Identifying an optimal machine learning model for election forecasting

A Project

Presented to the

Faculty of

University of North Carolina,

Greensboro

In Partial Fulfillment

of the Requirements for the Degree

Master of (Arts/Science)

in

Computer Science

by

Daniel Foster

May 7th, 2021

ABSTRACT

Election forecasting has become an important presence in political journalism and popular conversation. Contemporary machine learning modules have made the exercise available to the common data scientist and programmer. This project set out to find a model that would take an array of state specific data as input and return an output probability that a presidential candidate would win that state's electoral votes. The models were trained on 34 data features and tested on battleground states in the 2020 election. Random Forest Classifier was selected for its performance and evaluation usability. The model was retrained through 1000 iterations and the average probability of the iterations for each state was compared to popular forecasting models developed by FiveThirtyEight and The Economist. Random Forest Classifier shared an accuracy score with the more sophisticated corporate models, though it differed in the states that were incorrectly predicted on. Polling, past election margins and the percentage of advanced degrees were columns with high SHAP values, indicating feature importance for each battleground state.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER ONE

Introduction

Every four years pundits, journalists, researchers and everyday citizens pontificate on the outcome of the American election. How does it help anyone to know that a model predicts that Biden is 54% likely to win North Carolina? Will Biden supporters in North Carolina be lulled into a false sense of security with the knowledge that someone said Biden's got their state in the bag, thus potentially dampening turnout for Biden in that state? Or conversely, will grassroots Trump voters be mobilized by the idea that, in their view, the fake news media has released another story attempting to damage him? While both options are within the realm of possibility, that's not why people are interested in election forecasting. Similarly, the political campaigns themselves probably aren't too concerned with a particular forecast, other than its ability to give their opposition positive media coverage. For example, in 2016, there were a few Hillary Clinton surrogates that criticized one famous forecasting model for what they believed were unrealistic odds in favor of Donald Trump. (Politico, 2016) And while campaigns are surely interested in using polls to guide their operations, it's difficult to imagine a campaign allocating funding based on a specific forecasting model. In any case, these examples of forecasts impacting voters or the forecasts impacting the decision making of campaigns have meta utility, but I would argue they don't hold tangible value to anyone. So what's the point of all this? Ultimately, I think people are simply curious to know what math has to say about the chances of an event happening or not happening. It used to be that we looked at our favorite pollster, but we've come to appreciate that a single poll is not sufficient to explain things, we need some way to synthesize all the polls to give us a more complete snapshot. Machine Learning is a hot topic, in part, because the philosophical question is so intriguing: is it possible to predict the future based on past data? Maybe this is the real reason we're interested in election forecasting, especially when the stakes are so high.

This project was inspired by the popularity of FiveThirtyEight. FiveThirtyEight is an election forecasting website that was developed by Nate Silver. Silver gained notoriety in 2008 by correctly predicting 49 out of the 50 state outcomes and in 2012 correctly predicted all 50 state outcomes. Like most forecasters in 2016, FiveThirtyEight was wrong about the outcome, predicting a Clinton win. However, the site put Trump's odds at around 30%, higher than other forecasters. Silver has argued that models which rely primarily on "fundamentals," non-polling factors like the economy or Covid-19 rates, are not strong predictors. In his view, polling data is the prevailing gold standard of forecasting accuracy. In the search for an optimal machine learning model, I hoped to learn which features were the most important in predicting the election result in a state. Additionally, how would a classic machine learning model compare to the proprietary and secret model at FiveThirtyEight?

The primary aim of this project is to output a probability value given a battleground state's data. A battleground state is defined as a state with variable election outcome results over the past several elections, or one that has variability polling or low margin polling data. For the scope of this project, battleground states include Arizona, Florida, Georgia, Michigan, Minneota, New Hampshire, Nevada, North Carolina, Ohio, Wisconsin, Pennsylvania, and Florida. The project used 34 features to train the models on. The economic data is based on the number of normalized jobs in an arbitrary selected sector (Service, Construction, Financial, Manufacturing, and IT), normalized wages(Total Quarterly and Average Weekly), and the unemployment rate by state. Note: where the data is a binary difference between Trump and Biden, I selected the values that were the “Biden advantage”, in order to reduce redundancy in the model. Thus some features, such as polling, have negative values listed for Biden and there are no columns labelled for Trump. Other fundamentals include the campaign contributions difference of campaign contributions to Biden; the Cook Partisan Index indicating how a red or blue a state is; “elasticity” a FiveThirtyEight metric indicating how much a state can swing in one direction; the number of normalized Covid cases and deaths per state; a home state advantage; the Cost of Voting Index (how difficult it is to vote in that state); high school, college, and advanced degrees; urbanization; and religiosity race, age and gender demographic data. Polling data include the democrat marginal advantage in 2016 and 2012, and Biden’s poll advantage in 2020. Only the average polling advantage was used as of November 3rd, 2020, the day of the election.

Unlike many data science projects with thousands of observations, this project has the characteristic of only 44 observations, representing each state. While there are 50 states that vote in the Electoral College, seven of those states had little to no polling data in 2020. Pollsters likely avoid these states either due to their very large margins in presidential elections, their demographic homogeneity, or both. I therefore omitted these states, as any model forecasting model that avoided current polling data would be a non-starter. The states omitted are Kansas, Idaho, Illinois, North Dakota, South Dakota, and Rhode Island. The limited number of observations demanded that a wide variety of features were used.

The data was wrangled from a variety of methods. Most data was found on a number of different static websites and fortunately the need to write a scraper for the values of each feature was unnecessary due to the relatively few number of observations. Thus, most of the data was entered by hand. These included sources such as the Cook Partisan Index, Religiosity, demographic data, etc. For the 2020 polling data, I used a module developed to scrape Real Clear Politics which can be used to pinpoint polling data at a particular point in time during the election cycle, though I did not use trends in my model. A few other scripts were written to parse comma separated files. The actual data wrangled included economic fundamentals from the U.S. Bureau of Labor. I knew I wanted a snapshot of the economy and there were almost limitless options. Ultimately, I settled for normalized job totals for a handful of arbitrarily selected sectors, the unemployment rate of each state, and wages. The files had thousands of rows due to the organization. Each state had hundreds of rows further broken down into specific job sectors. The Jupyter Notebook hosted on UNC-Greenboro's servers was used as a workspace for hosting and running the code, written in Python 3. The data was wrangled with the Pandas library.

