MACHINE LEARNING: CLASSIFICATION

PREDICTING 2020 U.S. PRESIDENTIAL ELECTION RESULTS BY STATE

FINAL PROJECT SUMMARY

By vectorkoz, January 2024

Contents:

I. Introduction and specification of the project goals

Political analysts have long speculated that state of the economy has a great influence on election results in democratic countries. I have found it interesting to test this assumption. It is valid to assume that election results depend upon many other factors, such as social policies, popular perception of a candidate's personality, foreign policy situation, etc. However, these criteria are highly subjective and hard to quantify, as opposed to economic indicators. While I will not argue that all economic indicators are perfectly objective and highly informative, it is evident that they offer a quantifiable representation of the general state of society.

The main goal of this project is to construct models predicting U.S. presidential elections results by U.S. state using mostly economic data, and a few political characteristics, while employing methods taught in Coursera's "Supervised Machine Learning: Classification" online course. This project also serves as the final assignment of the course.

This project was created with Python in JupyterLab using scikit-learn, pandas, matplotlib, and a few other packages.

The constructed dataset contains 6 different target variables. The project includes attempts to achieve the best possible classification results for each target using specific types of models, and to understand why better results were not achieved. Also included are explanations of a well-performing model using 3 different methods, as well as suggestions for future work.

II. Summary of data gathering, feature engineering, and of the resulting dataset

For elections data, the project used the "U.S. President 1976–2020" dataset. In the U.S., votes in presidential elections are counted separately in 50 U.S. states and the District of Columbia. The winner of the national election is determined via the Electoral College and may be different from the candidate that won the most votes over all states, i.e. national popular vote winner. The dataset contains vote totals for every significant candidate in every state-level election for all national presidential elections from 1976 to 2020. While some third-party candidates, most notably Ross Perot in 1992, won a significant share of votes in some state-level elections, all 612 state-level presidential elections in that time period were won by a candidate from either Democratic or Republican party, which allows to treat predicting results of a state-level presidential election as a binary classification problem.

There are multiple ways to describe results of a U.S presidential election. In this project, for every one of the 561 state-level presidential elections from 1980 to 2020 the following target variables were constructed:

1) **"winner"**, encoding who won the state-level election (value of 1 representing a Democratic candidate's win, value of 0 representing a Republican candidate's win),
2) **"swing"**, encoding how the election results in a state had changed compared to the previous election in that state (value of 1 representing that the state had swung towards Democrats, which means that the difference between candidates' vote shares, a.k.a. the vote percentage margin, has improved for the Democratic party, i.e. Democrats won the state by a higher percentage margin than 4 years previously, or won the state after losing it 4 years previously, or lost the state by a lower percentage margin than 4 years previously, and value of 0 vice versa for Republicans),
3) **"percent_vs_national"**, encoding how the election results in a state compare to the national popular vote results (value of 1 representing that vote percentage margin in the state is better for the Democratic party than nationally, i.e. Democrats won the state by a higher percentage margin than the national popular vote, or the state was won by Democrats while the national popular vote was won by Republicans, or Democrats lost the state by a lower percentage margin than the national popular vote, and value of 0 vice versa for Republicans),
4) **"swing_vs_national"**, encoding how the change in election results in a state compares to the change in national popular vote results (value of 1 representing that the difference is favorable for Democrats, i.e. the swing towards Democrats in the state is higher than the national popular vote swing towards Democrats, or the state swung towards Democrats while the national popular vote swung towards Republicans, or the swing towards Republicans in the state is lower than the national popular vote swing towards Republicans, and value of 0 vice versa for Republicans),
5) **"winner_vs_national_winner"**, encoding how the winner of the election in a state compares to the Electoral College winner (value of 1 representing that these winners are different, value of 0 representing that they are the same),
6) **"winner_vs_national_pv_winner"**, encoding how the winner of the election in a state compares to the national popular vote winner (value of 1 representing that these winners are different, value of 0 representing that they are the same).

The choice of only the presidential elections from 1980 to 2020 was made due to data availability (calculating **swing** and **swing_vs_national** for 1976 would require 1972 data), as well political commonality. According to some political scientists, 1980 presidential election was the start of the new political era in the U.S. called the Sixth Party System, which, depending on the interpretation, either has finished in 2016, or still exists today.

The economic data was taken from the report of the Bureau of Economic Analysis of the U.S. Department of Commerce. The report consists of multiple datasets containing values of many economic indicators for each U.S. state, the District of Columbia and the U.S. as a whole across many years. Choosing the indicators with no missing values, and only those calculated using the same method over the period from 1976 to 2020, 69 economic indicators were selected, showing wages and salaries, employment levels, government benefits, etc., i.e. the economic indicators impacting day-to-day lives of the voters.

Since many values in the dataset were absolute, and populations of U.S. states are significantly different, all data was recalculated per capita using the population data contained in the report. Furthermore, most values in the dataset were monetary and not adjusted for inflation. Because the election year was not going to be a parameter in any of the models, these values needed to be adjusted for inflation to be useful by themselves. Therefore, the values of the economic indicators were used only in relation to their previous value and the per capita national value.

For each economic indicator, four features, identified by the following prefixes, were constructed:

1) **"4ych_**", equaling the current indicator value in the state divided by that value 4 years previously, i.e. during the previous presidential election,
2) **"1ych_"**, similarly showing the 1 year change in value,
3) **"vn_"**, equaling the current indicator value in the state divided by current indicator value in the whole U.S.,
4) **"vn_4ych_"**, equaling the **4ych_** value in the state minus the same value in the whole U.S.

