Utilization of Machine Learning to Simulate the Implementation of Instant Runoff Voting

Nicholas J. Joyner
Faculty advisor: Michele L. Joyner

Department of Mathematics & Statistics
East Tennessee State University, Johnson City, TN

Abstract

In election years when the popular vote winner and Electoral College winner differ, such as in the 2016 presidential election, there tends to be an increase in discussions about alternative voting strategies. Ranked choice voting is a strategy that has been discussed and is currently used in approximately fourteen cities across the United States and in six states for special elections and overseas ballots. Ranked choice voting (RCV), sometimes called Instant Runoff Voting (IRV), is a system of voting in which voters are allowed to rank the candidates. If no candidate wins over fifty percent of the vote, the election automatically goes to another round. The candidate with the least support is eliminated and their votes are redistributed to the voters' next choice. This process of elimination and redistribution continues until a candidate receives a majority of the vote. In this paper, we use predictive modeling strategies and simulation to investigate the potential implications of employing ranked choice voting in a presidential election using the 2016 presidential election as a case study.

1 Introduction

Ranked choice voting is a voting method in which the voters are allowed to rank the candidates in their order of preference [1, 2]. In broad terms, ranked choice voting encompasses an assortment of voting strategies such as instant runoff voting (IRV), single transferable vote (STV), and the alternative vote (AV) in Australia [3]. While the voting system currently used in most of the United States requires a simple plurality of the votes to win, ranked choice voting requires a candidate to have a majority of the remaining votes, i.e. over 50% of the votes. The different types of ranked choice voting attempt to achieve a majority vote winner using different strategies. We will focus on the instant runoff voting (IRV), a strategy used when electing a single candidate and often referred to simply as ranked choice voting. If a candidate has not received a majority of the first choice votes in the initial vote count, the candidate with the fewest votes is eliminated and their votes are redistributed to the voters' next choice. Once the votes have been redistributed, the vote totals are revised. This process of elimination, redistribution, and revision repeats until a candidate has a majority of the votes; see Figure 1.

Ranked choice voting, in general, is used in approximately fourteen cities across the United States and in six states for special elections and overseas ballots [4]. It is also used in a variety of other countries in various elections. Supporters of ranked choice voting claim that it is more democratic in that it requires the winner to have majority support [5, 6, 7]. Advocates of ranked choice voting also claim that it removes the need for expensive runoff elections, because ranked choice voting already ensures a candidate wins the majority of the vote using only one ballot.

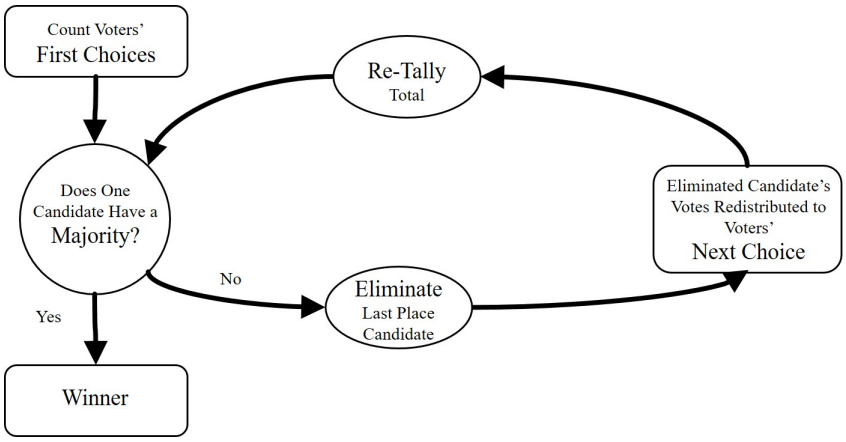

Figure 1: This figure shows the process by which a candidate is elected using instant runoff voting.

Opponents argue that many localities do not have the equipment to implement ranked choice voting and that it is too confusing, pointing to above average rates of uncountable ballots due to ballot errors [8, 9]. Moreover, the constitutionality of the implementation of ranked choice voting in federal elections has been a mainstream news story in Maine in which Maine's Supreme Judicial Court ruled on April 17, 2018 that it was indeed constitutional, ending the lengthy debate [10, 11]. In addition, it is not clear that the winning candidate will have a majority of all the votes cast [5]. When the candidate with the lowest number of votes is eliminated, those votes are transferred to the voter's next choice; however, for all ballots in which there is not a 'next choice' candidate, the votes are eliminated or "exhausted". Thus, the winner has a majority of those votes in the final round, which may or may not be a majority of all votes cast.

This case was illustrated in Maine's 2018 House of Representative's Election for the 2nd congressional district in which almost 3% of the original votes were exhausted in the final round (see Table 1) leaving the democratic candidate winning the election with only 49.2% of the original votes (`https://www.maine.gov/sos/cec/elec/results/results18.html#Nov6`). Moreover, as indicated in Table 1, in the initial round of votes, the Republican candidate was the original winner with 46.3% of the votes compared to 45.6% of the votes for the Democratic candidate. However, the ultimate winner was the Democratic candidate with 50.6% of the remaining votes (49.2% of the original votes) compared to the Republican candidate with 49.4% of the remaining votes (47.8% of the original votes). Similar outcomes will be shown in the simulations presented in future sections of this paper.

In this paper, we use mathematical techniques, in particular, predictive modeling and simulation, to analyze instant runoff voting from an alternate viewpoint, focusing on the potential results in an election if instant runoff voting were implemented. We use, as a case study, the 2016 presidential election. We examine these effects by considering multiple different scenarios. In Section 2, we overview the data utilized in this paper. Section 3 focuses on an overview of predictive modeling using decision trees. We then detail the methodology in Section 4 and results of the implementation in Section 5 under multiple different scenarios. We conclude in Section 6 with some final remarks and directions for future work. In the process of implementing the predictive modeling strategies, we additionally focus on the political ideologies predictive of the voting behaviors of different groups of voters.

Table 1: Implementation of IRV in Maine's 2nd Congressional District Election, November 2018*

Party	Round 1			Round 2		
	Votes	%	Transfer Votes	Votes	%	
Democratic	132,013	45.6%	+10,427	142,440	49.2%	50.6%
Republican	134,184	46.3%	+4,747	138,931	48.0%	49.4%
Independent	16,552	5.7%	-16,552	Eliminated		
Independent	6,875	2.4%	-6,875	Eliminated		
Total Active Votes	289,624	100%		281,371		100%
Exhausted Votes			+8,253	8,253	2.9%	
Total Votes	289,624	100%		289,624	100%	

*https://www.maine.gov/sos/cec/elec/results/results18.html#Nov6

2 Cooperative Congressional Election Study and Electoral College

In the modeling process, we use two main sources of data, the Cooperative Congressional Election Study, or CCES, and the official presidential election results. The CCES is used to create the predictive models necessary in the redistribution of votes in the IRV elimination process while the official election results are used as initial conditions for this process.

2.1 Official Election Results

The official vote counts were sourced from federal (https://www.fec.gov/pubrec/fe2016/2016presgeresults.pdf) and state governments (Maine and Nebraska) and are used to initialize the first choice votes (first step of the IRV process as displayed in Figure 1). These official results include major party, third party, and write-in candidates for each state. We are only considering five candidates: Donald Trump (Republican), Hillary Clinton (Democrat), Gary Johnson (Libertarian), Jill Stein (Green), and Evan McMullin (Independent). These candidates are the only ones considered, because they are the only candidates expressly asked about in the CCES survey (discussed in Section 2.2). Although these five candidates are considered, votes for these candidates are only considered if the candidate was officially on the state ballot.

