Foundations of Machine Learning
DS 3001 - Group 11
Wadie Abboud
Julian Krese
Jenny Schilling
Elizabeth Wu

Predictive Models for the 2024 Presidential Election in Virginia

Research Question:

What county-level factors can be used to build predictive models for forecasting the outcome of the 2024 presidential election in Virginia, and how high of an accuracy can we achieve with these models?

Summary:

Election forecasting has long been a goal of scholars and researchers who aim to inform voters and garner media attention and public interest. A wealth of data related to elections is available, including historical voting results, demographic information, and public opinion polls. For the state of Virginia (VA) alone, there exists a vast array of housing, economic, and social status data. Among these datasets is the National Historical Geographic Information System (NHGIS) which covers county-level summary statistics. For this study, the NHGIS data will be used alongside the general voting data for VA presidential elections from 2000 to 2020. Although incorporating data from other sources could have very well aided our predictions, we decided against including data sources not mentioned above. The reason is that outside data can vary by quality and methodologies for collection, so a realistic prediction model that has a data cleaning pipeline would be vastly more complicated and less practical in future predictions if we attempted to account for all influential factors.

The objective of this study is to identify the most significant features for building an accurate model to predict the outcome of the upcoming 2024 presidential election. In order to do so, the aforementioned datasets were first merged, cleaned, and processed for better readability and training. Our primary focus of prediction was the net vote, or the Republican vote minus the Democratic vote, which was scaled to improve predictability and later transformed back for analysis purposes. Subsequently, several predictive models were built and analyzed, including a Linear Regression Model, a Logistic Regression Model, a Decision Tree Model, a Random Forest Model, and an Artificial Neural Network. We tried different classification models as well as regression models, which resulted in using a multitude of metrics to evaluate all the models, all from the ‘sklearn.metrics’ library. Specifically, we used ‘accuracy_score’, ‘confusion_matrix’, ‘classification_report’, ‘r2_score’, and ‘mean_sqaured_error’. Given the same county-level statistics from 2020 to 2024, the models created should theoretically be able to predict the outcome to some degree of accuracy. However, we concluded that the various

models built did not achieve proficient enough performance to be used for reliable predictions at a professional level, which may be attributed to the choice of variables or inputs in the dataset.

Data:

In order to begin approaching this project, we first decided on which data sets to use to run our models. After much discussion, we concluded that merging “voting_VA_2020.csv” and “va_2016-20_demographics.csv” would form the most effective set of data to base our models on. While we did refer to the codebooks of the many counties, we ultimately decided to avoid working with all the county files directly as they contained similar demographic data over the course of the past two decades. We opted to use the most recent data set (2016-2020) in an attempt to more accurately represent the current voting population. These two files were then merged in the “data_wrangling.ipynb”file, creating a new data set named “merged_df”. This is where most of our data wrangling was done.

Next, we began to analyze the data, looking for any flaws or redundant columns/data. After looking over the raw data, the first step was to refer to the county codebooks found in the “data” folder. As defined by the codebook, the encoded column names were substituted. For example, the code “AM0KE002" represents the amount of people per county who gained their citizenship by being born in the United States. This column name was changed to “Born in US” to help with readability. A similar process occurred with the rest of the encoded columns which we believed could be useful with their current values. As for the rest, we noticed that the demographic data was split up into very specific groups. This type of distinction was not needed to predict voting patterns and were therefore simplified. This consisted of combining the values of multiple columns into one which could be used effectively when creating our models. An example of this would be the adjustment of the columns coded as "AM8FE003", "AM8FE006", "AM8FE022", "AM8FE025". According to the codebook, these columns separated population data into distinct combinations of age, gender, and disability. However, the multiple sections were unnecessary for us since anyone under the age of 18 cannot vote, regardless of any other factors. These columns were combined into a single “Total Children” column, which allows us to account for all underage citizens. This process was done with many other groups, simplifying extraneous distinctions. To finish, the unused columns were dropped. After further analysis, there seemed to be no missing values or outliers, meaning the county demographic data was now usable. Then, this data was further simplified through the removal of several voting_VA columns. These included values such as a “Libertarian” column, which was removed as the likelihood of a libertarian candidate winning is extremely low. Additionally, there seemed to exist several repeated county name columns. To deal with this issue, we ensured all county names were unique.

Finally, we needed to consolidate the multiple rows of party data (Democrat, Republican, Libertarian, Other) and mode data (Absentee, Election Day, Provisionary) which thus gave each county 12 rows of voting data. To do this, we created one column called ‘net_votes’. We first summed up all the vote counts for Democrats and Republicans across all modes, then subtracted

the total Democratic votes from the Republican votes. This means that for all negative values in 'net_votes', there was a majority Democratic win in that county, and for all positive values in 'net_votes', there was a Republican win. We opted to just omit the Libertarian and Other parties since there were no counties in Virginia in which either had a majority. Then, we scaled this net vote count using a hyperbolic arc-sine function with np.arcsinh() to make it easier to fit the prediction model. The data now seemed clean and organized enough to begin regression. To finalize our wrangling, a new file named "cleaned_VA_voting.csv" containing the clean data was added to our repository to use for our predictive models.