The fact that all resulting economic features were relative meant that all of them were roughly on the same scale.

A few dummy variables were added to the dataset, encoding well-known information that was assumed to have an impact on the election results: whether the incumbent president was running for reelection and which party had won the two previous national presidential elections. Furthermore, to encode the election jurisdiction, 51 dummy variables representing U.S. states and the District of Columbia were added. The dataset ended up having the following structure.

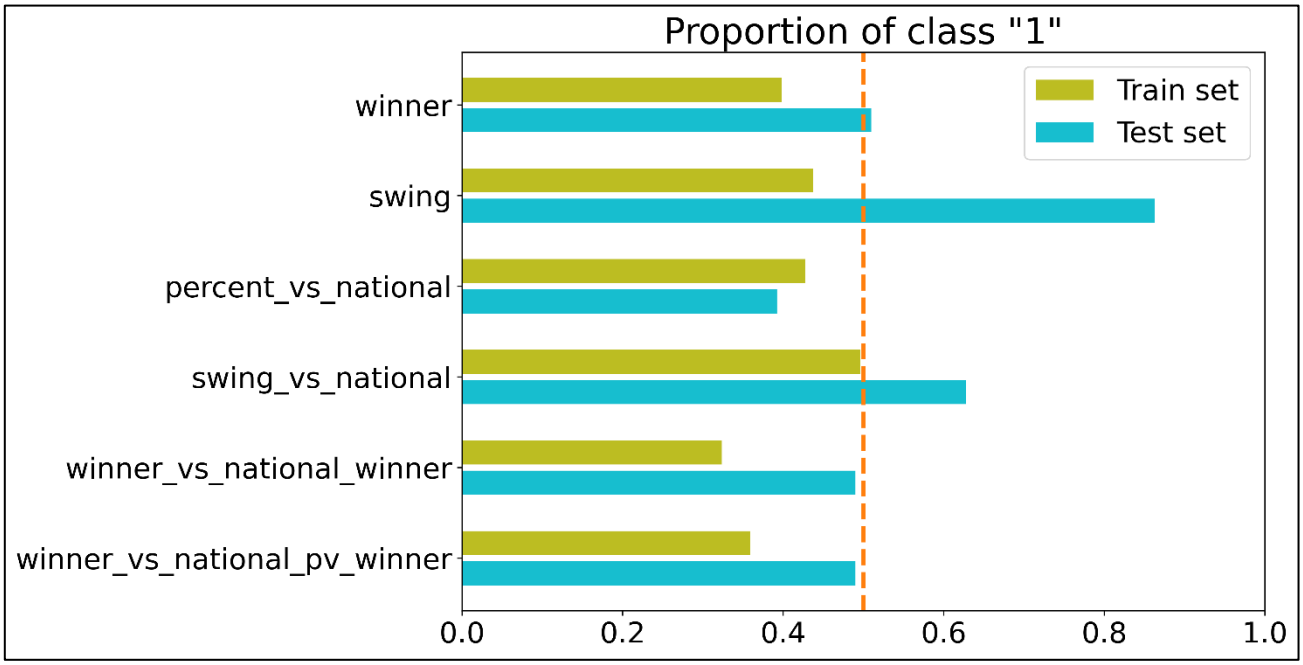
The dataset

Features			Targets
276 economic features	6 dummy variables representing election info	51 dummy variables representing U.S. states and D.C.	winner, swing, percent_vs_national, swing_vs_national, winner_vs_national_winner, winner_vs_national_pv_winner

Train set	Test set
1980-2016 : 510 instances	2020 : 51 instances

The year of the election was not used as a part of any models. However, it was used for splitting the data between train and test sets. The idea of all models was to use the historic data for training, and predict the 2020 election results based on economic features. In addition, in Logistic Regression and SVM models, election year was used for sample weighting: instances closer in time to 2020 received higher weights.

After splitting the dataset, class distribution of all targets among train and test sets was analyzed.



As seen on the bar plot, the classes in the train sets of all targets are more or less balanced, and the class proportions in the test sets are similar to the train sets for all targets apart from **swing**. In a usual machine learning problem, this could be easily remedied by performing a stratified train-test split. However, in this case the train-test split is dictated by the nature of the problem itself: it is possible to use 2020 data on some states to try to predict 2020 results in other states, but it can only make sense of interpretation, not prediction, is the goal. Resampling the train set does not make sense either: during feature engineering it was discovered that in most years, the large majority of states swing towards one party, and obviously, the exact party towards which the swing occurs can only be known with certainty after the election. Therefore, it was not possible to fix the imbalance of class proportions in train and test sets of **swing**. This proved to be a problem during modeling.

III. Summary of modeling, visualization and model explanations

This project used Logistic Regression, Support Vector Machines and Random Forests to predict all six targets in the dataset. Gradient Boosting was used to predict **winner**, **swing**, **swing_vs_national** and **winner_vs_national_winner**. Model Stacking was used to attempt to improve predictions of **winner** and **winner_vs_national_winner**.

All modeling, apart from Model Stacking, was done after a search for best hyperparameters. For Logistic Regression, Support Vector Machines and Gradient Boosting, hyperparameter search was done via cross validation using the same set of stratified train-validation splits. For Random Forest, hyperparameter search was done using out-of-bag error on the whole train set. For Model Stacking, cross validation was not attempted due to computational limitations.