The Electoral College is the process by which a president is elected. It is comprised of electors from all 50 states, with each state having a number of electors approximately proportional to the population of the state; Figure 2 shows the number of electoral votes for each state. Each elector has one vote and there are 538 electors total. A presidential candidate needs to receive a majority of the electoral votes (270) to win the presidency. Most states assign all of their electoral votes to the winner of the popular vote in the state. Maine and Nebraska, however, assign their electoral votes by congressional district. This results in the electoral votes for Maine and Nebraska being split, one for the winner of each congressional district and two for the winner of the statewide popular vote. This means that Maine and Nebraska can split their electoral votes between multiple candidates. This happened in Maine in the 2016 presidential election when Donald Trump received one electoral vote from Maine for winning Maine's 2nd congressional district while Hillary Clinton received the other three electoral votes, one for winning Maine's 1st congressional district and two for winning the statewide popular vote. As a result of this split, we needed to use Maine's (http://www.state.me.us/sos/cec/elec/results/results16-17.html#tally) and Nebraska's (http://electionresults.sos.ne.gov/resultsSW.aspx?text=Race&type=PC&map=CTY) state level official results, because they provide the official results by congressional district. This allowed us to independently simulate the election in each of these congressional districts.

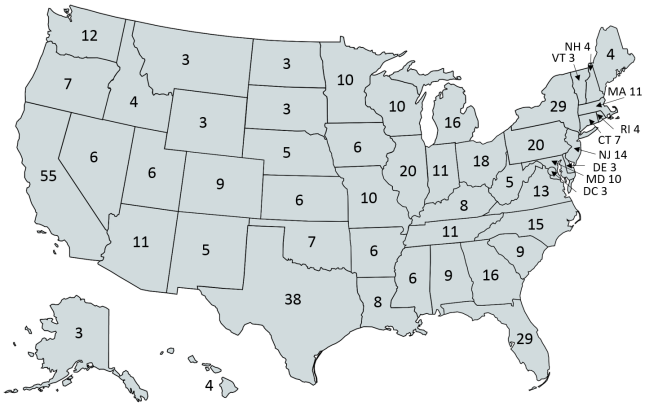

Figure 2: This figure shows the number of electoral votes per state in 2016. (Created with mapchart.net and PowerPoint.)

2.2 Cooperative Congressional Election Study

To simulate the use of IRV in the presidential election, we, in fact, simulate the implementation of "partial-ranked" ballots, because only 5% of the votes were cast for third party candidates, with the rest allocated to either Trump or Clinton (see Figure 4 in Section 4). Therefore, in simulating the use of the IRV process in this instance, the third-party candidates will always be eliminated first. We must thus consider to whom these third-party votes would be allocated; i.e., which candidate might have been ranked second on the ballot. Using machine learning to predict this second choice, it is necessary to understand the ideological factors behind why a person supports a particular candidate, other than partisanship. The Cooperative Congressional Election Study (`https://cces.gov.harvard.edu`) is a good source of information on both ideologies and voter choice.

The CCES is a national survey administered by YouGov/Polimetrix in which over 50,000 respondents are selected using a stratified sampling technique. The survey has been administered every year since 2005 except in years 2013 and 2015. In election years, the survey consists of two waves, a pre-election survey administered from late September to late October and a post-election survey administered in November. The pre-election survey includes questions pertaining to voter demographics as well as voter ideology. The post-election survey includes a few demographic questions with the majority of questions pertaining to voting experiences and rationale. The CCES is often used to make predictions and to evaluate election outcomes, but also to determine the political tendencies of different subsets of the population.

The pre-election survey for the 2016 election was used extensively in the creation of our models, as well as a handful of questions from the post-election survey [12]. Although the survey consisted of over 100 questions, some of the questions our model found to be most predictive are voter party identification (Democrat, Republican, or neither) and strength of party identification (strong or not very strong), political viewpoint (liberal, moderate, or conservative), and voter opinion of the major candidates as well as the Obama administration. Other questions which appeared occasionally in the modeling process were those pertaining to one's religious affiliation as well as one's stance on certain policy issues, e.g., one's stance towards the Affordable Care Act. Specifics about how these questions are used can be found in Section 5.

3 Predictive Modeling using Decision Trees

In simulating the effects of implementing instant runoff voting in the 2016 presidential election, we used predictive modeling to determine how the votes would be redistributed when a candidate is eliminated. This is a novel approach as we are using machine learning to gain insight into electoral information that isn't already captured explicitly in the data. Since there have not been a lot of studies in the third party voters' ideologies and how they relate to the major party candidates, we use predictive modeling as a means for estimating for which of the two major party candidates a third party voter might have voted as their second choice. We note that this methodology can be used to address other alternative voting methods or to gain insights into potential political driving factors for groups of voters.

3.1 Overview of Decision Trees

In general, predictive modeling can be thought of as the process of developing a mathematical model or tool which can then be used to generate the probability of an outcome [13]. One type of predictive model is a decision tree which is a classification algorithm. In its simplest form, it can be thought of as nodes, branches and leaves with the leaves being the predicted category (for our purposes, this is the predicted political party or candidate) and the branches being nested if/then statements between nodes. An example of a decision tree is shown in Figure 3 in which the predicted category was a vote for either Clinton or Trump.

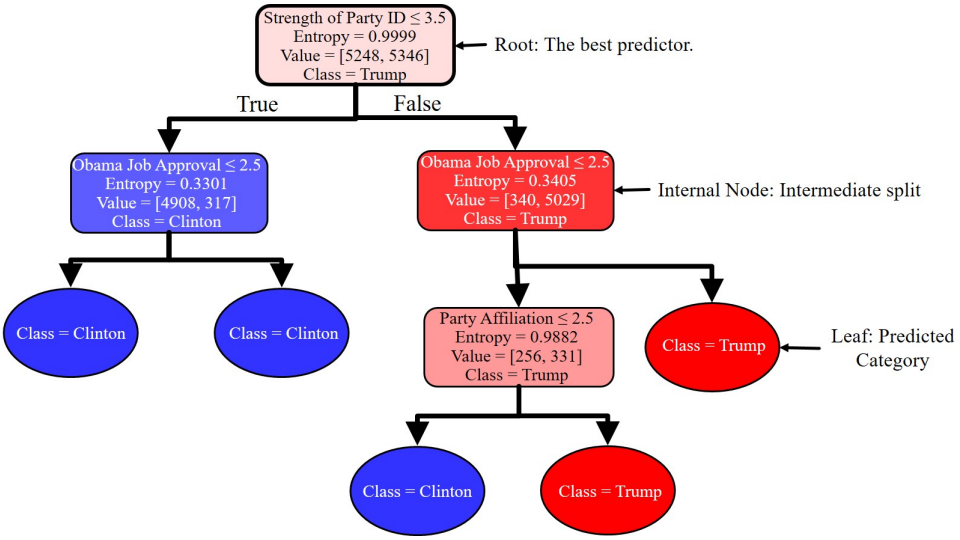

Figure 3: This figure shows a sample decision tree which could be used to predict whether a voter would vote for either Clinton or Trump. This tree is used for illustrative purposes to discuss the terminology associated with decision trees.

At the top of the tree is the root node which is considered the best predictor, i.e., the feature which best splits the target class into the purest possible children nodes. To determine the root node, the algorithm searches through all possible variables in the data to determine which variable gives approximately a 50/50

split across that node. For example, in Figure 3, there are 10,594 survey respondents used to create the decision tree. By splitting on the strength of their party affiliation at a value of 3.5, there were 5248 respondents (49.5%) with a score less than 3.5, i.e., these respondents were leaning democratic, and 5346 respondents (50.5%) with a score greater than 3.5, i.e., these respondents were leaning republican. The almost 50/50 split is reflected in the associated entropy value of approximately 1.

Entropy can be thought of as a measure of randomness, uncertainty or impurity, and, for a discrete random variable X, can be calculated using the formula

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{1}$$

where n is the number of classes and p_i is the probability of class i. For example, the entropy of the root node in the sample decision tree is calculated as

$$H(X) = -\frac{5248}{10594}\log_2\left(\frac{5248}{10594}\right) - \frac{5346}{10594}\log_2\left(\frac{5346}{10594}\right) = 0.9999.$$

In general, if the entropy is 1, then the sample is equally divided with maximum uncertainty; whereas if the entropy is 0, all targets are in the same class with absolute purity.

