, 13 Apr. 2019, github.com/scikit-learn/scikit- learn.

[9] Startups, Augmented. “Naive Bayes Classifier - Fun and Easy Machine Learning.” YouTube, YouTube, 26 Aug. 2017, www.youtube.com/watch?v=CPqOCI0ahss.

[10] deeplizard. “Supervised Learning Explained.” YouTube, YouTube, 22 Nov. 2017, www.youtube.com/watch?v=Quh6x4kG6VY.

[11] “Random Forests Classifiers in Python.” DataCamp Community, www.datacamp.com/community/tutorials/random-forests-classifier-python.

[12] “Decision Tree.” GeeksforGeeks, 7 Feb. 2018, www.geeksforgeeks.org/decision-tree/.

[13] Ray, Sunil, and Business Analytics. “6 Easy Steps to Learn Naive Bayes Algorithm (with Code in Python).” Analytics Vidhya, 7 Mar. 2019, [www.analyticsvidhya.com/blog/2017/09/naive-bayes-](http://www.analyticsvidhya.com/blog/2017/09/naive-bayes-) explained/.

[14] “1.2. Linear and Quadratic Discriminant Analysis.” Scikit, scikit- learn.org/stable/modules/lda\_qda.html.

[15] Joshi, Prateek, and Analytics Vidhya. “Comprehensive Hands on Guide to Twitter Sentiment Analysis with Dataset & Code.” Analytics Vidhya, 12 Feb. 2019, www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/.

[16] Ramteke, Jyoti, et al. “Election Result Prediction Using Twitter Sentiment Analysis.” 2016 International Conference on Inventive Computation Technologies (ICICT), 2016, doi:10.1109/inventive.2016.7823280.

[17] Wang, Hao, and Jorge A. Castanon. “Sentiment Expression via Emoticons on Social Media.” 2015 IEEE International Conference on Big Data (Big Data), 2015,doi:10.1109/bigdata.2015.7364034.

[18] “Tweepy.” Tweepy, www.tweepy.org/.

[19] “2016 United States Senate Elections.” Wikipedia, Wikimedia Foundation, 7 Apr. 2019, en.wikipedia.org/wiki/2016\_United\_States\_Senate\_elections.

[20] “Main Page.” Ballotpedia, ballotpedia.org/Main\_Page.

[21] “The Voter's Self Defense System.” Vote Smart, votesmart.org/.

[22] “Data.” MIT Election Lab, electionlab.mit.edu/data#tools.

[23] “Data Catalog.” Data.gov, catalog.data.gov/dataset?tags=election.

**Biography**

Tiffany Devlin-Drye was born in Dallas, TX and moved to the Tampa Bay Area in 2006. She currently resides in Palm Harbor with her husband and young son, Maxwell. She graduated Summa Cum Laude from St. Petersburg College with an associates degree and is currently completing her bachelor’s degree in computer science at the University of Florida (May 2019). She hopes to enter the field of machine learning shortly after graduation. Statistical analysis has been a lifelong interest of hers and the prospect of dedicating her career to using modern machine learning techniques to create sophisticated predictive models would be a dream come true.