The data was split using the scikit-learn's train_test_split function. The test rows included the battleground states, since that is the target observations for prediction. Predicting on trained data is not advisable as it can lead to overfitting. The remaining states constitute the training set. The X variables include the polling, economic, and demographic data. The Y variable is whether Biden won that particular state, represented as "1", or "0" for a loss. The training and test data is run through the following models: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier, and XGBoost. The models will be described in detail in Chapter 2, along with an overview of their accuracy. The selected model, Random Forest Classifier, will be further in Chapter 3 along with the results of predicting the probability of each battleground state, the predicted probability confidence intervals, and the feature importance of each state. Chapter 4 will include a discussion of the results followed by the conclusion in Chapter 5.

CHAPTER TWO

Overview of the Machine Learning Models

Classification

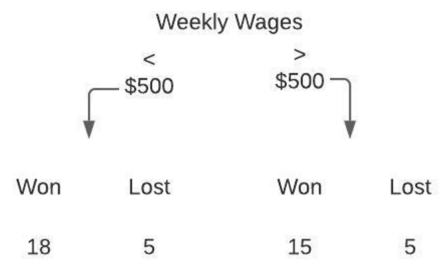
The project trains supervised machine learning models. Machine learning takes an input and uses a model's function to give an output based on a variety of parameters and mathematical functions. Supervised machine learning corrects the model until the algorithm reaches a maximum limit of performance. In this case the inputs consist of 34 independent polling, demographic and economic variables. The output is a dependent binary representation of Biden's outcome for that state (0 - did not win, 1- won). The output's win/no win defines this project as a classification problem, thus classification models are used.

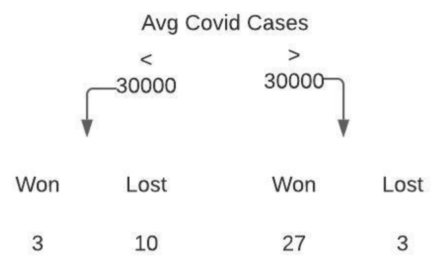
Logistic regression

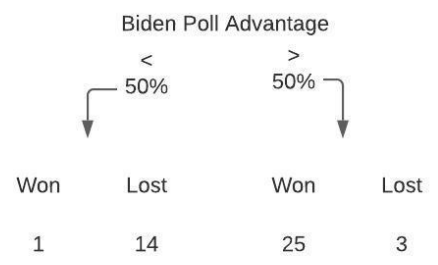
Logistic regression uses a sigmoid function, when mapped resembles an S-shaped curve. Most of the domain has output values that are close to zero or close to one, which make it appropriate for classification. Logistic regression uses a linear function $f(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_rx_r$, where the *b* series coefficients are weighted and adjusted to output the most accurate fit using a log-likelihood function. The linear regression is applied to the sigmoid function. (Stojiljković, 2020) (Navlani, 2019).

Decision Tree Classifier

Rather than a sigmoid function, a decision tree classifier uses a tree structure to output a prediction based on a quality variable split.







There are three splits in the above example: Weekly Wages, Average Covid Cases and Biden Poll Advantage. To measure the quality of the split , the sci-kit learn implementation of Decision Tree Classifier uses a method called gini impurity. The left leaves impurity of say Weekly Wages (<50%) is the square of the fraction of the won values minus the square of the fraction of the lost values. That value is subtracted by 1. The same process is applied to the right leaves. The impurity of the node itself is the square of the fraction of the left child minus the square of the fraction of the right child, and again subtracted from 1. The impurity of the left leaf is multiplied by the left child's fraction square and the impurity of the right leaf is multiplied by the right child's fraction squared. Take the difference of those values with the difference of the impurity of the node itself and the result value is the information gain. Repeat this algorithm for the remaining independent variables to find the highest gain. That's the root. The process is repeated by splitting the nodes again to find the remaining children. Sci-kit learn implements a minimum impurity decrease to determine whether a node should be split. (Maklin, 2019)

Random Forest

Building on the previous model, a random forest consists of many decision trees. Each tree in the mode makes a prediction and the output with the most votes is the model's prediction. For example, after populating 100 trees, 90 would predict that Kansas would vote for Trump, thus the model would correlate To guarantee that the model has a diversified set of values, the trees are selected at random with replacement in a process known as bagging. (Yiu, 2019)

Gradient Boosting Classifier

Gradient Boosting also uses trees, but each successor tree is intended to learn from the previous with the assistance of a loss function. The loss function determines the difference between the correct value and the predicted value and thus measures how well the tree is at predicting, in this case, the electoral outcome of a state. The next tree is built sequentially on this modified tree intended to reduce the residual loss. Once the loss is below a certain threshold or a set number of trees has been built, then the process terminates.

A variation of gradient boosting, called extreme gradient boosting, offers a range of hyperparameters that allow for more control over the model. (Nelson, n.d.)

CHAPTER THREE

RESULTS OF MODELS

Classification Report

The classification results in Table 1 show somewhat similar results. Recall that each model is tested on the battleground states, and the remaining states are the training set. With the exception of the logistic regression and random forest classifier, each of the models produced similar results. Precision represents the percentage of predictions that are correct. Recall is the percent of positive cases that are identified. The F1 score is the percent of positive predictions that are correct. And support is the number of occurrences, in this case, the battleground states. Logistic regression can be eliminated, but evaluating the remaining models for the final section requires another method since there is variability with each classification report.

Table 1 - Classification Report

Weighted Avg	Precision	Recall	F1-score	Support
Logistic Regression	.53	.73	.61	11
Decision Tree	.86	.73	.74	11
Random Forest	1	1	1	11
Gradient Boost	.86	.73	.74	11
XGBoost	.86	.73	.74	11

K-Folds Cross Validation

The K-Folds method splits the data set into random K folds. The model can then be fit at each iteration and the total folds can be evaluated with an accuracy score. [insert table of k-folds cross validation].

Table 2 - K-Folds Cross Validation Score

	Score
Decision Tree Classifier	.83
Random Forest Classifier	.88
Gradient Boosting	.81
XGBoost	.91

While XGBoost scores slightly higher than Random Forest, the latter performed better with other evaluation models, such as confidence interval bootstrapping, I therefore selected Random Forest for further evaluation.

