Stat 195 Midterm Election Prediction

Ashwin Krishna
Hamish Nicholson
Ryan Plunkett

November 2018

1 Introduction

Upon being tasked with accurately forecasting the results of the 2018 midterm elections for the House of Representatives, our class was immediately concerned with collecting informative data that could assist in our pursuit of accurate predictions. When each group was required to download a specific feature that would be shared with the class to minimize the amount of data scraping and cleaning that the project would require, we selected incumbency status, believing that the minimal historical turnover in the House of Representatives could perhaps be leveraged to better predict these upcoming elections. In order to collect data regarding incumbency, our group produced a Python script that used the BeautifulSoup library to scrape the Wikipedia pages detailing the 2010, 2012, 2014, 2016, and 2018 elections in the House of Representatives, then stored this information in a pandas dataframe before writing the file to a CSV and submitting our feature to the course staff.

After all groups submitted their features, the files were made available via the course page on Canvas; we downloaded the data and then wrote a Python script to concatenate the features and produce a master dataframe. At this point, it was necessary to one-hot-encode the categorical variables at our disposal as numerical predictors, then perform feature selection to determine which covariates would be most helpful in forecasting the midterms.

2 Seat Prediction

2.1 Early Forecast

2.1.1 Choice of Method

Our group selected the random forest classifier as our machine learning method of choice, believing that its ease of implementation and relative interpretability were both quite valuable in this situation. Random forests have also been shown to handle high feature dimensionality well while avoiding overfitting, and thus, our group felt that such a machine learning method was appropriate for the task at hand. After determining that a random forest would be used to generate our predictions, we then needed to actually fit the classifier to the data.

2.1.2 Initial Feature Consideration

For our initial forecast, we considered the following features as potential predictors: the recent presidential vote of the district, the percentage of the district's residents with a bachelor's degree or higher, name commonness of the candidate, the share of the vote a given candidate's party had received in the previous election, the candidate's incumbency status, the percentage of minority voters in the district, the candidate's gender, the state in which the election was being held, the presidential approval rating within the district, an indicator variable that identified whether the candidate belonged to the same party as the current president, the median income of voters within the district, the amount of money raised by the candidate, and the amount of money spent by the candidate. While all of these covariates were initially considered, many were eliminated by backwards selection (described below in section 2.1.3).

2.1.3 Feature Selection

The one non-data-driven decision we made regarding covariates was the choice to eliminate the name commonness feature. This was not informed by a third-party source, but rather by gut feeling–we felt that the commonness of a candidate's name was simply not going to have an impact significant enough to warrant its use. After manually removing this predictor, we turned to a data-driven backwards selection method (implemented by the sklearn library) in an effort to hone in on those features deemed to be most important while eliminating dead weight. As discussed in class, this process

begins by considering all possible features, then recursively eliminating the feature found to be least important until such a removal noticeably impacts the performance of a machine learning method, at which point all remaining features are selected as the optimal set. After beginning with the pool of all features in consideration, the predictors we found to be the least important (and ultimately decided to eliminate) were the state in which the election was occurring and the candidate's gender. This left us with the following set of features to use as predictors: the recent presidential vote of the district, the percentage of the district's residents with a bachelor's degree or higher, the share of the vote a given candidate's party had received in the previous election, the candidate's incumbency status, the percentage of minority voters in the district, the presidential approval rating within the district, an indicator variable that identified whether the candidate belonged to the same party as the current president, the median income of voters within the district, the amount of money raised by the candidate, and the amount of money spent by the candidate. For specifics on feature importance, see section 2.2.1 (includes post-early-forecast features which will be described in more detail later).

2.1.4 Missing Data

After establishing the feature set, our group had to decide how to deal with missing data, because the random forest as implemented using the scikit-learn library is incapable of handling null features. Because of the limited time available to us prior to producing our early forecasts, the group decided to assume that the unavailable data was missing at random (while reminding ourselves to examine this assumption more closely when preparing the final forecast), and thus, rather than simply throwing away observations that contained missing values, we instead imputed the mean of all non-null values in a given column. Though easy to implement, such an approach is not particularly data-driven, as we could perhaps obtain better estimates of missing data by somehow relating a missing observation to those others in the dataset that are most similar (this technique is implemented for our final forecast and is discussed in section 2.2.2) However, via this original procedure, we were able to account for all missing elements in the data, and with a newly-completed dataset, we were ready to actually train our random forest classifier.

