Leveraging Bellwether Counties for National Election Forecasting with Random Forest

Evan Shields

eshie003@odu.edu

Abstract

This project explores the use of a random forest machine learning model to predict presidential election outcomes in bellwether counties, which historically align closely with national results. By leveraging demographic and historical voting data, a Random Forest classifier was trained to identify patterns and predict whether a Republican or Democratic candidate would win in these key counties. The dataset was divided into training data (elections before 2020) and testing data (the 2020 election) to evaluate the model's performance. Hyperparameter tuning using RandomizedSearchCV improved the model's accuracy by optimizing parameters like tree depth and the number of trees. While the model achieved a test accuracy of 73.3% and performed well in predicting Republican victories, it struggled to predict Democratic wins. These findings highlight the challenges of election prediction, including the unique factors of each election and the exclusion of candidate-specific attributes, which are difficult to quantify. Nevertheless, the study underscores the potential of machine learning to provide valuable insights into political patterns and serves as a foundation for future advancements in election forecasting.

1 Introduction

Bellwether counties have long been known as reliable indicators of the results of United States presidential elections (Matsumoto, 2021). These counties have voting patterns that closely mirror the national outcome, making them valuable for understanding broader political trends. This project focused on using the machine learning model, random forest, to predict election outcomes in these counties based on demographic data and historical voting results. The aim was to build a model that could identify patterns in demographic and historical voting data in these key counties and predict whether a Republican or Democratic candidate would win. By leveraging a Random Forest classifier, the study sought to analyze how well a machine learning technique could perform in this context. However, each election is unique, shaped by specific events and candidate attributes that are not easily captured in historical datasets. This project does not consider such

factors, focusing instead on party affiliations and broader demographic trends, which pose challenges for achieving highly accurate predictions.

2 Data

The data used for this project consisted of county-level presidential election results from 2000 through 2020 (MIT Election Data and Science Lab, 2021) and demographic information for all counties in the United States (Glozab, 2024). For my purposes, I followed the definition of a bellwether county to be a county that has correctly predicted all or all but one presidential election outcome from 1980 through 2020 (Matsumoto, 2021). Filtering to only counties that match this standard, thirty counties remained. The target variable (election winner) was created based on the 'candidatevotes' and 'candidate' columns. The target variable was made binary, so 1 indicated a Democratic victory and 0 indicated a Republican victory. The resulting class balance for the target variable was 79 1s and 101 0s. The dataset also had columns for year, state, county, and total number of votes, as well as information about third party votes as well. Third party votes were not taken into account when determining the winner of a county. The demographics data included a range of predictors such as population size, race and ethnicity proportions, and age range counts. Altogether, these variables provided a comprehensive view of the factors influencing voting behavior in the counties.

Feature engineering was applied to enhance the dataset by creating additional variables that could capture trends and provide historical context. Cumulative averages of voting metrics were calculated to allow the model to identify long-term patterns in voting behavior. Ratios including voter turnout, margin of victory, and population growth since the last election year were also added. These features provided a deeper understanding of changes over time and offered the model a way to learn from shifts in demographic and voting dynamics. For example, voter turnout ratios highlighted engagement levels in past elections, while margin-of-victory ratios emphasized the competitiveness of them. Together, these engineered features enriched the dataset and improved its suitability for predictive modeling.

After joining these two datasets, adding the additional features, and removing unnecessary columns, the data was split into two parts: training data included all election years before 2020, while testing data focused on the 2020 election. This time-based split ensured that the model was trained on historical trends without being exposed to the test data beforehand. The training set allowed the model to learn patterns from previous elections, while the test set provided a way to evaluate how well the model generalized to new, unseen data. The training data had 150 rows (30 counties x 5 elections) and 32 columns, and the testing set consisted of results specifically from 2020 (30 rows and 32 columns).

3 Machine Learning Method

The Random Forest algorithm was selected as the machine learning method for this project due to its robustness and versatility. Random Forest is an ensemble

technique that generates multiple decision trees during training and combines their outputs to make predictions. Each tree is trained on a random subset of the data, a method called bootstrapping, which helps reduce overfitting and improves the model's ability to generalize to new data. This approach makes the algorithm particularly effective when working with datasets that contain complex relationships between features.

In the context of predicting election outcomes, Random Forest was well-suited to handle the intricate interplay of demographic variables, such as population size, racial proportions, and voter turnout, alongside historical voting patterns. Its ability to capture non-linear relationships and interactions between features provided a significant advantage for this task. Moreover, Random Forest is robust to outliers and noise in the data, making it a reliable choice for datasets with diverse and varied characteristics.

Another benefit of Random Forest is its capacity to quantify feature importance, offering insights into the variables that most influenced predictions. Hyperparameter tuning was performed to optimize the model's performance, adjusting key parameters such as the number of trees, the maximum tree depth, and the minimum number of samples required for splitting nodes. These enhancements ensured the algorithm achieved a balance between predictive accuracy and computational efficiency, making it well-suited for analyzing the test data.