Selected Model: Random Forest Classifier

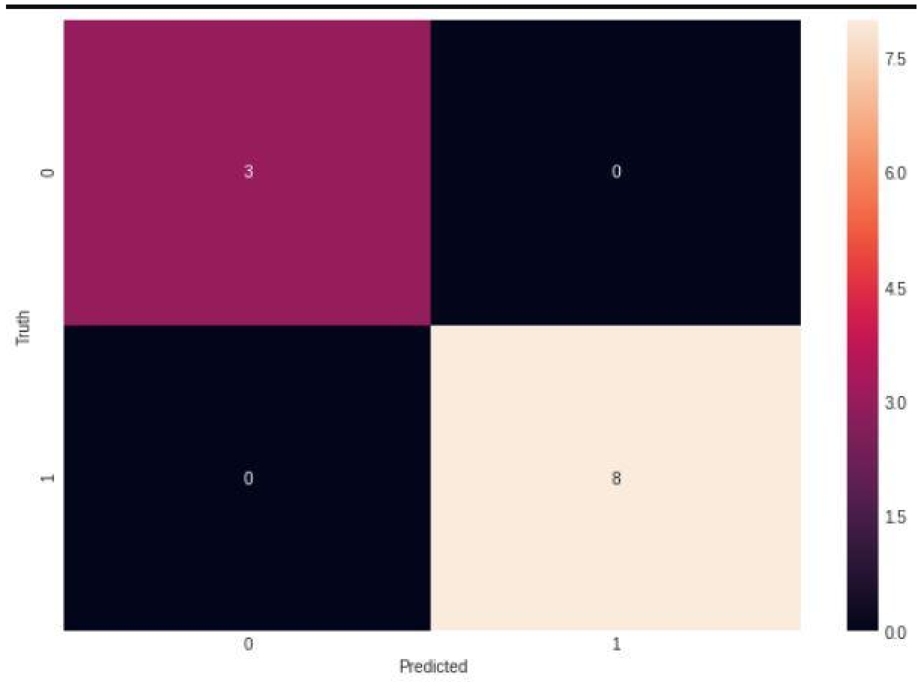


Figure 1: Random Forest Confusion Matrix

I used 100 n-estimators for the Random Forest Classifier. The accuracy score was 100% and the confusion matrix in Fig. 1 shows the model predicted zero false positive and zero false negatives. The random forest function feature_importances shows in Figure 2 that the polling data having greatest significance as a predictor , as well as the cook partisan index and some demographic variables also contributing.

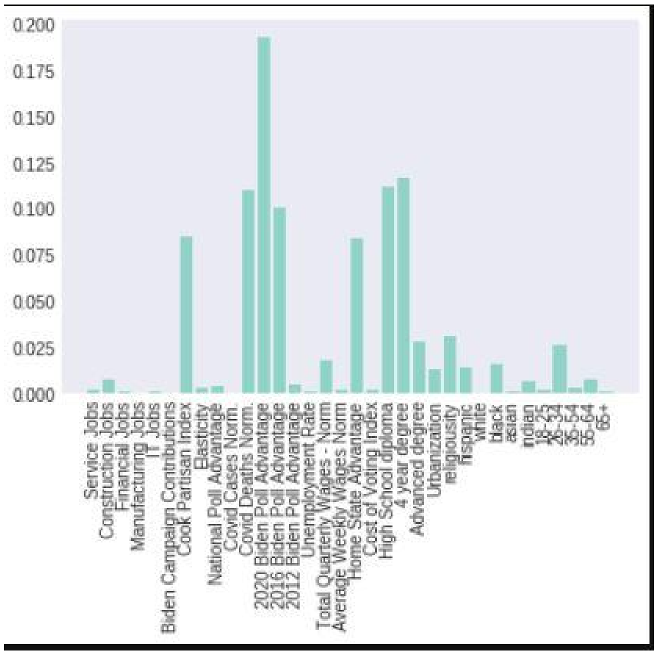


Figure 2: Feature Importance using scikit-learn

SHAP Feature Importance

SHAP is a module for python that uses game theory to explain the output of models The Shapley are calculated by finding the mean among all permutations of the marginal features. SHAP values are useful in that they can show the positive or negative relationship for each feature. Figure 3 shows the overall SHAP values for the test states. Additionally, we can view the set of SHAP values for each state's outcome using force_plot, revealing potentially different feature importances depending on the state, shown in Figure 4. SHAP can also be applied to different kinds of models, in this instance I used the TreeExplainer() method.

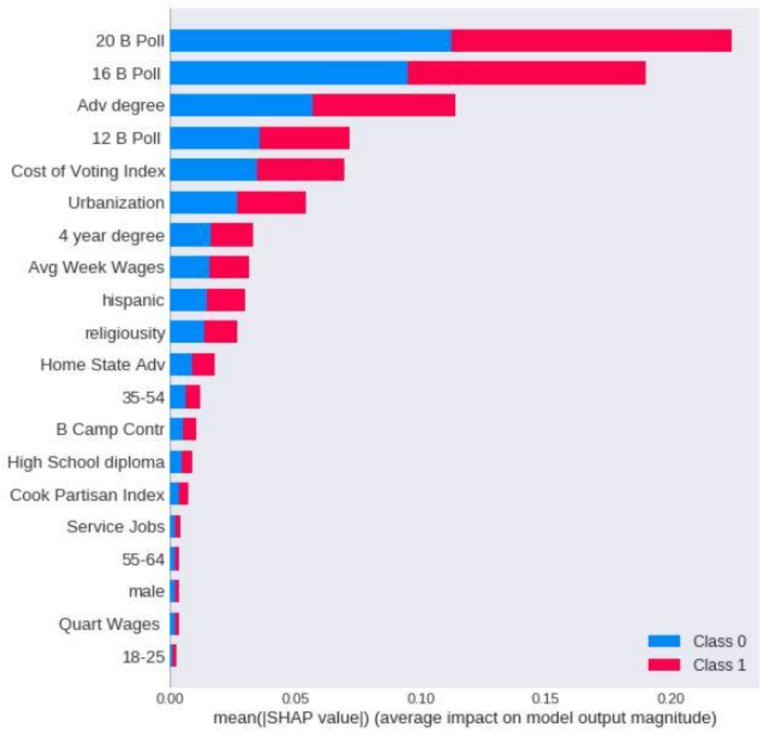
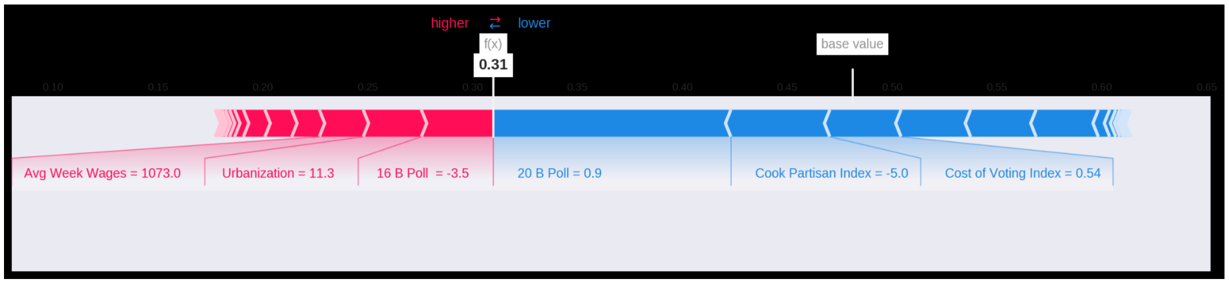