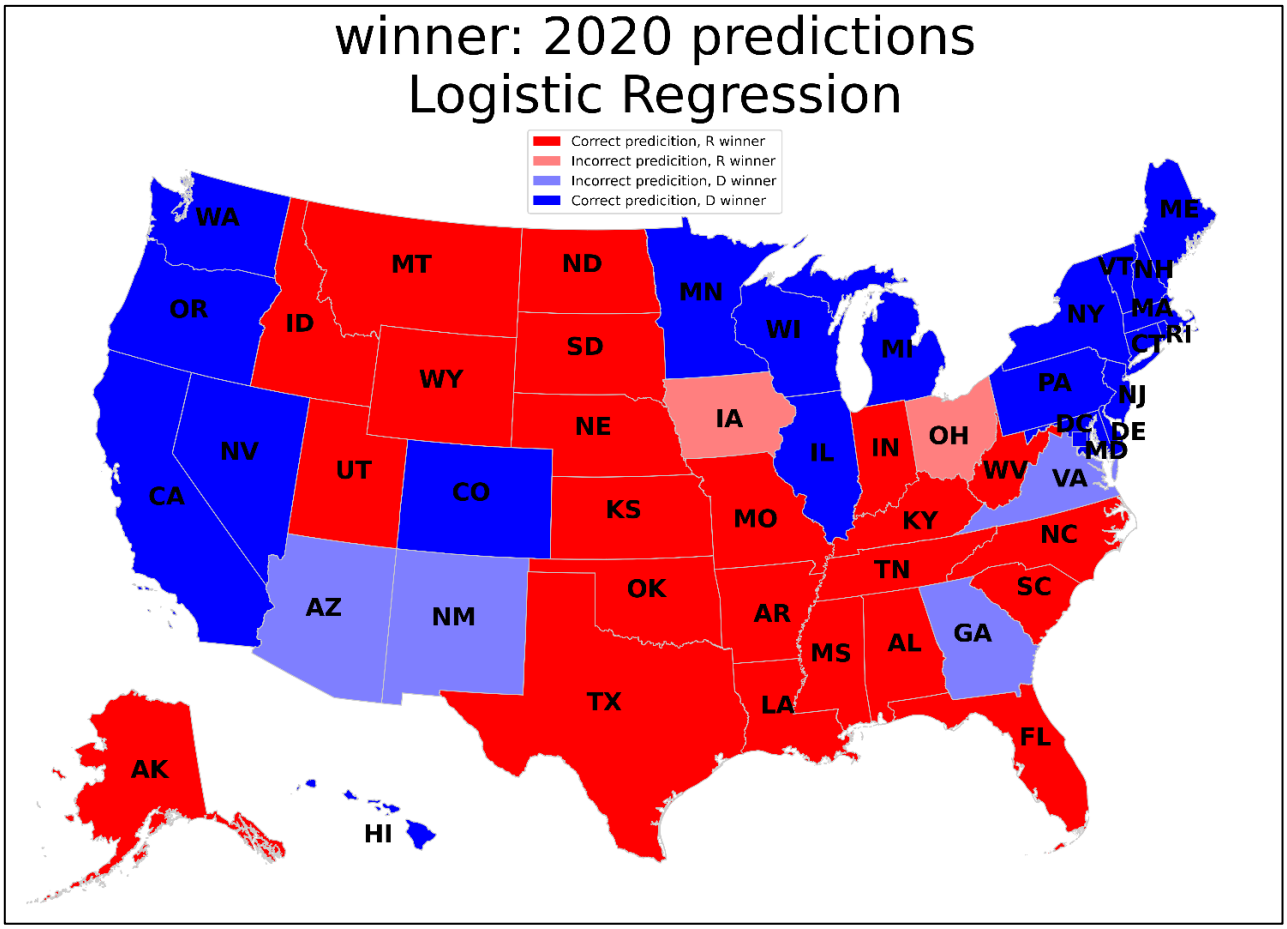
To visualize model performance, a function was written constructing choropleth-like maps of the contiguous U.S. with the addition of Alaska and Hawaii using matplotlib and geopandas packages.

For model explanation, a well-performing Random Forest model was analyzed via MDI importance, permutation importance, and a test set instance was explained via LIME.