Results:

It should be noted that the goal of our models was to predict the votes of a county, from which we could predict the election for an entire state and then the entire county. As such, we tested the accuracy of our models on predicting the county, and not the country as a whole. The main reason for this is that any biases or inaccurate predictions at the county level would show at the state and national level, and so an accurate state and or national predictor would be reliant on an accurate county predictor.

After cleaning the data, we moved onto developing models for voter prediction. As mentioned before, we made a Linear Regression Model, a Logistic Regression Model, a Decision Tree Model, a Random Forest, and an Artificial Neural Network. For all these models, the process was very similar as the data had already been cleaned. We split the data into 'X' and 'y' datasets with the predictive variables in X and the predicted variable, 'netvotes', in 'y'. We split the data into 'X_train', 'X_test', 'y_train', and 'y_test' with 'train_test_split' from 'sklearn.model_selection'. We also used 'StandardScaler' from 'sklearn.preprocessing' on our 'X' dataset to scale variables so the order of magnitude of a variable wouldn't skew the model. We used the same test and train data for all our predictive models.

We began with the Linear Regression Model. We fit it on the training data and graphed the r-squared value against the actual values in Figure 1. The result was a Linear Regression model with a 0.665 r-squared on the test set and a 0.580 on the test set. As Figure 1 shows

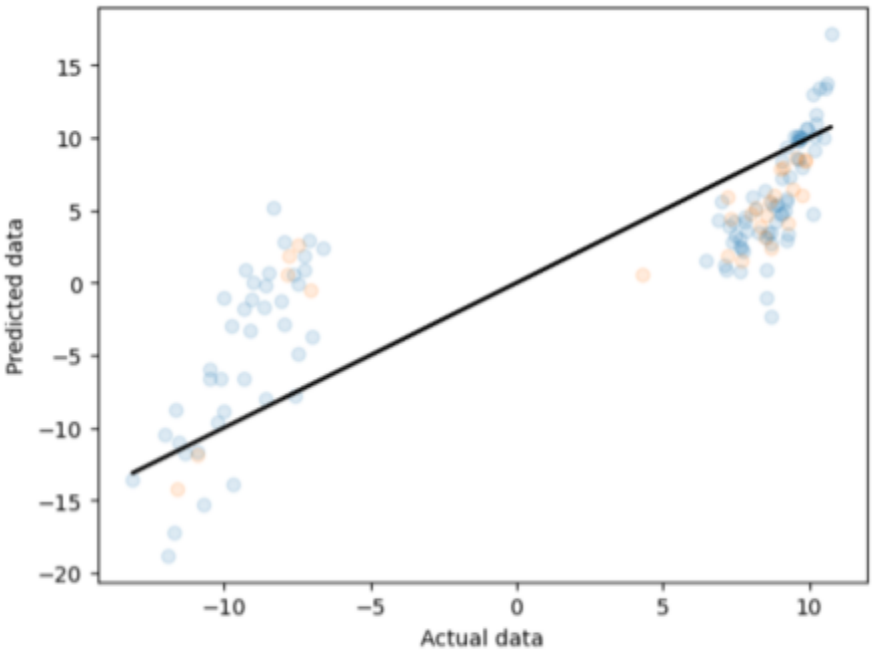

Figure 1. Scatterplot of Predicted Value Against Actual Value from Test and Train Data

visually, the Linear Regression Model was not very accurate at and reliable for predicting the votes of a county. Although this was the case, we still attempted to predict the state. The result was correct in predicting a Democratic majority vote, which we would evaluate to be a democratic win at the State level. However, it predicted this win by a margin of 81%, or 131% of the popular vote, which is impossible in the real world. This high win margin is the result of the model's low accuracy and an accumulation of inaccurate predictions.

Next we implemented a Logistic Regression Model. Because Logistic Regression only works as a classification model, we first had to transform the "netvotes" into a binary variable. This was a trivial difference, because we already defined Republican majority votes as 'netvotes' with a value greater than 0, so we just defined any 'netvotes' greater than 0 as 1 for Republican majority or 0 for a Democratic majority. After this, we were able to train the model for Logistic Regression. We were left with an accuracy score of 0.852, which once again is not great. We were also left with the confusion matrix shown in Table 1 and a report from 'classification_report' from 'sklearn.metrics' in Table 2. In Table 2's report, we can see the precision is fairly high, but the Democrat recall and f1-score are fairly low, but the model does perform fairly well with Republican majority predictions, showing a bias. This could be an effect of the train-test split for the data, or perhaps certain features disproportionately influencing Republican predictions.