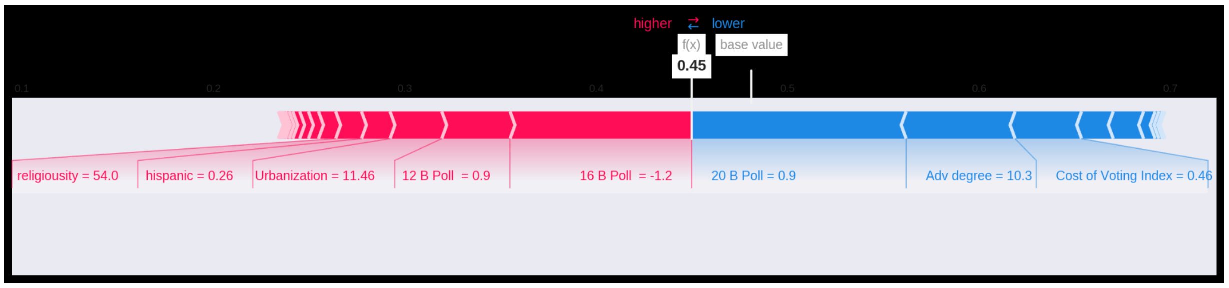
Figure 3 - Test State SHAP Values

Figure 4 - Test state force plots

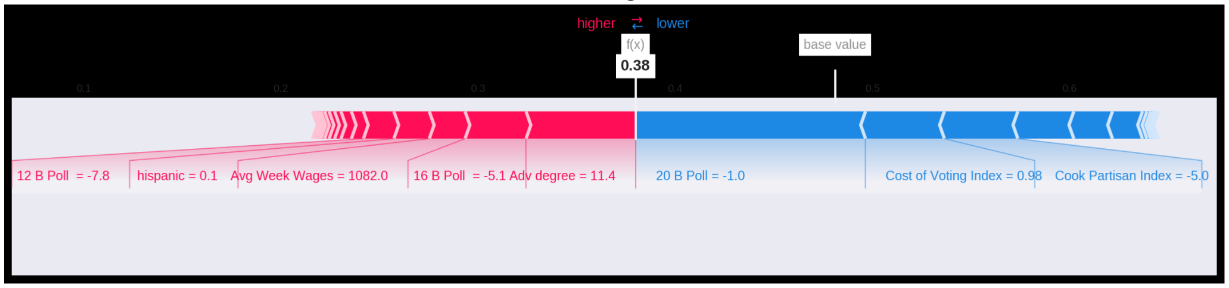
Arizona



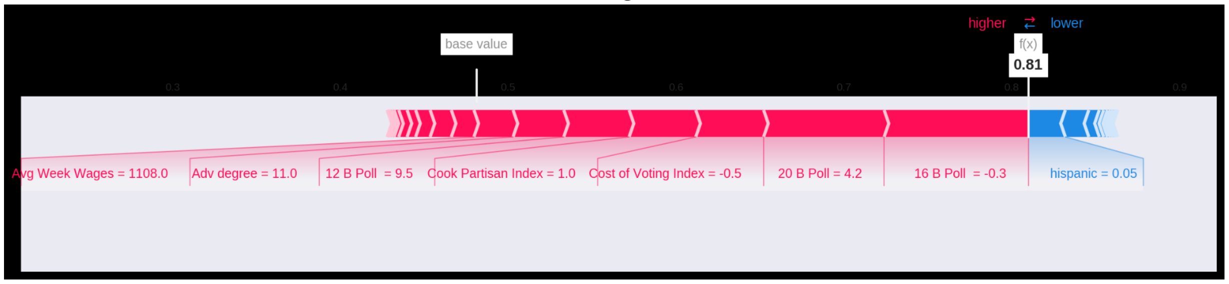
Florida



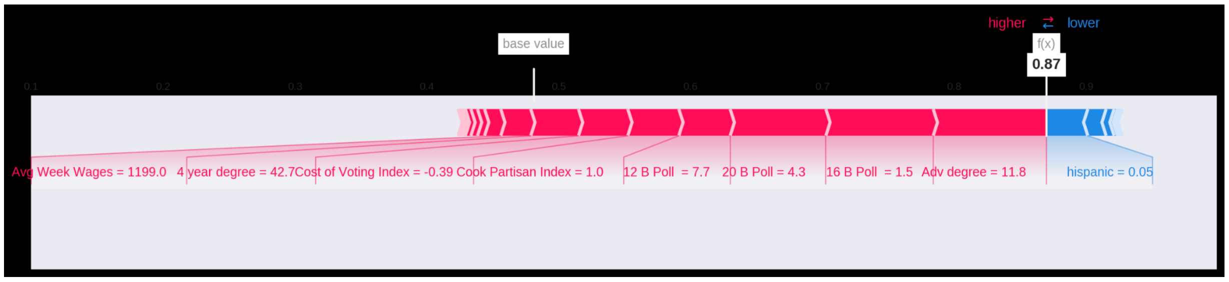
Georgia



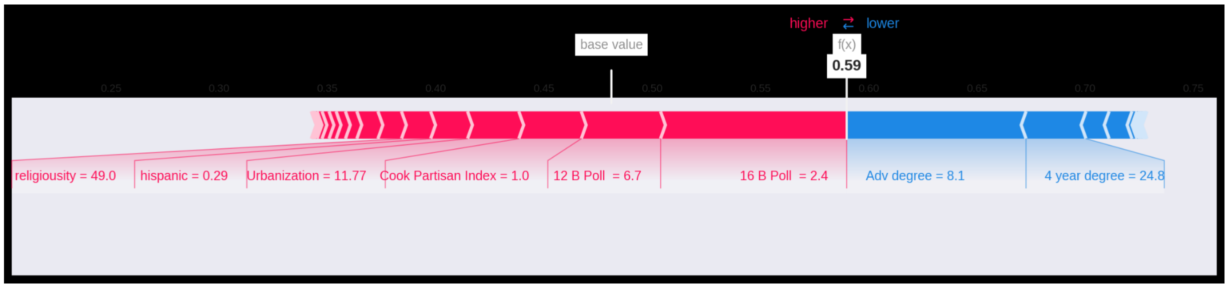
Michigan



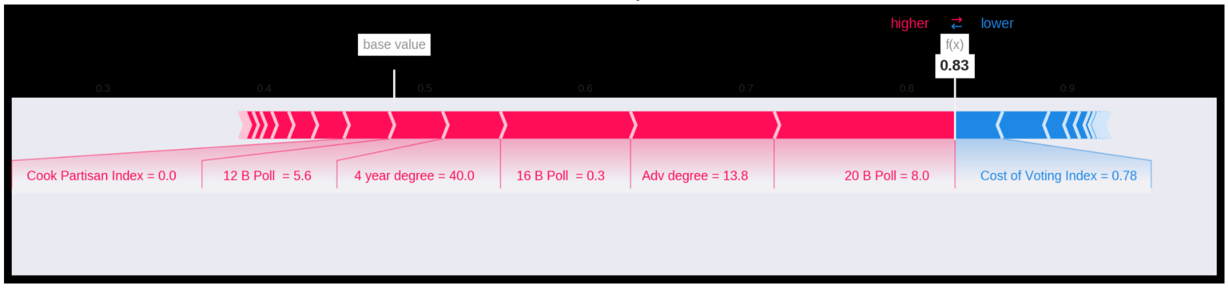
Minnesota



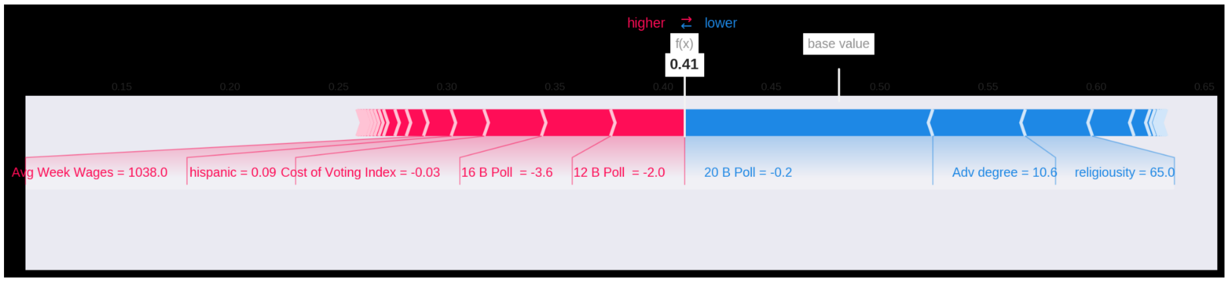
Nevada



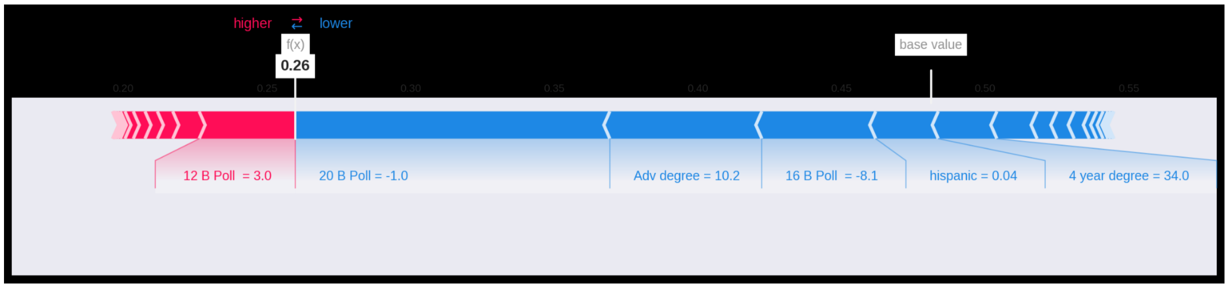
New Hampshire



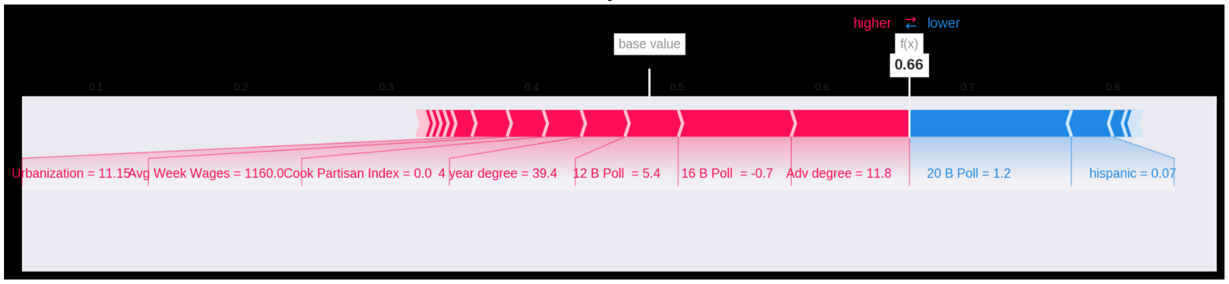
North Carolina

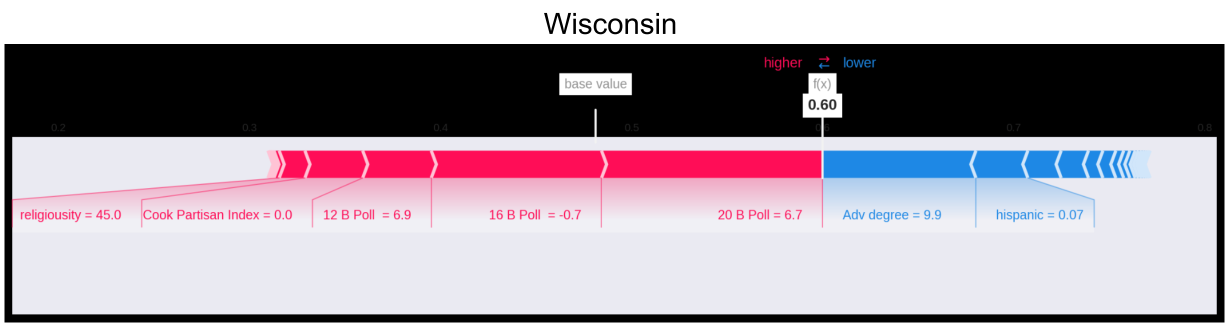


Ohio



Pennsylvania

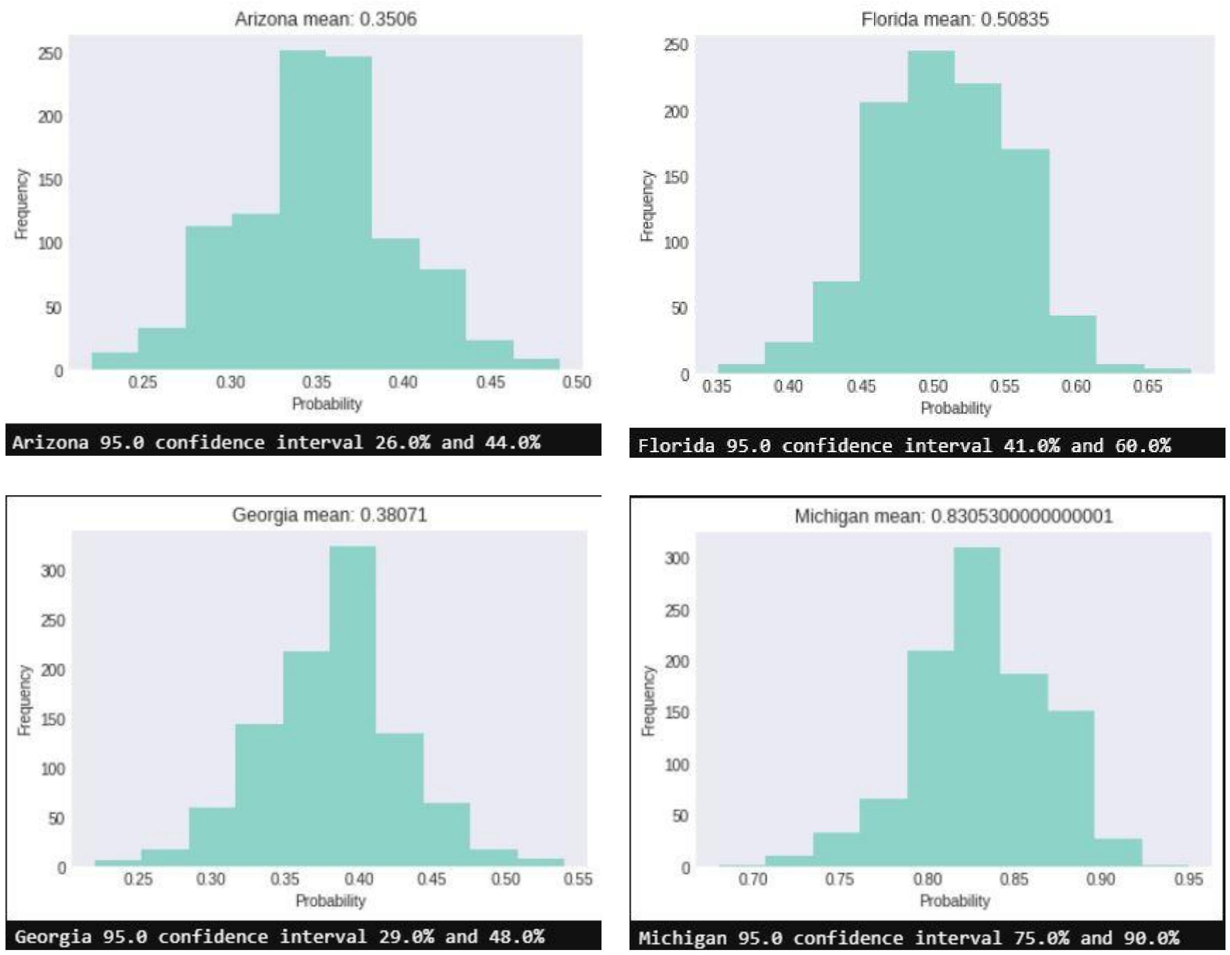




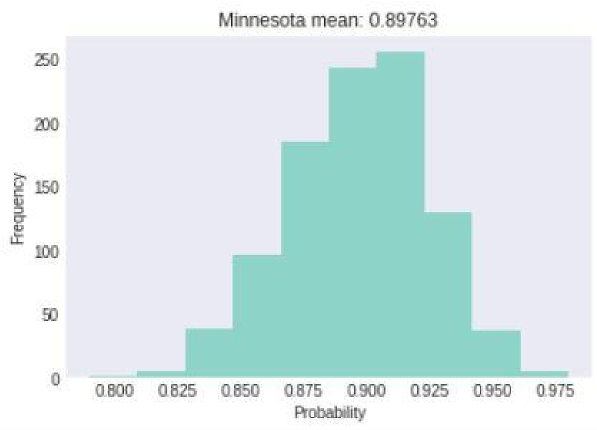
Unfortunately, the color options for individual observations are not currently functioning, per the module's github requests, so we must interpret the red and blue differently than our typical associations with the Republican red and Democrat blue. In this case, the red colors forcing the higher shap values indicate features that are higher predictors for a Biden win. Conversely, blue colors that force lower shap values indicate feature predictors for a Biden loss. I abbreviated some names for display purposes. For example, Biden's 2020 Poll advantage is abbreviated to 20 B Poll. Unsurprisingly, the features with the most influence are polling numbers, though Advanced degrees and Urbanization had a showing as well.

Finally, the objective of this project is to input a state and output a probability that a candidate will win that particular state's electoral votes. Even with the n_estimators argument for fitting the random forest classifier, merely training the set once seems insufficient as there is variance after several successive fits. I wrote a bootstrap function to fit the model over n_iterations and append a list of predict_proba values. Then I found the average of those values, representing our final probability of victory with a normal distribution graph and confidence interval. 1000 iterations were used, which required about 20 minutes to run.The margin of error of the probability distribution was at minimum 5 points, at maximum 10 points, depending on the state.