IV. Logistic Regression models

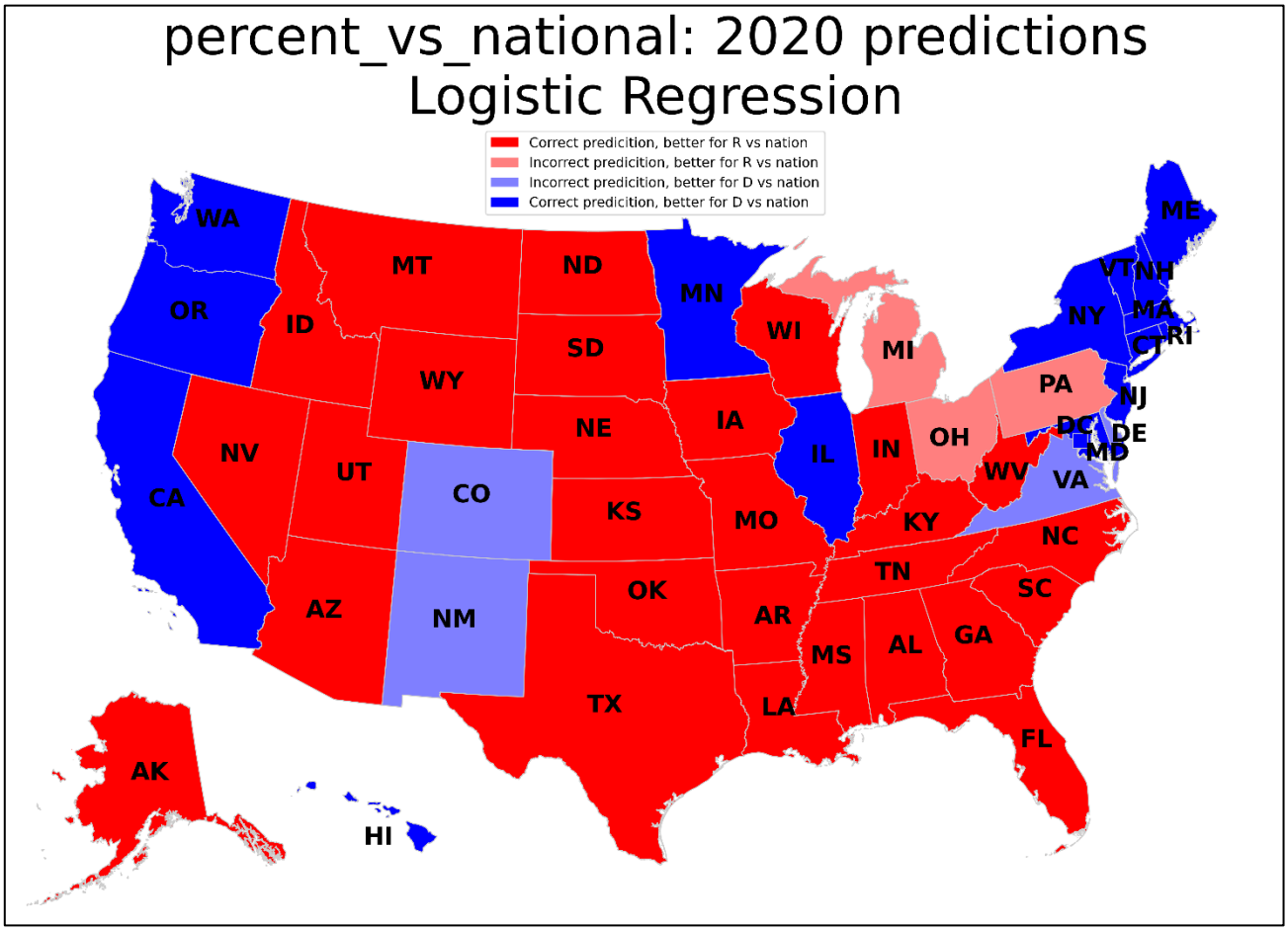
For all targets, Logistic Regression models with elastic net regularization were constructed. During cross validation, 10 best pairs of values of regularization strength and ratio between L_1 and L_2 penalties were found for every target. After that, most of them were used to fit Logistic Regression on the whole train set and predict the test set.

For **winner** and **percent_vs_national**, relatively good model performance was achieved.



On this choropleth, the output of the best-performing Logistic Regression model for **winner** is shown. Republican-won states are colored red, Democratic-won states are colored blue, correct predictions are shown in bright color, incorrect predictions are shown in pale color. Every state can be identified via commonly used two-letter abbreviations. The main area of the choropleth is centered on the lower 48 U.S. states. Note that Alaska and Hawaii are not shown to scale, or in the correct geographical spot. Also, a larger square was added to the choropleth instead of District of Columbia's borders, as D.C. would be too small to see by itself.

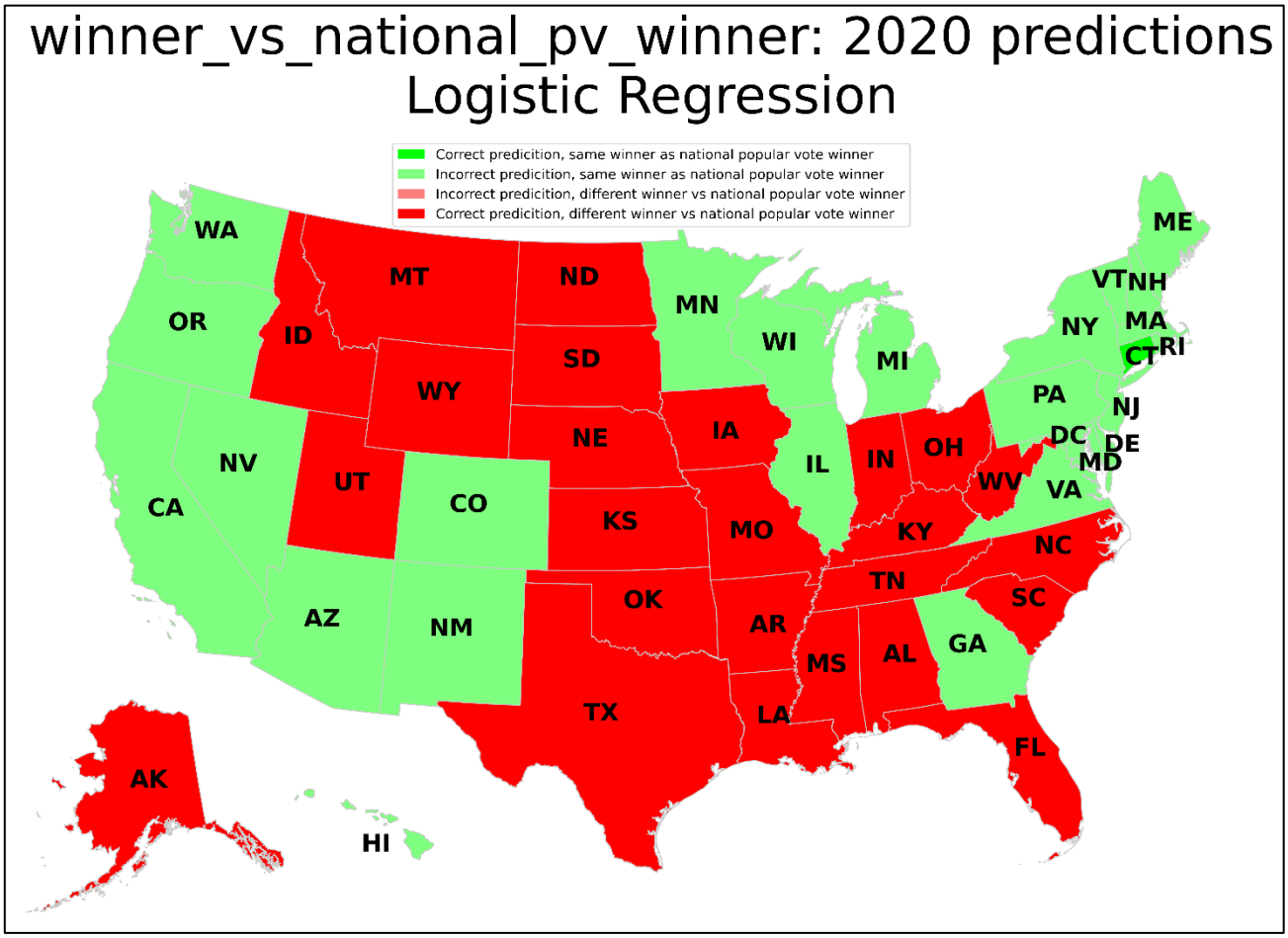
Overall, this particular model achieves accuracy of 45. (In this report, accuracy will be given in terms of the number of correct predictions, since the train set always has the same size of 51). This is a relatively good prediction for a simple model: it got most states right, the ones that it got wrong had voted for a different party in some previous elections, and if all states voted exactly as the model predicted, the winner of the Electoral College would be the same.