Once the root node is identified, the predictors for the internal nodes or intermediate splits are determined according to which attributes result in the highest information gain. Information gain is defined as the difference in the entropy before and after the split. To calculate the entropy after the split, a weighted average of the entropies for the left and right splits is calculated where the weights are determined by how many instances went along the left versus right branch. For instance, in Figure 3 the first intermediate split along both branches is Obama's job approval. Of the 10,594 entries, 5225 went left and 5369 went right; therefore, the left and right entropies are calculated using Equation (1) and then are combined to obtain the entropy after the split,

$$H_{after}(X) = \frac{5225}{10594}(0.3301) + \frac{5369}{10594}(0.3405) = 0.3354,$$

which results in an information gain of about 0.66. This process of splitting continues until a leaf is reached where the leaf is the predicted category of the result.

This process of splitting can continue until the decision tree provides 100% accuracy for the data set from which it has learned. Depending on the number of variables in the dataset, the tree can be extremely deep with potentially hundreds of splits. However, this typically results in the 'overfitting' of the data. In other words, the model obtained has not only learned the major trends in the data but also the noise unique to the original data set. This results in inaccurate predictions when trying to predict a case not found in the original data. To control for overfitting, it is necessary to prune the tree, i.e. use a tree with fewer splits. The number of splits necessary is often obtained by training the decision tree on a random portion of the data (called the training set) and then testing the model on the remainder of the data (the testing set); in most cases in this paper, we train on 2/3 of the data and use 1/3 for testing. If the training score is 'good' (dependent on the problem one is modeling) and the ratio of the training and testing score is close to 1, then the tree has been pruned well without overfitting.

3.2 Random Forest

One of the advantages of a single decision tree is its interpretability. For our purposes, in some cases, we discuss a single decision tree which gives us additional insights into voter preferences and the political leanings

of different subsets of voters. However, decision trees suffer from high variance [14]. Since we randomly split the data into testing and training sets, the decision tree produced can vary depending on the portion of the data which is used for training; the resulting decision tree for one training set can be quite different than for another training set. As a result, the prediction for a case not in the data can also vary considerably depending on which decision tree is used (i.e. which training set is used).

To reduce the variance, one can use techniques such as bagging or random forests [14]. Bagging, sometimes referred to as bootstrapping or aggregation, is implemented by constructing N different decision trees using N bootstrapped training sets and then simply averaging the resulting predictions. When implementing bagging, the likelihood is that the trees will look quite similar to each other as there are usually only a handful of moderately strong predictors out of all the possible predictors. As a result, the bagged trees will likely be highly correlated. Averaging highly correlated quantities does not reduce the variance greatly.

Random forests is similar to bagging in that a number of decision trees are built on bootstrapped training samples; however, the advantage of random forests over bagging is that the trees created are uncorrelated [14]. Random forests overcomes this problem by only considering a subset of randomly chosen predictors for each split. In other words, when constructing a typical decision tree, all possible predictors are considered at each split. In random forests, for each of the decision trees, the set of possible predictors is restricted to say d randomly chosen predictors, and then the best possible predictor out of this subset is chosen according to the criteria discussed above, i.e. to maximize the information gain. The idea is that a combination of weak learners will result in a strong learner [14, 15].

The process of forming trees using subsets of the predictors at each split is repeated N times to create a 'forest' of N trees. Once the collection of trees have been created, each of the N trees makes a prediction for the classification of new data. The final prediction is then given by the 'majority vote', i.e. the mode, of all the predictions from the collection of trees [14, 15]. It was shown by Oshiro et. al. [16] that N does need to be large; after reaching a certain number of trees in the forest ($N = 128$ in their studies), there was no significant gain in performance. In the remainder of the paper, we implement random forest using 128 trees in the algorithm for all predictive models.

4 Methodology

In this section, we describe in detail the methodology for simulating the implementation of instant runoff voting using machine learning strategies by using a simplified scenario with the nation's overall popular vote count. We first note that although there are dozens of candidates officially on the ballot and many more write-in candidates, as discussed above, we only consider Hillary Clinton, Donald Trump, Gary Johnson, Jill Stein, and Evan McMullin. Figure 4 shows the official initial vote counts for each of the five candidates prior to the implementation of instant runoff voting. Notice that Clinton and Trump carry the largest percentage of the votes; however, neither candidate has a majority of the votes. Therefore, we assume that third party candidates will be eliminated first and when eliminated, the next-choice vote goes to one of the major party candidates, either Donald Trump or Hillary Clinton. In other words, we are assuming a partially ranked ballot. This is a simplifying assumption as each voter could potentially rank another third party candidate as their next choice; however, we expect the results would only differ slightly as, in this scenario, all third party candidates will need to be eliminated before a candidate receives a majority vote.

This method of implementation is, in effect, more similar to the supplementary vote used in the UK in which voters mark their first and second favorite choices [17]. If there is no majority winner, all candidates are eliminated except the top two candidates. Therefore, by assuming the democratic and republican candidates are the third party voters' next choice, these votes would be used in the run-off round. To determine to which major party candidate a third party's vote would be reapportioned, we use the random forests algorithm in

Python [18] to first generate a forest of decision trees using only CCES data (discussed in Section 2.2) for those respondents who indicated they voted for Clinton or Trump in the final election. The decision tree's predicted categories are whether a voter is expected to vote for Trump or Clinton. We then assume that the predictive attributes for first-choice Clinton and Trump voters are the same for second-choice Clinton and Trump voters and that the survey respondents are representative of the voters nationwide. To reapportion the votes, we isolate each third party candidate's survey respondents and run their responses through the model to determine the percent of third party votes likely to go to either Trump or Clinton (we assume all votes are reapportioned).

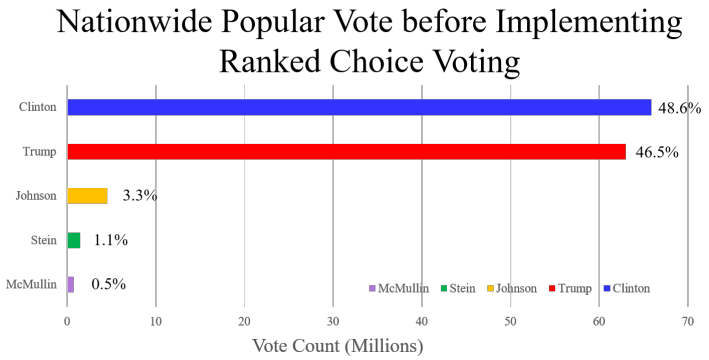

Figure 4: This figure shows the initial popular vote totals (`https://www.fec.gov/pubrec/fe2016/2016presgeresults.pdf`) and percent of the popular vote for each of the five candidates we consider (Clinton, Trump, Johnson, Stein, and McMullin).

In the implementation process, we utilized the set of weights in the survey data for each respondent. In addition, we restricted the maximum number of leaves of the decision trees to six leaves to avoid overfitting. This number was determined by considering decision trees with two through twenty-one leaves and averaging the testing and training scores across 100 different trials of random forests with 128 estimators, balanced weight for both classifications, and the entropy criterion (to indicate that predictors were chosen according to maximum information gain). With a maximum of six leaves, the average testing score was 0.9392 with a training score of 0.9396. A sample decision tree with six leaves without the additional restriction of the random forests algorithm is given in Figure 5.