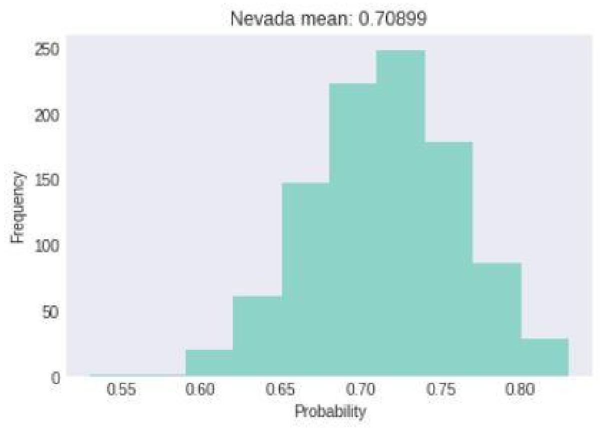
Figure 5: Confidence intervals and predicted probability distribution by state.



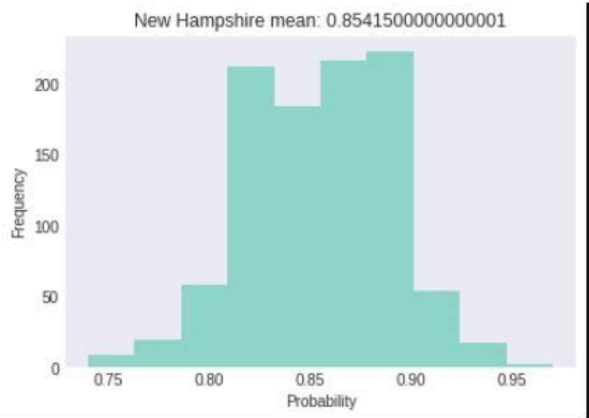
```
Arizona 95.0 confidence interval 26.0% and 44.0%
Florida 95.0 confidence interval 41.0% and 60.0%
Georgia 95.0 confidence interval 29.0% and 48.0%
Michigan 95.0 confidence interval 75.0% and 90.0%
```



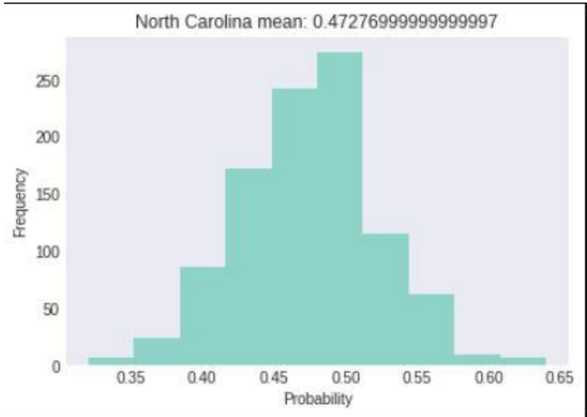
```
Minnesota 95.0 confidence interval 84.0% and 95.0%
```



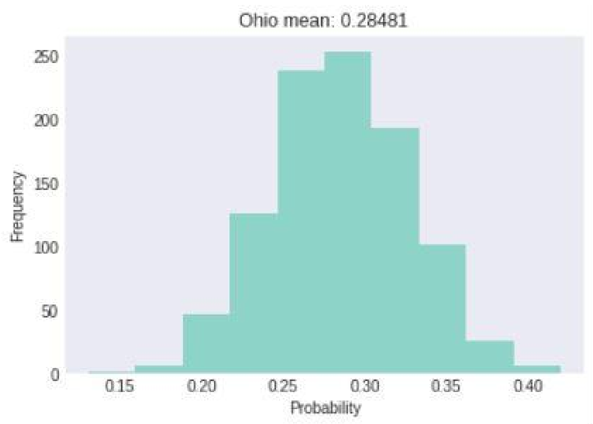
```
Nevada 95.0 confidence interval 62.0% and 80.0%
```



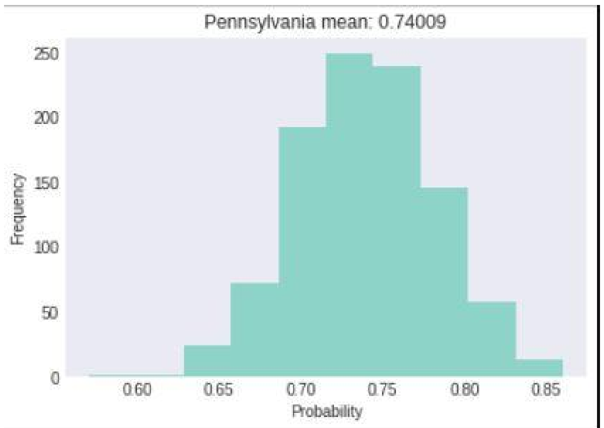
```
New Hampshire 95.0 confidence interval 78.0% and 92.0%
```



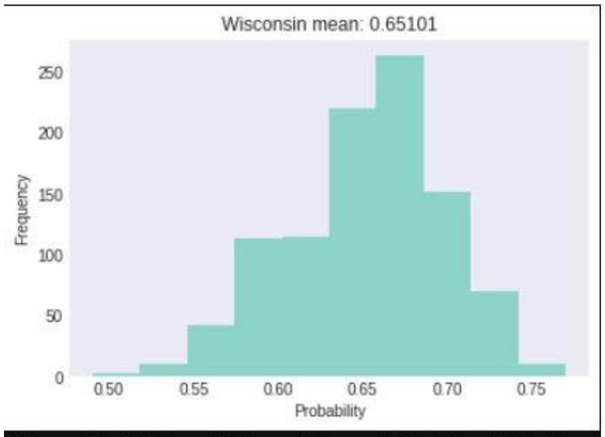
```
North Carolina 95.0 confidence interval 38.0% and 57.0%
```



```
Ohio 95.0 confidence interval 20.0% and 37.0%
```



```
Pennsylvania 95.0 confidence interval 65.0% and 83.0%
```



```
Wisconsin 95.0 confidence interval 56.0% and 74.0%
```