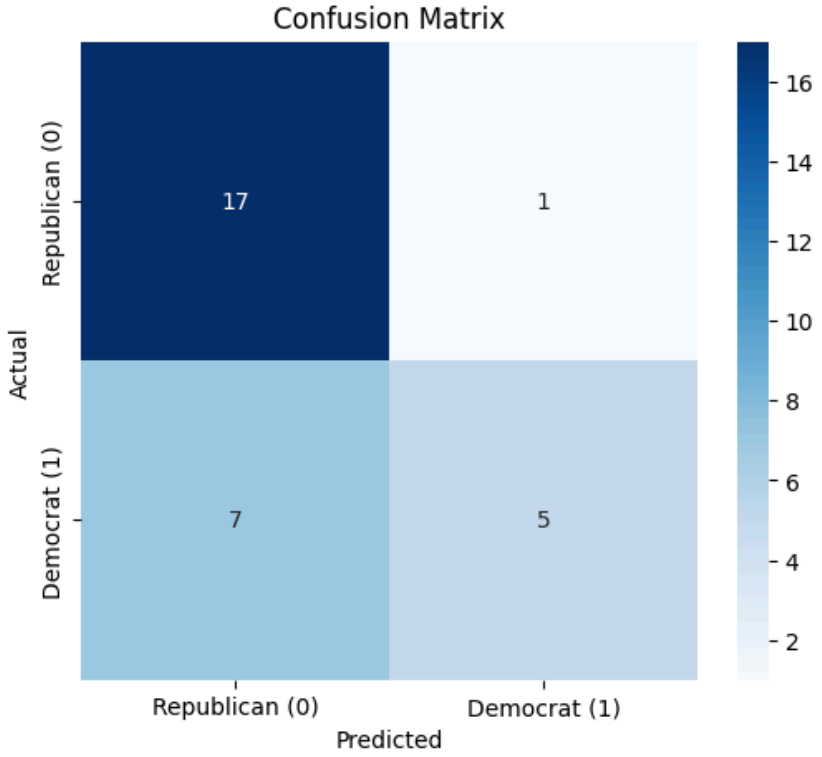
4 Implementation

To enhance the Random Forest model's predictive performance, hyperparameter tuning was conducted using RandomizedSearchCV from Python's scikit-learn library. This technique explores a wide range of parameter combinations. Key hyperparameters were adjusted, including the number of trees in the forest, the maximum depth of each tree, and the minimum number of samples required to split a node. These parameters significantly influence how the model learns patterns from the data. For instance, increasing the number of trees generally improves stability and reduces variance, while adjusting the tree depth balances the trade-off between underfitting and overfitting. The optimized parameters identified through the random search process included 50 trees (n_estimators), a maximum depth of 10, a minimum of 5 samples required to split a node (min_samples_split), and a minimum of 2 samples per leaf node (min_samples_leaf). Additionally, the best configuration set max_features to None, meaning all features were considered at each split, and enabled bootstrapping.

RandomizedSearchCV also incorporated cross-validation, a method that partitions the training data into multiple folds. During the tuning process, the model was trained and validated on different combinations of these folds, ensuring a more reliable measure of its performance. This step reduced the risk of overfitting the model to the training data and provided a stronger indication of how the model would perform on unseen data. By systematically testing various hyperparameter settings and leveraging cross-validation, the process identified an optimal configuration that maximized the model's accuracy on the test set.

5 Results and Discussion

The results showed that the model performed well in predicting election outcomes for Republican victories, achieving high precision and recall for this class. However, its performance was weaker for predicting Democratic victories. The test accuracy of the model was 73.3%, which suggests okay predictive power and room for improvement. When examining the classification report, the precision for Republican wins was 70.8%, and the recall was 94.4%. For Democratic wins, the precision was 83.3%, but the recall dropped to 41.7%. This means the model was more consistent in identifying Republican victories but often misclassified Democratic wins.



The confusion matrix above further highlights this imbalance. Of the 12 Democratic wins in the test data, the model only correctly identified 5 of them, while it predicted 17 of the 18 Republican wins accurately. This discrepancy suggests that the model struggled to capture all the nuances associated with Democratic victories.

6 Conclusion

The findings of this project demonstrate that while machine learning models, like the Random Forest classifier, can provide valuable insights into electoral patterns in bellwether counties, there are important limitations to consider. The model showed strong performance in predicting Republican victories, but struggled with Democratic wins, highlighting potential issues in feature selection and the balance

of data. Expanding the dataset to include economic indicators, social factors, and more dynamic, election-year-specific data could enhance future models' predictive power. Exploring other algorithms, such as gradient boosting or ensemble methods, may also lead to better accuracy and balance between classes.

It is essential to acknowledge that each election is unique, influenced by current events, voter sentiments, and candidate-specific factors that are difficult to capture in historical datasets. These nuances can significantly impact outcomes, making it challenging to create a one-size-fits-all predictive model based on past elections alone. The current approach, focused on party affiliation and demographic trends, does not account for personal candidate characteristics that could sway voter decisions. Addressing these limitations will be crucial in future research.

Despite these challenges, this project illustrates the potential of machine learning to provide meaningful insights into political behavior and election forecasting. As data collection and modeling techniques advance, machine learning could become an increasingly important tool for understanding electoral trends and informing political strategies. Further refinement and expansion of the dataset, along with the incorporation of more sophisticated techniques, could help build more comprehensive models capable of capturing the complexity of election dynamics.

7 References

Glozab. (2024). County-level US demographic data, 1990-2020 [Data set]. Kaggle. https://www.kaggle.com/datasets/glozab/county-level-us-demographic-data-1990-2020/data

Matsumoto, R. (2021). Where did all the bellwether counties go? FiveThirtyEight. https://fivethirtyeight.com/features/where-did-all-the-bellwether-counties-go/

MIT Election Data and Science Lab. (2021). County presidential election returns 2000-2020 (Version 13) [Data set]. Harvard Dataverse. https://doi.org/10.7910/DVN/VOQCHQ