In Figure 5, we can see that a voter's opinion of Obama's job performance is the most predictive factor. For people who approve of Obama's job performance, the next most important factor is their strength of party identification or how strongly the voter identifies with a certain political party. The next factor after strength of party identification is a voter's opinion of the Affordable Care Act. Although the two previous factors play a role, when restricting the model to the six leaves, our model predicts that if a voter approves of Obama's job performance, they will vote for Clinton. For the voters who respond with disapproval of Obama's job performance, the next most important factor is their opinion of Hillary Clinton. If a voter disapproves of Obama's performance as well as of Clinton, they are predicted to vote for Donald Trump. For voters who approve of Hillary Clinton, the next factor is the voter's strength of party identification. If a voter responds as leaning more Democratic, they are predicted to vote for Clinton; whereas, those who respond as leaning more Republican are predicted to vote for Trump. The decision tree in Figure 5 includes the most

predictive features across all 154 potential factors considered in the data; however, when implementing the random forest algorithm, only a random subset of all predictors is considered in the forming of individual decision trees and thus may be quite different than this sample tree.

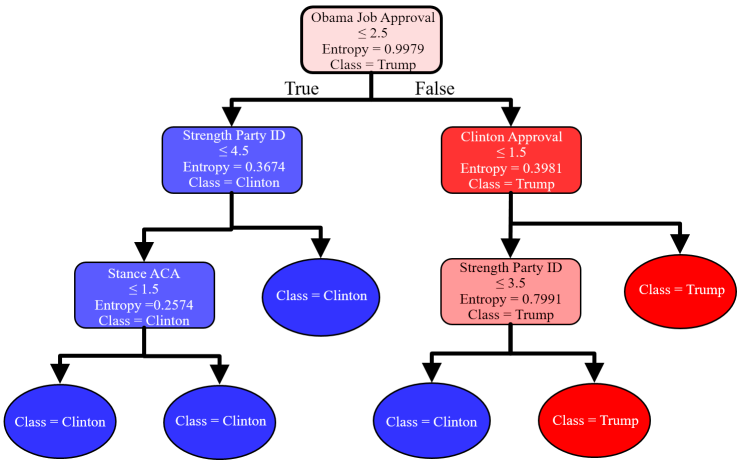

Figure 5: This figure shows a sample 6-leaf decision tree using CCES weighted data for predicting whether a voter will vote for Clinton or Trump.

Using the random forests algorithm to generate a predictive model for whether a survey respondent is predicted to vote for Clinton or Trump, we isolate all CCES respondents who indicated they voted for a particular third party candidate. We use the algorithm to predict what proportion of those respondents would have been predicted to vote for either Trump (Republican) or Clinton (Democratic). Using the model, the probabilities of third party candidates' votes to be reapportioned to each candidate are given in Table 2. The model indicates that the majority of both Johnson's and McMullin's votes will be reapportioned to Trump; whereas, Stein's votes will mainly go to Clinton. However, to account for error in the predicted probabilities, we use a randomly generated probability, ρ, for each third party vote to be reapportioned to Trump taken from a truncated normal distribution with 95% of the probabilities lying within 15% of the average probability and within the range [0,1]. We utilized a normal distribution, because it allowed us to assume symmetry about the mean while specifying the probability of having a given percentage (95% of our case) within an established distance (15%in our case) from the mean. Moreover, we truncated the distribution to assume non-negative probabilities between 0 and 1. A histogram for the probabilities for each of the candidates is given in Figure 6. The percentage of votes reapportioned to Clinton are determined by $1 - \rho$.

Table 2: Nationwide Average Predicted Probability of Third Party Voters

Third Party Candidate	Percent Predicted to Vote Republican	Percent Predicted to Vote Democratic
Gary Johnson	0.67	0.33
Jill Stein	0.21	0.79
Evan McMullin	0.88	0.12

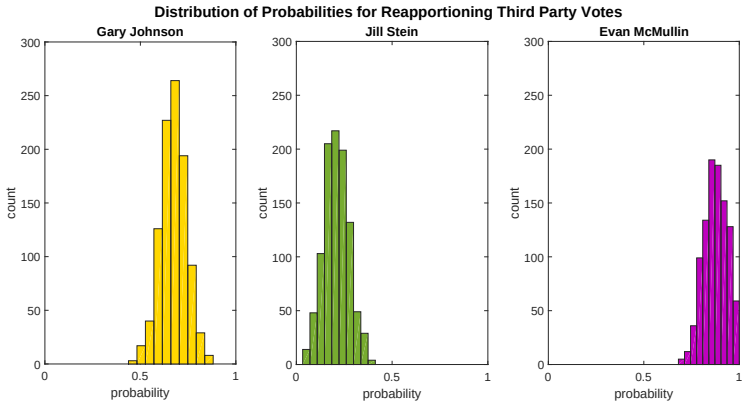

Figure 6: This figure shows the distribution of probabilities used in reapportioning third party votes to Republican candidates. It assumes mean value given in the second column in Table 2 and a standard deviation requiring 95% of all points to lie within 15% of the mean.

In each of the scenarios in Section 5 and Appendix A, we apply instant runoff voting (IRV) to the popular vote in each state using the following algorithm (see sample code in Appendix B):

Algorithm for Implementation of RCV for Popular Vote

Step 1: Initialize votes for all candidates to the official 2016 results

Set r = number of Trump's votes

Set d = number of Clinton's votes

Step 2: For each of the three third party candidates, reapportion votes to either Trump or Clinton

Step 2a: Set N = number of third party candidate's votes

Step 2b: Determine the mean probability, μ, for the candidate to vote for Trump in Table 2 (obtained from the random forests predictive model for voting for either Clinton or Trump)

Step 2c: From a normal distribution with mean μ and standard deviation 0.15/1.96 (95% of the points within 15% of μ), pick a probability ρ

Step 2d: Generate a uniformly distributed vector $\mathbf{v}$ of numbers between 0 and 1 of length N

Step 2e: Determine n_T, the amount of numbers in $\mathbf{v}$ which are less than ρ

Step 2f: Add votes to either Trump or Clinton:

Trump: $r = r + n_T$
Clinton: $d = d + (N - n_T)$

We illustrate the implementation of the algorithm using the *nationwide* popular vote as an example with one result shown in Figure 7. We start with the initial vote count as given in Figure 4 with Trump's votes initialized to approximately $r = 62.98$ million, while Clinton's votes are initialized to approximately 65.85 million. Since McMullin has the lowest percentage of votes, his votes are reapportioned first. We use the average probability of $\mu = 0.88$ generated by the model and given in Table 2 and take a random draw from the distribution shown in Figure 6 to obtain $\rho = 0.92$, i.e., 92% of McMullin's votes are reapportioned to Trump while only 8% are reapportioned to Clinton. As seen in the first bar graph in Figure 7, Clinton's overall percentage of the popular vote remains at 48.6% while Trump increases to 47% of the popular vote. Using the model generated probability for Stein (Table 2) and random draw from the truncated normal distribution ($\rho = 0.21$), Stein's votes are now reapportioned as shown in the second bar plot in Figure 7. After this round of reapportioning votes, a majority vote is still not achieved; therefore, Johnson's votes are reapportioned in a similar manner (the final bar plot in Figure 7) giving Clinton a final majority vote percentage of 50.3% to Trump's 49.7%. We repeat this 1000 times with random draws from the distributions shown in Figure 6. The results are shown in Figure 8. In all but 6 cases, Clinton wins the majority vote. In the six cases in which Trump wins, the proportion of Johnson's votes which are reapportioned to Trump are at the extreme right of the distribution for Johnson in Figure 6. We use this same methodology to examine several different scenarios in Section 5.