Table 1. Logistic Regression Confusion Matrix

	Positive (predicted)	Negative (predicted)
Positive (actual)	2	4
Negative (actual)	0	21

Table 2. Logistic Regression Classification Report

	Precision	Recall	f1-score	Support
0 (Democrat)	1.00	0.33	0.50	6
1 (Republican	0.84	1.00	0.91	21

Next, we implemented a Decision Tree Model as well as a Random Forest Model. Similar to the Logistic Regression Model, we had to transform the input into binary classifiers for Republican or Democrat voting majorities. Therefore, we simply used the same previous split as the Logistic Regression Model and quickly fit the new models. Our Decision Tree Model had a mean squared error of 0.222 with an accuracy of 0.778, while our Random Forest Model had a mean squared error of 0.185 with an accuracy of 0.815. We also calculated the rest of the metrics and confusion matrices, which can be seen in Table 3 and Table 4, and Table 5 and Table 6 for the Decision Tree Model and Random Forest Model respectively. Once again, neither had a great mean squared error or accuracy rating, and as such we felt them ill fit for trustworthy predictions

at the national level. It is interesting to note that once again the f1-score for Republicans was higher than Democrats, perhaps suggesting the same bias in our split that appeared in the Logistic Regression Model.

Table 3. Decision Tree Confusion Matrix

	Positive (predicted)	Negative (predicted)
Positive (actual)	4	2
Negative (actual)	4	17

Table 4. Decision Tree Classification Report

	Precision	Recall	f1-score	Support
0 (Democrat)	0.50	0.67	0.57	6
1 (Republican	0.89	0.81	0.85	21

Table 5. Random Forest Confusion Matrix

	Positive (predicted)	Negative (predicted)
Positive (actual)	3	23
Negative (actual)	2	19

Table 6. Random Forest Classification Report

	Precision	Recall	f1-score	Support
0 (Democrat)	0.60	0.50	0.55	6
1 (Republican	0.86	0.90	0.88	21

Finally, we implemented multiple Artificial Neural Networks with the ‘Keras’ API. We used the same binary classifier train-test split from the previous models and tried multiple hyper-parameters, including 1-2 hidden layers, a range of densities per layer, a range of epochs, and a range of batch sizes. After creating many different models with different combinations of hyper parameters, the greatest r-squared value we reached was 0.509, coming from a model with 2 hidden layers with 16 units each, with 18 epochs and a batch size of 6. Once again, despite many different models with many different combinations of hyper parameters, we were unable to get an accurate model with a high r-squared value. In fact, the simpler Linear Regression Model out-performed the complex best Artificial Neural Network Model we created.

In summary, despite implementing multiple models, none were sufficiently adept at predicting a counties voting outcomes in support of one party or another. Therefore, we feel our models would not be qualified to predict county votes and aggregate these to a state and national level in order to predict the outcome of the Presidential Election. While our current models have not yet achieved the desired accuracy in predicting county voting outcomes for specific parties, this initial phase has provided several insights that might be useful for future applications. As such, these insights can be used to refine our methodologies and enhance our models' capabilities. Furthermore, we feel that with further adjustments and analysis, our models will be better able to predict county votes and thus contribute to forecasting state and national election outcomes.

Conclusion:

In conclusion, none of the predictive models we implemented seemed to be satisfactorily accurate to where we feel they could be used for professional predictions in real elections. There are multiple reasons for why this ended up being the case, although it is difficult to pinpoint just one.

The first reason for poor predictive models could be the fault of the selected input. We only used data from 2016-2020 because we wanted our model to represent the current voting population. However, doing so limited the data we had significantly and as such our training and testing split may have just varied too much and resulted in poor predictions.

Another fault of the input could be the variables included. It is possible the variables did not have a strong enough preference and just added noise to the models, reducing their accuracies. Another likely problem is that perhaps there are just too many variables that influence the outcome of a county's votes. Variables such as did a candidate struggle against controversial publicity during their running, or is a candidate from a particular county or state. Variables such as these that arise during the candidates campaign are especially hard to predict with past data, but are still very influential on voting outcomes. Although we initially ruled out using other datasets, in hindsight, their inclusion may have eliminated this problem and actually increased the quality of our models.

Furthermore, though the model was not technically tested on the county-level data for 2020-2024, it is plausible that demographics in Virginia have substantially shifted in recent years, rendering models trained on past datasets less applicable for future predictions. The years spanning from 2020 to 2024 were undoubtedly marked by turbulence and factors such as major policy changes, international crises, the pandemic, and other unprecedented events may have influenced voter behavior in ways that were not captured by the county-level statistics. All in all, the models' sole reliability on limited county-level statistics likely diminished its effectiveness in accurately predicting electoral outcomes.

For related studies in the future, exploring relevant variables not included in the datasets we used would be a very interesting and likely a worthwhile route to explore. There is a high chance it would lead to effective predictive models, but may also lead to skewed models that are

only effective on a year-by-year basis. Another study should also be conducted that changes what is actually being predicted. We were predicting on a per-county basis and from there would have the opportunity to add up the county votes to reach a state prediction and then a national prediction for the winner of the Presidential Election. However, this may simply have introduced too much noise to the models. Perhaps predicting at the state level without considering counties would lead to more accurate models, or predicting based off of gerry-mandered regions instead of grouping the state as a whole. Another future study to conduct would be to only focus on one predictive model and get it to a maximum accuracy by focusing intently on its hyper-parameters. We looked at multiple models at a higher level, but it could be that we needed to focus instead on one model at a lower level.