Here, the output of the best-performing Logistic Regression model for **percent_vs_national** is shown. This model achieved the accuracy of 44. This is also not a bad result in general, however, the particular prediction of Delaware (DE) voting more Republican than the nation is pretty bad, since Delaware is a safe Democratic state, and Delaware was the home state of the Democratic candidate Joe Biden, and, according to political scientists, presidential candidates tend to do better in their home states. This information was not included in the dataset. Perhaps, its inclusion would have made the model perform better.

However, it was also found that Logistic Regression models were not stable in terms of connection between cross-validation and test scores. In other words, for **winner** and **percent_vs_national**, good cross validation scores did not guarantee good test scores: while some other models performed as well as the ones shown, many models with similar cross-validation scores did not perform well on the test set.

As for the other targets, their best cross-validation scores using Logistic Regression were lower, and no models performed well on test sets, with all of them mostly predicting only one class, such as the model for **winner_vs_national_pv_winner** shown below.



This is not a useful model, since it predicts all but one instances to be of class 1, when in reality there are 25 out of 51 class 1 instances in the test set.

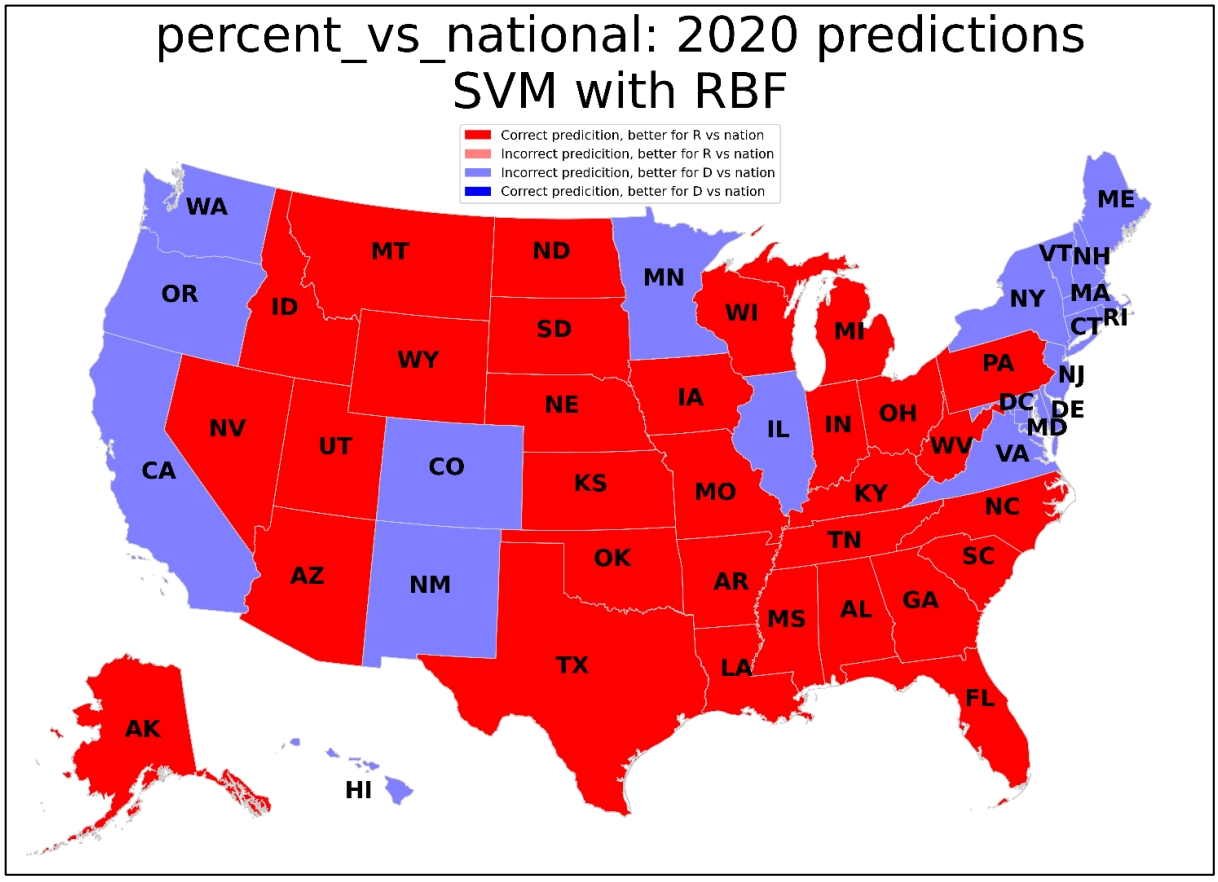
V. Support Vector Machine models

Before building SVM models, all features were transformed using standard scaling (the scaler was fit only on the test set). As noted previously, all features were roughly on the same scale to begin with, however, this step was taken to attempt to improve SVM performance, since SVM models are very sensitive to scale.

For all targets, SVM classifier models with various kernels were constructed. During cross validation, hyperparameter search was performed among kernel types (polynomial, Gaussian (RBF) and sigmoid kernels), values of regularization strength and values of polynomial function degree (only for polynomial kernels). 10 best combinations of values were found for every target.

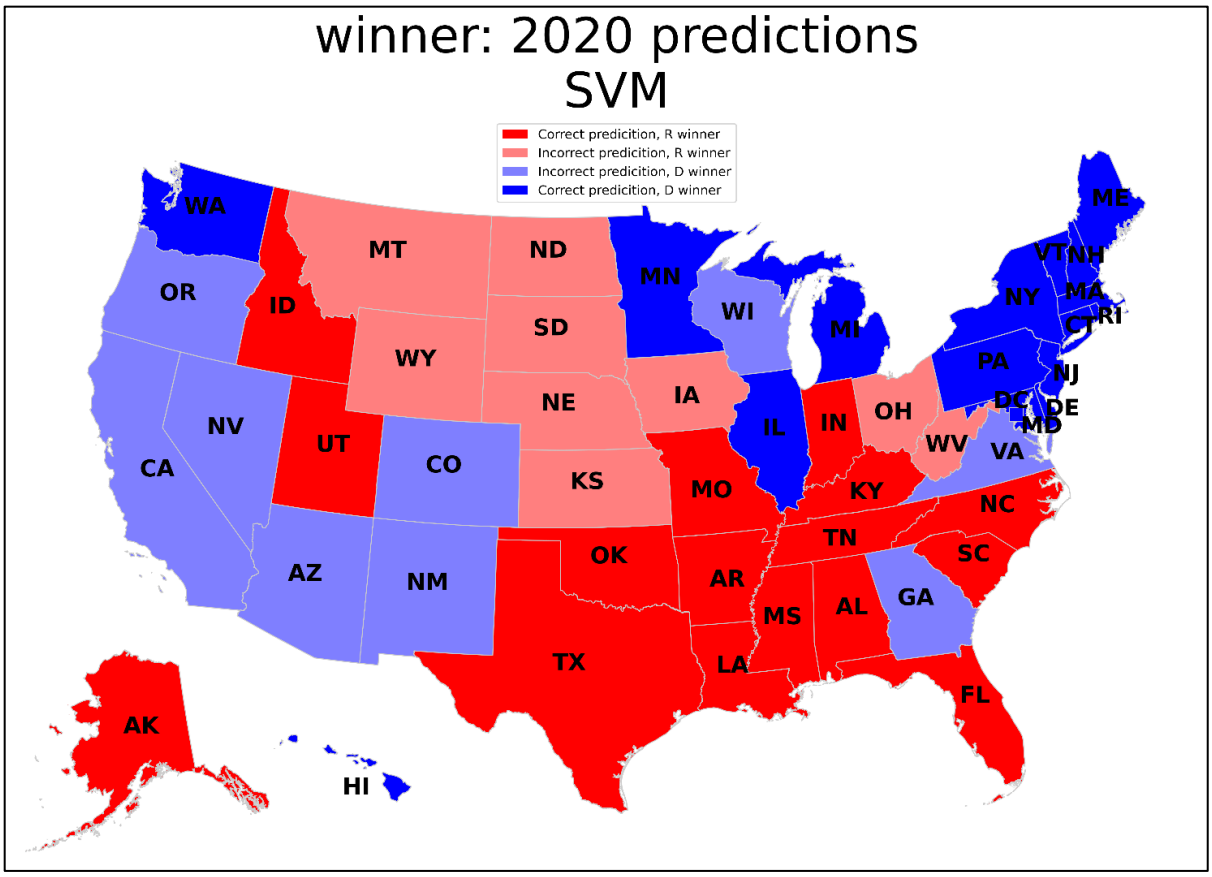
No models with sigmoid kernels achieved good scores during cross-validation. Some models with RBF and polynomial kernels achieved relatively good cross-validation scores, although lower than Logistic Regression models. After that, many models with RBF and polynomial kernels were used to fit Logistic Regression on the whole train set and predict the test set.

However, on the test sets, no SVM model achieved good performance for any target. Models with RBF kernels, such as the one for **percent_vs_national** shown below, almost exclusively predicted only one class.