2.1.5 Cross-Validation

Like many machine learning methods, the architecture of the random forest is in part determined by its hyperparameters (number of estimators, maximum tree depth, etc.), and in order to achieve optimal performance, these aspects of the classifier must be tuned to best fit previously unseen data. While this task can be accomplished by setting aside a separate validation set that is viewed only for tuning, such an approach is rather costly, as it effectively throws away data that could be used for training, potentially offsetting any gain in performance achieved by the hyperparameter tuning. Thus, for smaller datasets, rather than setting aside a validation set, implementing some form of k-fold cross-validation is often deemed to be a more principled approach. To briefly summarize k-fold cross-validation, perhaps intuitively, the training data is divided into k folds, and a machine learning method (with specified hyperparameters) is fit on $k-1$ of these folds, with its accuracy evaluated on the remaining "validation" fold, cycling through the data such that each fold is used for validation exactly once. This process is repeated for each of the hyperparameter combinations in question, and that which produces the best results when averaged over the k validation sets (where "best" in this instance refers to minimizing the 0-1 classification loss function), is selected as the optimal set of hyperparameters. The machine learning method can then be fit to the entire training set using these optimal hyperparameters, and its generalization accuracy on an unseen test set can then be reported.

For our preliminary predictions, we implemented 5-fold cross-validation in order to determine the optimal number of estimators in our random forest and their maximum depth, finding that 100 decision trees with depth less than or equal to 2 resulted in the best performance. Unfortunately, cross-validation relies upon the assumption that our observations are both independent and identically distributed. Here, however, our data clearly violates that assumption, as the same district is represented in the dataframe multiple times over the period in question. Thus, if data from a later election were used to make predictions on earlier elections found in the held-out set, this form of data leakage could artificially improve our prediction accuracy without generalizing to the 2018 data. While we were unable to address this concern for the original forecast, based on feedback received from the course staff, we did modify the cross-validation procedure for our final predictions to better account for the correlated nature of our data (see section 2.2.3).

2.1.6 Predictions

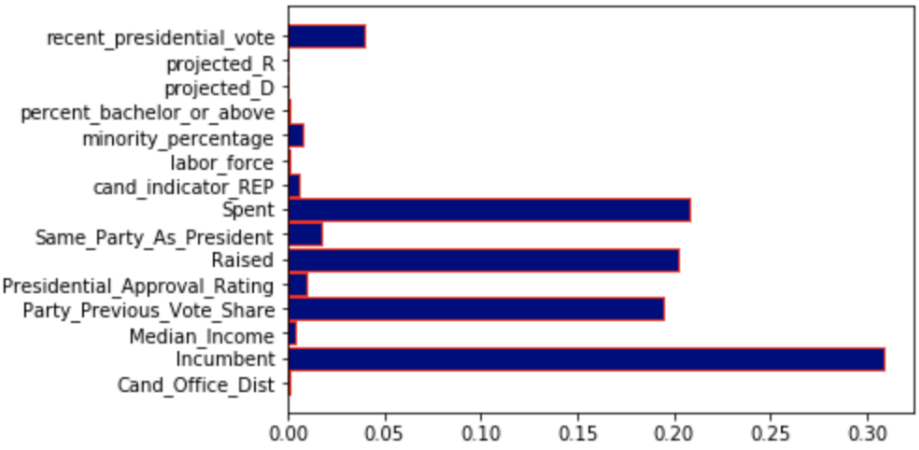
After training and cross-validating the performance of our random forest on all election data from 2010, 2012, 2014, and 2016, we used our fitted machine learning method to generate predictions for the upcoming midterm elections, where we constrained our classifications such that one and only one candidate from each district could win, guaranteeing that this requirement would be fulfilled by accepting only the maximum softmax probability of every candidate in the district as our projected winner, arbitrarily breaking ties if necessary. These results were used to produce the 435×3 matrix that was submitted as part of our early forecast, and these probabilities were then passed to our simulator in order to estimate the probability that Democrats or Republicans would carry the House following the midterms. The methodology of our simulator is described in more detail below (see section 3), though after running 100,000 trials, we concluded that Democrats are likely to hold a majority in the House of Representatives with just over 77 percent probability. Though our initial predictive process exhibited some obvious flaws, given how closely our estimates aligned with other public-facing projections, we were relatively satisfied with its performance and prepared to tweak our machine learning method for the final round of predictions.