CHAPTER FOUR

DISCUSSION OF MODEL RESULTS AND MODEL COMPARISON

Clearly, the model was consistently wrong about two states: Arizona and Georgia. Biden won the first two states while the random forest model predicted the opposite. With Georgia, Biden did not have a lead and past elections did not favor the democrat. The model showed Biden having a coin toss chance in Florida. Arizona is more peculiar. The republican candidates had the advantage in 2012 and 2016. In 2020, however, Biden had the polling advantage, though not by a full point. Additionally, the 2020 feature of his positive advantage influenced the model's prediction that he would lose! I double checked the data to make sure the values were being populated from the correct row and found no issues. And it turns out that this wasn't the only state to display the phenomena of a positive poll advantage working against his favor in the model's prediction per the SHAP features. Of course, these features are only being populated on one fit. At this time, there is no way to extract a specific shapley value per observation in order to run a bootstrap random sampling similar to producing the confidence intervals. In any case, it's difficult to ascertain why random forest's distribution for Arizona does not capture a potential win for Biden, though it may reveal a potential fundamental flaw in the model's use of this data. Finally, it's worthwhile to compare the results of this model to more sophisticated models produced by FiveThirtyEight and The Economist.

Table 3: Biden's chances of winning each battleground state by model

	Random Forest Classifier	FiveThirtyEight	The Economist	Result (lead difference)
Arizona	35%	68%	73%	.3%
Florida	50%	69%	80%	-3.3%
Georgia	38%	58%	58%	.23%
Michigan	83%	95%	98%	2.8%
Nevada	70%	88%	95%	2.4%
New Hampshire	89%	89%	99%	6.9%
North Carolina	47%	64%	70%	-1.3%
Ohio	28%	45%	44%	-8.1%
Pennsylvania	74%	84%	95%	1.2%
Wisconsin	65%	94%	98%	.63%

(FiveThirtyEight, 2020) (The Economist, 2020)

While FiveThirtyEight and The Economist correctly predicted Georgia and Arizona, they incorrectly predicted North Carolina and Florida. It should be noted that a probability about a binary win/loss is not the same predictor as the share of the vote used to calculate the lead difference. However that doesn't mean nothing can be said about the bigger discrepancies. While the Random Forest Classifier did not accurately predict Arizona, the actual Biden lead in the state was less than half a percentage point. Additionally, Biden's very narrow lead in Wisconsin should make observers rethink their confidence in FiveThirtyEight and The Economist's predictions of 94% and 98%, respectively, that Biden would win. Additionally, the media's prediction that Ohio was a coin toss doesn't seem consistent with Trump's landslide victory. Pennsylvania and Wisconsin each enjoyed 94% chances in favor of Biden but each of their lead differences were within the margin of error. Clearly, Trump's chances were being underestimated by polling and that was not captured in the media's models.

The Random Forest Classifier shared an accuracy score with the other models: 81%. The Random Forest Classifier notably did not overestimate Biden's chances and the states that it was incorrect about ultimately voted for Biden. They're states which are also less consequential in terms of number of electoral votes (270 to win) than North Carolina and Florida. With the assumption that the remaining non-battleground states voted consistently with history (red states voted red and blue states voted blue), the Random Forest Classifier would forecast that Biden would be the ultimate victor with 279 electoral votes.

In a write up on how FiveThirtyEight's model works, Nate Silver explains that one of it's primary mechanisms is to calculate uncertainty based on the idea that the polling is incorrect. However, in a 2020 retrospective, Silver's tendency is to downplay the model's uncertainty algorithm and attribute the underestimating of Trump's chances to polling errors. (Silver, 2020) Since the code for the FiveThirtyEight model is proprietary, we can't view how exactly the model works and how it handles uncertainty. Thus, it's impossible to make any serious academic comparisons between that model and others.

While outside the scope of this project, future work should explore how to anticipate uncertainty and neural network models could be trained on the data. Similar to corporate models, the project's model could output an updated probability multiple times a day based on the polling averages up until that moment. The model could also be trained on other economic and demographic data not used here. Finally, the model can be tested during the 2024 election cycle.

CHAPTER FIVE

CONCLUSION

This project set out to develop an election forecasting model that predicts a probability that a candidate will win a US battleground state and to determine which features are most significant in those predictions. The Random Forest Classifier model was trained only on non-battleground states so as to reduce the chance of overfitting. The model shares an accuracy score in line with corporate media forecasts, although there is a difference in which states the models were incorrect about. Additionally, we learn that that model with the applied data set does not overestimate Biden’s chances, which was a mistake corporate media made, to great criticism in 2016 and also in 2020, to less. Without knowing exactly which data would be most important, the model trained on somewhat arbitrarily chosen 34 features that included polling, economic, and demographic data and past electoral margins. Unsurprisingly, current polling and previous electoral margins had consistently higher SHAP values, although Advanced Degrees in bluer battleground states were an unexpected high value feature.

References

Bibliography

The Economist. (2020, November 3). *Forecasting the US elections*. The Economist. https://projects.economist.com/us-2020-forecast/president

FiveThirtyEight. (2020). *2020 Election Forecast*. FiveThirtyEight. https://projects.fivethirtyeight.com/2020-election-forecast/

Maklin, C. (2019, July 27). *Decision Tree in Python*. Towards Data Science. https://towardsdatascience.com/decision-tree-in-python-b433ae57fb93

Navlani, A. (2019, December 16). *Understanding Logistic Regression in Python*. Datacamp. https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

Nelson, D. (n.d.). *Gradient Boosting Classifiers in Python with Scikit-Learn*. Stack Abuse. https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/#:~:text=Gradient%20boosting%20classifiers%20are%20a%20group%20of%20machine,trees%20are%20usually%20used%20when%20doing%20gradient%20boosting

Politico. (2016, November 5). *Nate Silver rages at Huffington Post editor in 14-part tweetstorm*. Politico. https://www.politico.com/story/2016/11/nate-silver-huffington-post-polls-twitter-230815

Silver, N. (2020, November 11). *The Polls Weren't Great. But That's Pretty Normal.* FiveThirtyEight. https://fivethirtyeight.com/features/the-polls-werent-great-but-thats-pretty-normal/

Stojiljković, M. (2020, January 13). *Logistic Regression in Python*. Real Python. https://realpython.com/logistic-regression-python/

Yiu, T. (2019, July 21). *Understanding Random Forest*. Towards Data Science. https://towardsdatascience.com/understanding-random-forest-58381e0602d2