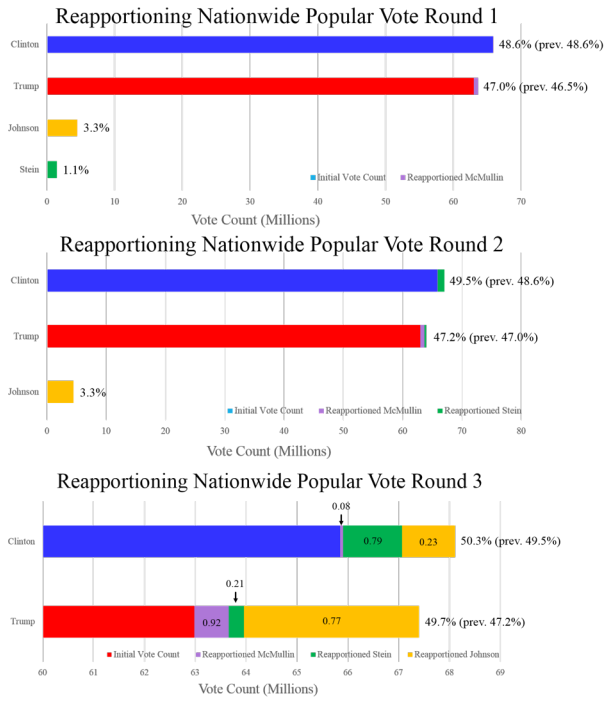

Figure 7: One example of the implementation of instant runoff voting for the popular vote.

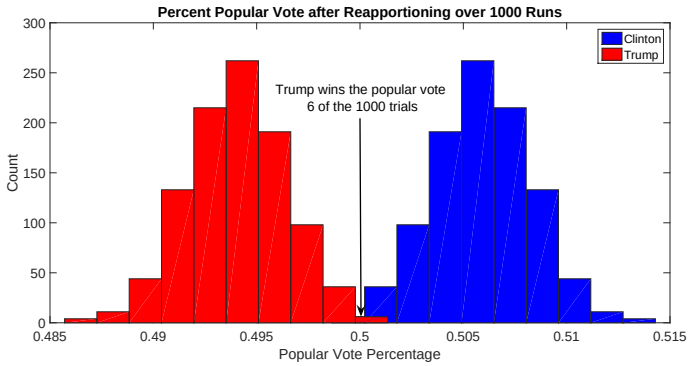

Figure 8: The result of 1000 runs of implementing instant runoff voting for the popular vote in the 2016 presidential election using random draws from the distributions given in Figure 6 for reapportioning votes.

5 Results

In this section, we use the methodology in Section 4 to consider several different scenarios when examining the potential effects that instant runoff voting might have had on the results of the 2016 presidential election. In Section 5.1, we first examine the implementation assuming instant runoff voting within individual states and assign appropriate electoral votes for the majority winner of the state. In Section 5.2, we assume that if voters had been given the opportunity to rank candidates, more voters may have voted for third party candidates initially. Therefore, in this scenario, we consider a modified initial vote count and then re-examine the analysis from Section 5.1. For interested readers, we also consider one final scenario in Appendix A where there is a strong third party candidate in the election and instant runoff voting is implemented.

5.1 Electoral College with Official Vote Counts using Ranked Choice Voting

In this section, we consider the results of the electoral college if ranked choice voting had been implemented. Recall that the only situation in which ranked choice voting would come into play would be if a majority vote were not obtained initially. In the official election results, Clinton received 227 electoral votes compared to Trump's 304 electoral votes (270 required to win the presidency), with seven electoral votes being cast for someone other than the winner of the state's popular vote. In our study, we assign all electoral votes according to the popular vote of the state. Following this method, Clinton would have received 232 electoral votes to Trump's 306 as shown in the first map in Figure 9. The second map in Figure 9 indicates that in 13 states and 1 of Nebraska's congressional districts and both of Maine's congressional districts, there was no majority winner. Therefore, these are the only states in which ranked choice voting would make a potential impact. We note that when considering only the states with a majority winner, Trump procures 213 and Clinton has 181 of the necessary 270 electoral votes. There are a total of 144 electoral votes which can be impacted by ranked choice voting.

In this section, we use the methodology described in Section 4 with one small modification. In Section 4, Table 2 gave the average probability that a third party candidate's vote would be reapportioned to either Trump or Clinton based on using survey respondents from *all* states who voted for Trump or Clinton in the final election. We assume that voters from different states may have differing political ideologies, and therefore, a decision tree representing the Trump/Clinton voters across the nation as a whole may be different than that using only Trump/Clinton voters from a specific state. Moreover, third party voters may

also behave differently politically; therefore, if there are more than 40 survey respondents within a state indicating that they voted for a particular third party candidate, we estimate a new statewide probability for the candidate. To generate the new probability, we first use random forests to create a prediction model for whether a voter will vote for Trump or Clinton using respondents from *only* that particular state and then submit the respondents from the state who voted for a third party candidate through the model to obtain a predicted category for each respondent. The mean probability for reapportioning votes to either Trump or Clinton is based on the total percentages of those votes predicted to fall into each category. The new statewide probabilities are given in Table 3. We note that although we only consider ranked choice voting in the 13 states (Figure 9), we still obtain the probability for all states for simulations discussed in the following sections.

Table 3: Average Predicted Probability of Third Party Voters by State

Third Party Candidate	State	Percent Predicted to Vote Republican	Percent Predicted to Vote Democratic
Gary Johnson	Nation	0.67	0.33
	CA	0.65	0.35
	FL	0.65	0.35
	GA	0.66	0.34
	IL	0.71	0.29
	IN	0.63	0.37
	MA	0.60	0.40
	MI	0.64	0.36
	MO	0.74	0.26
	NY	0.63	0.37
	NC	0.81	0.19
	OH	0.70	0.30
	PA	0.60	0.40
	TX	0.62	0.38
	VA	0.69	0.31
	WA	0.77	0.23
Jill Stein	Nation	0.21	0.79
	CA	0.12	0.88
	FL	0.18	0.82
	IL	0.23	0.77
	MI	0.26	0.74
	NY	0.28	0.72
	PA	0.24	0.76
Evan McMullin	Nation	0.88	0.12

To simulate instant runoff voting in this scenario, we first use the same algorithm given in Section 4 to determine a majority winner within a given state, using the mean probability μ from Table 3 instead of Table 2 in Step 2b. The majority winner is then assigned the electoral votes for that state. In Maine and Nebraska, both the separate congressional districts and majority vote for the state are used in determining the assigned electoral votes for these states. We then sum the electoral votes for each candidate to determine the final vote. We ran simulations using 1000 different probabilities ρ generated from a truncated normal distribution

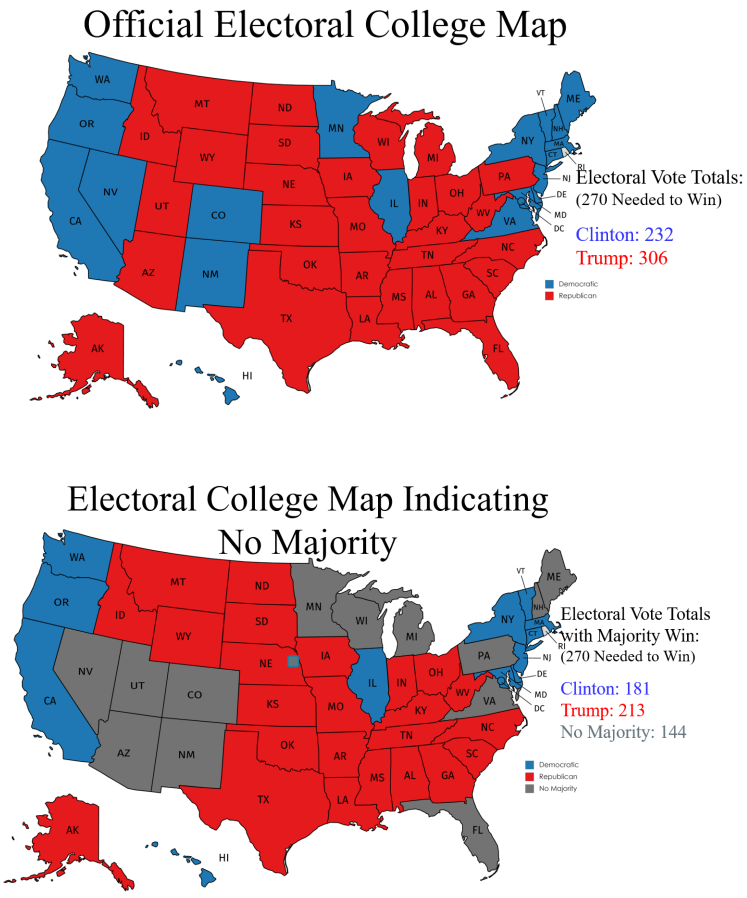

Figure 9: The first figure shows the official results of the electoral college vote in the 2016 Presidential election. The second figure shows those states which have a majority win (either blue or red) or no majority win (grey). (Created with mapchart.net and PowerPoint.) *Note: The official electoral college map shows the winners of the popular vote; however, there were 7 faithless electors who voted for a candidate other than the winner.