2.2 Changes Made in Last-Minute Forecast

2.2.1 Updated Features

In addition to using all of the features that were leveraged in our early forecast, we considered adding numerous supplementary covariates to bolster our final predictions. In an effort to address the bias introduced by Republican success in recent elections, we decided to collect national polling data from multiple sources. For consistency, we averaged the Democrat-Republican spread (compiled together at realclearpolitics.com) reported by 10 of the same outlets across all of the elections in consideration. This resulted in two additional features: the national projected Democrat vote percentage and the national projected Republican vote percentage, both for the election year in question. We also considered adding features for the party of the candidate's state's governor and seat transitions from the previous year, but ultimately decided not to include these because of excessive missingness and our inability to impute sensible values. Figure 1 demonstrates the relative importance of each feature used in the random forest, with incumbency status leading the way, and the amount of money spent, the amount of raised, and the previous share of the vote received by

Figure 1: Feature Importances



a candidate's party also exhibiting significant predictive power.

2.2.2 Improvements on Dealing with Missing Data

As described earlier, when dealing with missing values in our early forecast, we simply calculated the mean of each column and used this result for imputation, replacing all null features with this generic technique. While quite simple and easy to implement, after submitting our first round of predictions and receiving feedback from the course staff, our group realized that this method of imputation was rather crude, as we had simply assumed that our data was missing at random and did not use the other features at our disposal to impute missing values in a more principled manner. For our next forecast, we decided to perform more systematic imputation: if a value was missing for a given district in a certain election, we first considered only instances of that same district in the other elections available to us. If these observations contained any non-null values for our desired feature, we calculated the mean of these and then imputed the result in place of the missing value. If this procedure did not resolve the missingnesss, we proceeded to the state level, calculating the mean of all non-null values found across all elections within the same state and then imputing this result. If data were still missing after this step, we reverted to our original imputation technique of simply replacing all null entries with the column's mean. In so doing, we hope to have captured the inherent relationships between congressional districts that are geographically near to one another, perhaps improving the

predictive power of our machine learning method.