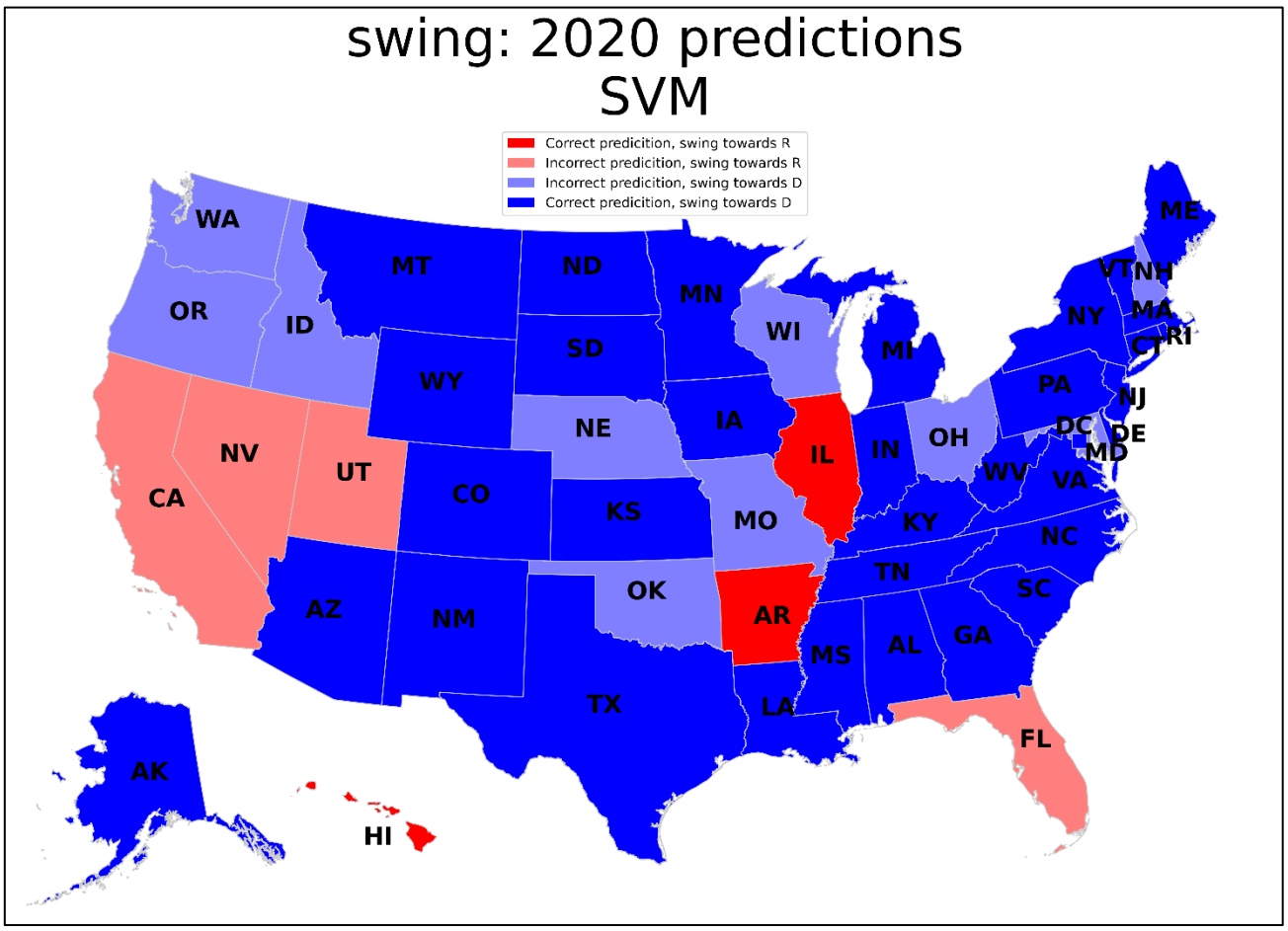
This model is obviously not useful, since it predicts all instances to be of class 1.

Some models with polynomial kernels were able to provide results somewhat better than random guessing, however, best SVM models did worse than best Logistic Regression models.



This is the output of the best SVM model for **winner**. It achieves the accuracy of 33, which is better than guessing or predicting only one class. However, it is obvious that a model incorrectly predicting a Republican win in California (CA) and a Democratic win in Wyoming (WY) and West Virginia (WV) in 2020 does not represent political reality well.

On the other hand, a SVM model with a polynomial kernel was able to achieve a relatively good result for **swing**.