as described in the previous section and obtained the results in Figure 10. As shown in the figure, in only 5% of the runs, the final electoral vote stayed the same. In 83% of the simulations, the implementation of instant runoff voting further solidified Trump's win of the presidency. In 12% of the simulations, there was an increase in electoral votes for Clinton, although in no case did Clinton obtain the 270 votes necessary to win the presidency. The largest number of electoral votes received was 264 votes. In this rare instance, instant runoff voting resulted in both Michigan (16 electoral votes) and Pennsylvania (20 electoral votes) flipping from Republican to Democratic while New Hampshire (4 electoral votes) flipped from Democratic to Republican. In fact, instant runoff voting had an impact on statewide election results in 950 simulations, in which at least one state flipped from one party to the opposing party. The breakdown of which states flipped

party and how many times is given in Table 4. We note that these results are in contrast to a common narrative that it was third party voters who lost Clinton the "Big Three" states (Wisconsin, Pennsylvania and Michigan). According to our simulations, New Hampshire and Minnesota were more likely to flip than the "Big Three", indicating that those states has less stable plurality results. Although our simulations indicate instant runoff voting would not have had an effect on the overall winner of the 2016 election, in an election in which the electoral college is closer, there is a potential that instant runoff voting could ultimately effect the winner.

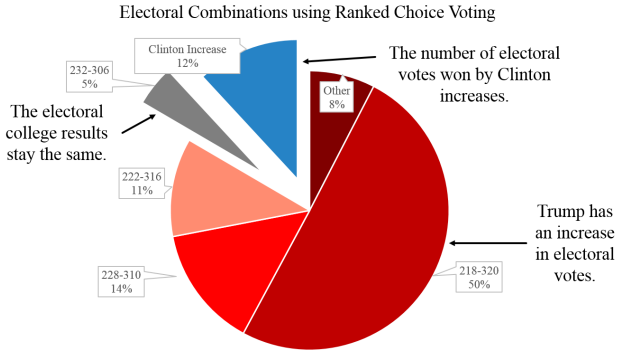

Figure 10: Results when using instant runoff voting in those states without a majority winner.

Table 4: Number of Times a State Flipped Party Due to Instant Runoff Voting

State	Total Electoral Votes for State	Number of Times Flipped (out of 1000 Simulations)
New Hampshire	4	808
Minnesota	10	735
Michigan	16	88
Pennsylvania	20	31
Wisconsin	10	9
Nevada	6	7

5.2 Electoral College with Modified Vote Counts using Instant Runoff Voting

In the 2016 election, there was a lot of support for third party candidates; however, voting for a third party candidate is sometimes seen as 'throwing your vote away'. If instant runoff voting had been implemented, this sentiment might not have been as strong since a person's vote would go to their second choice if their first choice candidate is eliminated. Therefore, in this section, we assume that if voters had been able to rank the candidates in the presidential election from the beginning, fewer people may have voted initially for Clinton or Trump (making them their second choice). Therefore, in this section, we use the same methodology from Section 4, but we consider a *modified* initial vote count by allocating some of Clinton and Trump's initial

votes back to the third party candidates. To determine the number of votes to allocate, the number of total third party votes in a state were arbitrarily increased by 10% of the original percentage of votes in the state. For example, if, in a given state, 5% of the total votes in the state were for a third party candidate initially, this would be increased to 5.5% ($1.1 * 5 = 5.5$). The extra votes are arbitrarily taken equally from Clinton and Trump. In this scenario, we were only concerned with gauging whether such a redistribution would have an affect on the outcome of the election; therefore, 10% seemed like a reasonable increase. However, in a future study, it might be prudent to further investigate this allocation and whether the votes should come equally from each candidate or whether there might even be an increase in third party voter turnout, itself, and hence an increase in votes. Some of the CCES data along with machine learning strategies might aid in this assignment in future studies, since, within the data, there were both respondents who both voted in the final election and those that did not.

In this scenario, we use a predictive model to determine where to allocate Clinton and Trump's votes. So, in the example, if third party candidates receive 5% of the vote initially in a state, then the total number of votes was increased by .5%, say a total increase of $2M$ votes. Within that state, a total of M votes was initially taken from Clinton and M votes from Trump. These $2M$ votes were allocated to either Johnson, Stein, or McMullin. In five states (Georgia, Indiana, Nevada, North Carolina and South Dakota), Johnson is the only third party candidate; in these states, the total $2M$ votes were allocated to Johnson's initial vote count. In eleven states, Johnson, Stein, and McMullin are all official third party candidates on the ballot, while the rest of the states only have Johnson and Stein as the official third party candidates. In the states with only Johnson and Stein, we use all the CCES survey respondents who indicated they voted for either Johnson or Stein and the random forests algorithm to predict whether a person who voted for Clinton (or Trump) would be predicted to be more likely to initially vote for Stein or Johnson. The average training score was 0.81 with an average testing score of 0.80. In these states, 30% of Clinton's M votes are initially allocated back to Johnson with the remaining 70% allocated back to Stein. Most of the respondents (96%) who voted for Trump were predicted to vote for Johnson using the same predictive model. Therefore, we initially allocate 96% of Trump's M votes back to Johnson and the other 4% back to Stein. The predictive score when trying to accurately predict whether a voter votes for either Johnson, Stein or McMullin in the other eleven states was much lower with an average training score of only 0.65 and testing score of 0.61. In these eleven states, 14% of Clinton's M initial votes were allocated back to Johnson, 83% to Stein and 3% to McMullin. Of Trump's M initial votes, 54% were allocated back to McMullin, 38% to Johnson and 8% to Stein. Using the new initial vote counts, we find different electoral combinations than those given in Figure 10; however, we still obtain the same distribution of increase in electoral votes for both Clinton and Trump (12% of the runs resulted in an increase in electoral votes for Clinton while 83% of the runs resulted in an increase in electoral votes for Trump). Again, the implementation of instant runoff voting in this scenario did not have a significant impact in the final winner of the 2016 presidential election; however, as discussed above, it did result in at least one state's electoral votes flipping from one candidate to the opposing candidate in 956 of the 1000 simulations.

6 Conclusions and Future Work

In this paper, we examined several different scenarios and modeled the potential outcome of the 2016 presidential election had instant runoff voting been implemented. To simulate the instant runoff voting process, it was necessary to determine which candidate particular voters might choose as their next choice in the voting process. Therefore, we created predictive models using the random forest algorithm and data from the Cooperative Congressional Election Study which could then be used on a particular set of voters to determine, based on their answers to survey questions, who their likely next choice candidate might be. Using

the results from the predictive models, we simulated instant runoff voting for both the popular vote and electoral vote using official election vote counts as initial data for the model. We then modified initial vote counts assuming more voters might have been more apt to initially vote for a third party candidate had they been given the chance to choose a major party candidate for one of their following choices. In each of these scenarios, although instant runoff voting resulted in some states switching from one candidate to another, in no case did the swing in electoral votes result in a change in the outcome of the presidential election. The closest scenario was when Clinton received 264 electoral votes (still 6 short of the necessary 270 votes) in 1 of the 1000 simulations. Moreover, in only two cases did instant runoff voting result in Trump winning the popular vote when simulating instant runoff voting on the nation as a whole.