2.2.3 Improved Cross-Validation

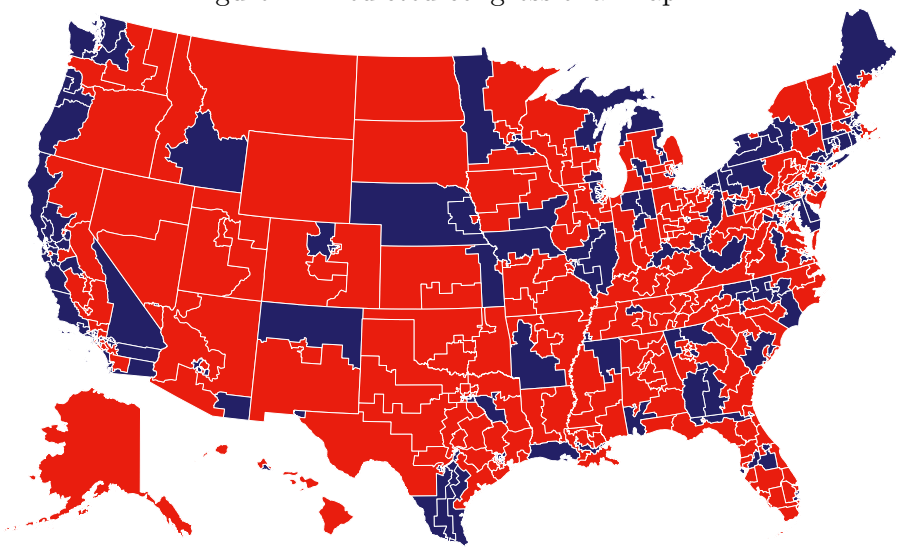
Rather than implementing basic k-fold cross-validation as we did for the early forecast, our group instead decided to devise our own custom cross-validation strategy to better account for the correlations found within our data. Typical k-fold cross-validation assumes that observations are both independent and identically distributed; because our data covers multiple elections in the same districts, it represents something of a time series, and thus, this key assumption is almost certainly violated. To better approximate reality, when tuning our hyperparameters, our group decided to train only on data that was strictly older than the fold being used as the current validation set, so as to avoid data leakage by "peeking into the future," as we did in our first round of predictions. Thus, our data resembled 2-fold cross-validation: we trained on 2010 midterm data and evaluated its performance on the 2012 elections, then trained a random forest on 2010 and 2012 before considering its accuracy in the 2014 elections. After performing this cross-validation and determining which set of hyperparameters resulted in optimal performance averaged across the validation folds, we then trained our machine learning method on 2010, 2012, and 2014 data, only then introducing the 2016 elections to determine how well our final random forest generalized to unseen data. By taking into account the structure of our data when fitting our prediction function, we hope to have better optimized our machine learning method's accuracy for this final forecast.

2.2.4 Last-Minute Predictions

After imputing missing features and performing custom cross-validation in the above manners, we concluded that the optimal random forest classifier was made up of 500 decision trees, each with a maximum depth of 2. Such a classifier was trained on the 2010, 2012, and 2014 elections, and when applied to the previously unseen 2016 data, it produced a test accuracy score of 95.2 percent, implying that our machine learning method did in fact generalize well. Having ensured that we did not overfit to the training data, our group produced another 435×3 matrix containing our last-minute predictions for the upcoming midterms, and using our simulator, we estimate the likelihood that Democrats or Republicans will carry the House of Representatives based on our generated probabilities. Again, our simulation methodology is described in more detail below (see section

3), but ultimately, we concluded that Democrats will win the House with 77.6 percent probability, most likely gaining 38 seats to earn a majority of 233 according to our projections. Like our early forecast, these predictions seem to align well with public-facing estimates informed by more granular polling data, and thus, we are relatively satisfied with our results.

Figure 2: Predicted congressional map



3 House Majority Simulation

To obtain the predicted probability of Democrats or Republicans carrying a majority in the House following the upcoming elections, we used a truncated multivariate normal distribution with means set to the probabilities found by the random forest and a covariance matrix constructed using empirical data.

3.1 Covariance

The covariance matrix was calculated using the previous party vote share feature over all years available to us in the data (i.e we calculated the covariance matrix using vote percentages in the years 2008-2016). Unfortunately, this implies that there were only 5 data points for each district, and so our input matrix X has shape 435×5. The covariance matrix is given by subtracting the mean of each row from each element in the row, and thus,

our covariance matrix can be represented as $X^T X$. Obviously, because of the limited sample size at our disposal, the empirical covariance matrix will not be a very good estimate of the truth, but given our domain knowledge regarding recent elections, even a poor approximation seems better than treating the seats as independent. In general, we would assume that most seats are positively correlated, (i.e. if Democrats are winning in one race, then it is more likely that they will win another similar seat). though there may be some instances of negative corrleation as well (i.e. if Democrats only do well in seats with high minority populations then we may expect them to do worse in less diverse districts). Upon examining the entries of our estimated matrix, we found that only 15 percent were negative, which seems to mostly align with our intuition.