Although we used machine learning to predict what might have happened in the 2016 presidential election had instant runoff voting been implemented, machine learning could be used to test the effect of implementing multiple alternative voting methods, such as the Borda count. Moreover, machine learning could play an even larger role in predicting potential election results prior to the election. One could also use predictive modeling as an aid prior to an election, to help determine which population might be more easily swayed towards one candidate or another, giving a target audience for campaign purposes. In summary, we have illustrated just one application of machine learning in election studies, but predictive modeling might be useful in a variety of other scenarios.

References

[1] FairVote, "How rcv works." `www.fairvote.org/rcv#how_rcv_works`. Accessed: 2017-08-15.

[2] OpaVote, "Ranked choice voting(rcv)." `https://www.opavote.com/methods/ranked-choice-voting`. Accessed: 2018-02-03.

[3] C. Bean, "Australia's experience with the alternative vote," *Representation*, vol. 34, no. 2, pp. 103–110, 1997.

[4] FairVote, "Ranked choice voting in us elections." `www.fairvote.org/rcv_in_us_elections`. Accessed: 2019-01-10.

[5] C. M. Burnett and V. Kogan, "Ballot (and voter) "exhaustion" under instant runoff voting: An examination of four ranked-choice elections," *Electoral Studies*, vol. 37, pp. 41–49, 2015.

[6] FairVote, "Benefits of ranked choice voting." `www.fairvote.org/rcv#rcvbenefits`. Accessed: 2017-08-14.

[7] R. Richie, "Instant runoff voting: What mexico (and others) could learn," *Election Law Journal*, vol. 3, no. 3, pp. 501–512, 2004.

[8] S. Bowler and D. M. Farrell, "Voter strategies under preferential electoral systems: a single transferable vote mock ballot survey of london voters," *British Elections and Parties Yearbook*, vol. 5, no. 1, pp. 14–31, 1995.

[9] P. Dunleavy, H. Margetts, B. O'Duffy, and S. Weir, *Making votes count: replaying the 1990s general elections under alternative electoral systems*. Democratic Audit of the United Kingdom Colchester, 1997.

[10] S. Thistle, "Maine's highest court rules ranked-choice voting is unconstitutional." https://www.pressherald.com/2017/05/23/maine-high-court-says-ranked-choice-voting-is-unconstitutional/. Accessed: 2019-01-11.

[11] M. Shepherd, "Maine's top court clears way for ranked-choice voting in june." https://bangordailynews.com/2018/04/17/politics/maines-top-court-clears-way-for-ranked-choice-voting-in-june/. Accessed: 2019-01-11.

[12] S. Ansolabehere and B. F. Schaffner, "Cces common content, 2016." Harvard Dataverse, http://dx.doi.org/10.7910/DVN/GDF6Z0,, 2017.

[13] M. Kuhn and K. Johnson, *Applied predictive modeling*, vol. 810. Springer, 2013.

[14] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[15] S. Raschka, *Python machine learning*. Packt Publishing Ltd, 2015.

[16] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?," in *MLDM*, pp. 154–168, Springer, 2012.

[17] E. R. Society, "Supplementary vote." https://www.electoral-reform.org.uk/voting-systems/types-of-voting-system/supplementary-vote/. Accessed: 2018-01-23.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] F. E. Commission, "Federal elections 92: Election results for the u.s. president, the u.s. senate and the u.s. house of representatives," tech. rep., Washington, D.C., June 1993.

[20] N. J. Joyner and M. L. Joyner, "Simulating electoral college results using ranked choice voting if a strong third party candidate were in the election race." http://faculty.etsu.edu/joynerm/PrimaryPartyCandidatesSimulation.pdf.

Appendix A: Electoral College Results using Instant Runoff Voting if a Strong Third Party Candidate were in the Election Race

In the final scenario, we wanted to simulate the effects of the final election results if there had been a strong third party candidate in this election. In the 1992 election, Ross Perot was such a candidate, winning approximately 19% of the popular vote and having over 30% of the final vote in Maine and 27% in Utah [19]. Since there were no third party candidates which rivaled the major party candidates in the 2016 presidential election, we considered an alternative scenario in which several of the major party *primary* candidates were assumed to be added to the general election ballot. Since candidates such as Bernie Sanders and Ted Cruz rivaled Clinton and Trump, we hoped to discern the potential impact a candidate with a lot of support might have had on the election process had they been included and instant runoff voting been implemented. Therefore, we assume that instant runoff voting is implemented in the 2016 presidential election with Democratic candidates Bernie Sanders and Hillary Clinton and four Republican candidates, Donald Trump, Ted Cruz, John Kasich, and Marco Rubio, along with the previous third party candidates, Gary Johnson, Jill Stein, and Evan McMullin. The details of the implementation can be found in [20].

In the simulation algorithm, we used the same methodology as in Section 4, but we first re-initialized the original official vote counts to account for the inclusion of multiple candidates not on the final election ballot. We did this using data from the CCES for primary voter preference as discussed in [20]. After the re-initialization, Clinton still has the largest percentage of the popular vote with 29.41% of the votes; Trump and Sanders follow closely behind with 27.62% and 21.43% respectively as shown in Figure 11. Cruz has 11.27% of the initial popular vote with diminishing percentages for Kasich, Rubio, Johnson, Stein, and McMullin. We accounted for error in the re-initialization of vote counts by considering 100 different trials with small random perturbations about the percentages shown in Figure 11. We note that in 54 of the 100 trials, Trump is the initial winner of the electoral college prior to implementation of instant runoff voting. The results after instant runoff voting is implemented is given in Figure 12.

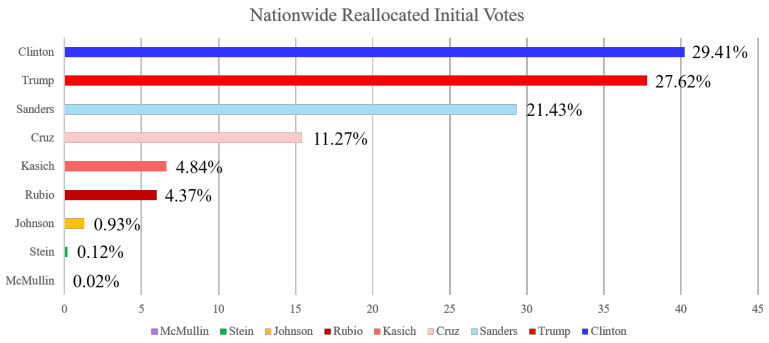

Figure 11: This bar graph shows the nationwide totals after all the initial votes are reallocated to include the new candidates from the primary election.

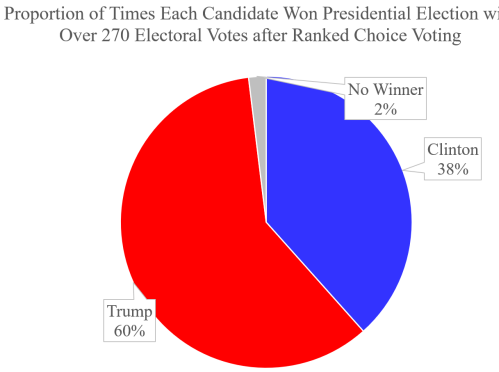

Figure 12: This figure shows the proportion of times each candidate won the presidency with over 270 electoral votes after instant runoff voting was implemented when considering a strong third party candidate.

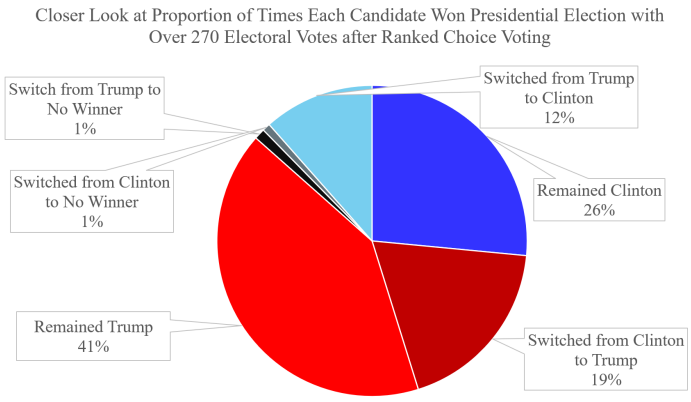

Figure 13: This figure shows details about the effects of instant runoff voting on potential outcomes of the presidential election when considering a strong third party candidate.

On first glance it appears as if instant runoff voting had only a slight impact, since Trump won in 60% of the simulations while he started as the winner in 54% of the initial vote counts. However, upon closer examination of the results, in 33% of the trials, the presidency switched from one winner to either another winner or to no winner. See Figure 13 for details on the percentage of times instant runoff voting changed the original outcome of the election in our simulated studies. In the trials in which Trump started with over 270 electoral votes initially, 76% (41/54) of the time, Trump still wins the presidency with over 270 electoral votes; however 22% (12/54) of the time, instant runoff voting leads to Clinton winning the presidency and 2% (1/54) of the time there is no winner with over 270 electoral votes. Similarly, when the election starts with Clinton having over 270 electoral votes, 57% of the time, Clinton still wins the presidency with over 270 electoral votes; however, when instant runoff voting is implemented, 41% (19/46) of the time, Trump wins the presidency and 2% of the time there is no winner with over 270 electoral votes. The spread in total

electoral votes for both Trump and Clinton is given in Figure 14. We note that although the initial number of electoral votes are centered away from the critical 270 value threshold, after implementing instant runoff voting there is a wider spread in votes and they are centered towards the critical value of 270 electoral votes required to win the presidency.

We further note that in this scenario, in several instances, Sanders acted as a spoiler candidate. In fact, 13.54% of the simulations results in Sanders getting at least one electoral vote. The total number of electoral votes for Sanders comes from winning one or more states, a direct result of instant runoff voting. We have listed the most common states in which Sanders wins electoral votes in Table 5. As seen from this table, of the times that Sanders wins a state, 50% of those times, the state is Hawaii with 4 electoral votes. Less likely, Sanders also wins New Mexico and Maine's second congressional districts with 12% and 10% of Sanders' wins respectively. Although the number of electoral votes for these states are small, if the race is close, these votes could sway the election. Since voters tend to stay within the same party, we hypothesize that if Sanders did not win these electoral votes, Clinton most likely would have won them. In 194 simulations, no candidate receives over 270 electoral votes, and in 115 of these cases, the sum of Clinton's and Sanders' electoral votes result in over 270. Although infrequent in our simulations (given our assumptions), in 71 of the simulations (0.7% of the total simulations), Clinton and Trump are the only candidates to procure electoral votes; however, both candidates tie with 269 votes each. In all the instances in which no candidate receives the required 270 electoral votes, the presidency is decided upon by the House of Representatives. We do note that Sanders is not simply a strong third party candidate but instead a strong Democratic candidate. Therefore, to interpret these simulations in terms of a strong third party candidate, we assume the strong third party candidate has a substantial influence with voters of one particular party. The results may be very different if the strong third party candidate has substantial support from voters associated with both major parties.

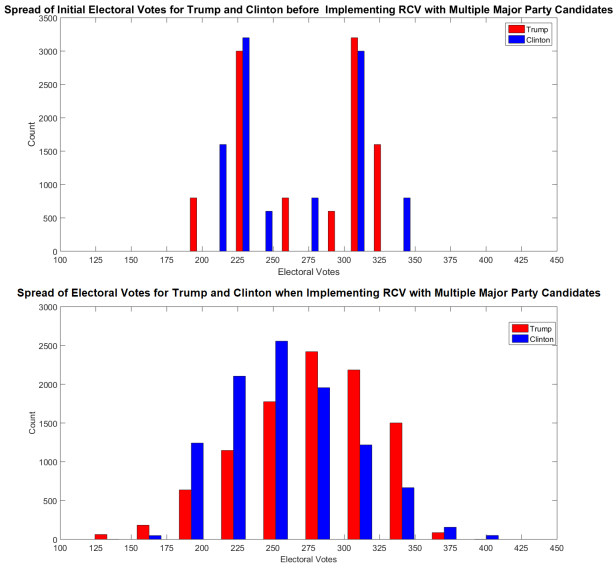

Figure 14: The first figure shows the initial spread in electoral votes won before instant runoff voting was implemented and the second figure shows the spread after instant runoff voting was implemented when considering a strong third party candidate.

Table 5: Percentage of Times Sanders Win Selected States when Receiving Electoral Votes

State	Total Electoral Votes for State	Percentage of Sanders' Wins
Hawaii	4	50%
New Mexico	5	12%
Maine 2nd CD	1	10%
California	55	6%
Nebraska 2nd CD	1	3%
Arizona	11	3%
Vermont	3	2%
Ohio	18	2%
Minnesota	10	2%
Colorado	9	2%
Iowa	6	1%
Alaska	3	1%
Texas	38	1%
Oregon	7	1%
Wisconsin	10	1%

Appendix B: Python Code

In this section, we include the basic code for predicting whether a third party candidate will vote for Trump or Clinton.

```
# Libraries
import numpy as np
import pandas as pa
import sklearn
import matplotlib.pylab as plt
# from mpl_toolkits.mplot3d import Axes3D
from matplotlib.colors import ListedColormap

# Decision tree classifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier

#read in the data
data = pa.read_csv('Data/CCES Harvard/LimitedData_PredictPrimary_PredictTrump_Clinton_red.csv')

# X contains the predictor variables
X = np.array(data.iloc[:, 1:-1])

# y contains the response variable
```

```
y = np.array(data.iloc[:, -1])

# import and modify the weights for the data
weight_data = np.array(data.iloc[:,0])
new_weight = weight_data/sum(weight_data)*len(data)

# create a permutation numbers from 0 to no. rows
a = np.arange(len(X))
np.random.shuffle(a)

# compute the train and test vectors
trainPct = .66
maxTrainIdx = int(trainPct*len(a))

# Get the testing and training data using a random shuffle
trainX = X[a[:maxTrainIdx]]
trainY = y[a[:maxTrainIdx]]
testX = X[a[maxTrainIdx:]]
testY = y[a[maxTrainIdx:]]
trainweight = new_weight[a[:maxTrainIdx]]
testweight = new_weight[a[maxTrainIdx:]]

# Random Forest model
clf = RandomForestClassifier(criterion='entropy', class_weight = 'balanced', n_estimators = 128,
max_leaf_nodes = 6)

fitted_tree = clf.fit(trainX, trainY,sample_weight = trainweight)
print('training score',clf.score(trainX,trainY,sample_weight = trainweight))
print('testing score',clf.score(testX,testY,sample_weight = testweight))

# import the data for those who voted for Cruz
dataCruz = pa.read_csv('Data/CCES Harvard/LimitedData_PredictPrimary_VoteCruz.csv')

# XCruz contains the predictor variables
XCruz = np.array(dataCruz.iloc[:, 1:-1])
weightCruz = np.array(dataCruz.iloc[:,0])
weightCruz = weightCruz/sum(weightCruz)*len(dataCruz)


# Calculates the predicted output given the variables in XCruz
PredictionValues = clf.predict(XCruz)

# Precentage Predicted to Vote Trump
(PredictionValues==4).sum()/((PredictionValues==1).sum()+(PredictionValues==4).sum())

# Precentage Predicted to Vote Clinton
(PredictionValues==1).sum()/((PredictionValues==4).sum()+(PredictionValues==1).sum())
```