3.2 Running the Simulations

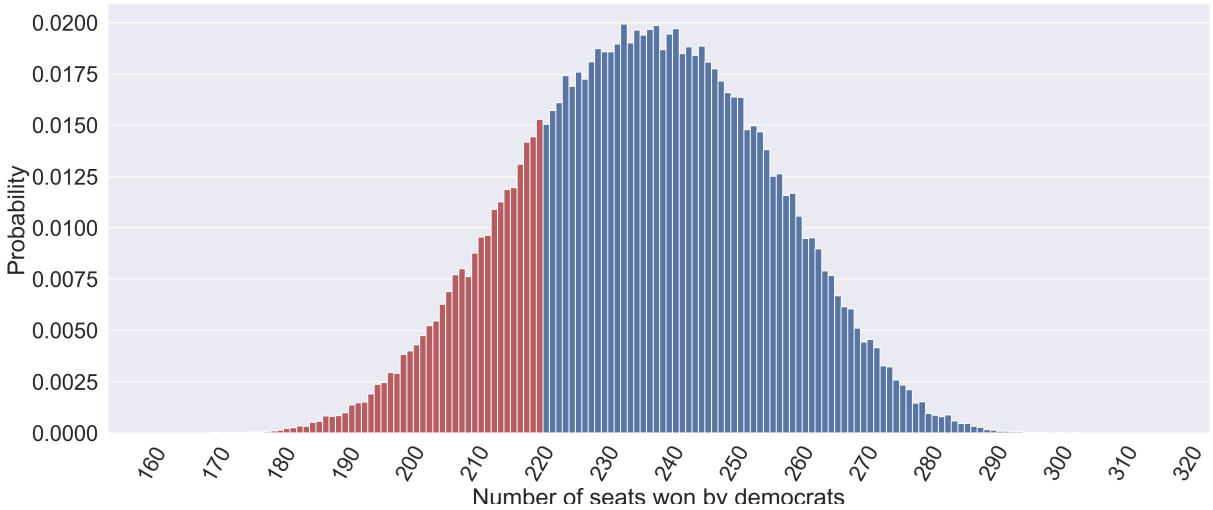
After constructing the covariance matrix, we can run a simulation, in which we predict the probability that each seat will be won by a Democrat or a Republican. After generating these probabilities, we classify them via rounding, where a 1 corresponds to a Democrat victory. These individual classifications can then be summed to identify how many seats the Democrats are expected to win, and if the result is greater than or equal to 218, we treat the simulation as a Democrat victory.

This simulation was run 100,000 times, repeatedly drawing from the resulting truncated multivariate normal distribution to calculate the probability that either the Democrats or Republicans would carry the House of Representatives. We found that Democrats will carry the House with 77.6 percent probability, capturing a mean of 233 seats with standard deviation 19. Below is a visualization of our simulation (figure 3), with the horizontal axis representing the number of seats won by Democrats, and the vertical axis as the probability of that event occurring. The bars are colored red for simulations which Republicans won and blue for simulations in which Democrats emerged victorious, thus demonstrating our rather strong belief that Democrats will win the House.

4 Conclusions

Having been tasked with predicting the results of the 2018 midterm elections, our group began by downloading a single feature (incumbency

Figure 3: Results of 100,000 simulations



status) and submitting it for use by all groups. Upon receiving all features, we concatenated the files and cleaned the resulting dataset, preparing our features for prediction. After performing simple imputation and basic cross-validation, our group trained a random forest classifier to predict the upcoming elections as part of our early forecast, using backwards feature selection along with domain knowledge to eliminate unnecessary covariates from consideration. After receiving feedback on our initial report, we realized that our group had been a bit careless with the assumptions made while producing our early forecast, and we thus chose to revise both our imputation and cross-validation techniques, as well as our features, before re-fitting a new properly trained and tuned random forest to the data. If given more time, and if more granular polling data were readily available, our group would have liked to include this information as additional features to improve our predictions, though given the performance of our machine learning method (95.2 percent accuracy on unseen 2016 election data), we feel relatively confident in our estimates. After generating seat-by-seat predictions and running our aforementioned simulator, our group concludes that Democrats should be expected to win the House of Representatives with probability 0.776, most likely capturing 233 total seats.

5 Sources

On cross validation of time series:
Why every statistician should know about cross-validation [Internet]. 2010 [cited 2018Nov5].

On random forests:
Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.