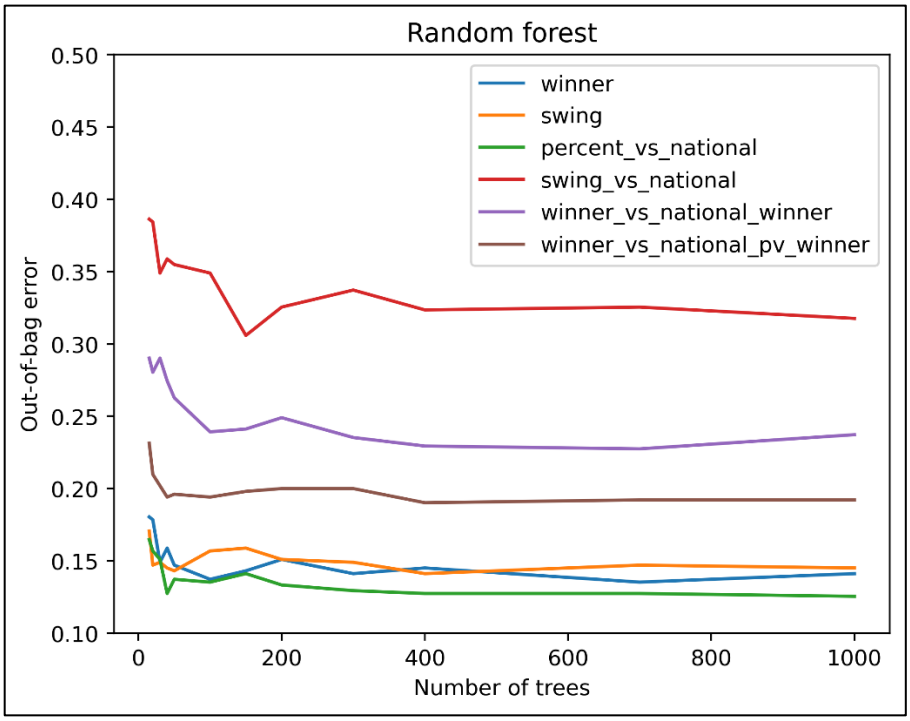


This model predicted 3 out of 7 class 0 instances and 34 out of 44 class 1 instances correctly, which amounts to a balanced accuracy score of 0.60. Curiously, this ended up being the best performance for **swing** across all models in the project.

Similarly to Logistic Regression models, SVM models were not stable in terms of connection between cross-validation and test scores. Relatively good cross validation scores did not guarantee good test scores: even for relatively more "successful" features, most hyperparameter combinations found through cross-validation led to poor results.

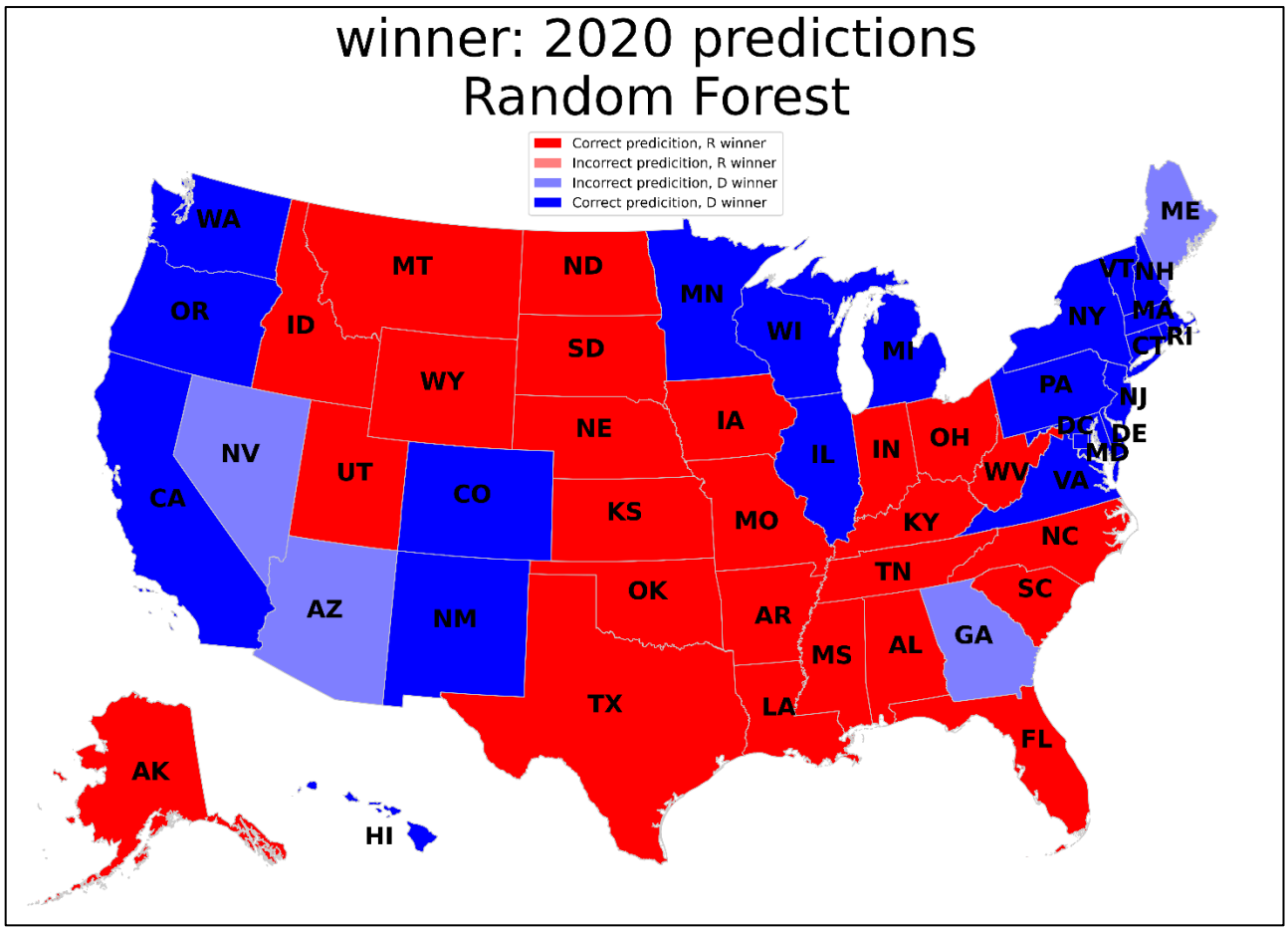
VI. Random Forest models

For Random Forest models, number of trees was the only hyperparameter tuned in this project. For all targets, various values of number of trees from 15 to 1000 were attempted. Since higher number of trees in Random Forest models usually doesn't lead to overfitting, there was no need for cross validation in this case, and model performance was estimated via out-of-bag error.



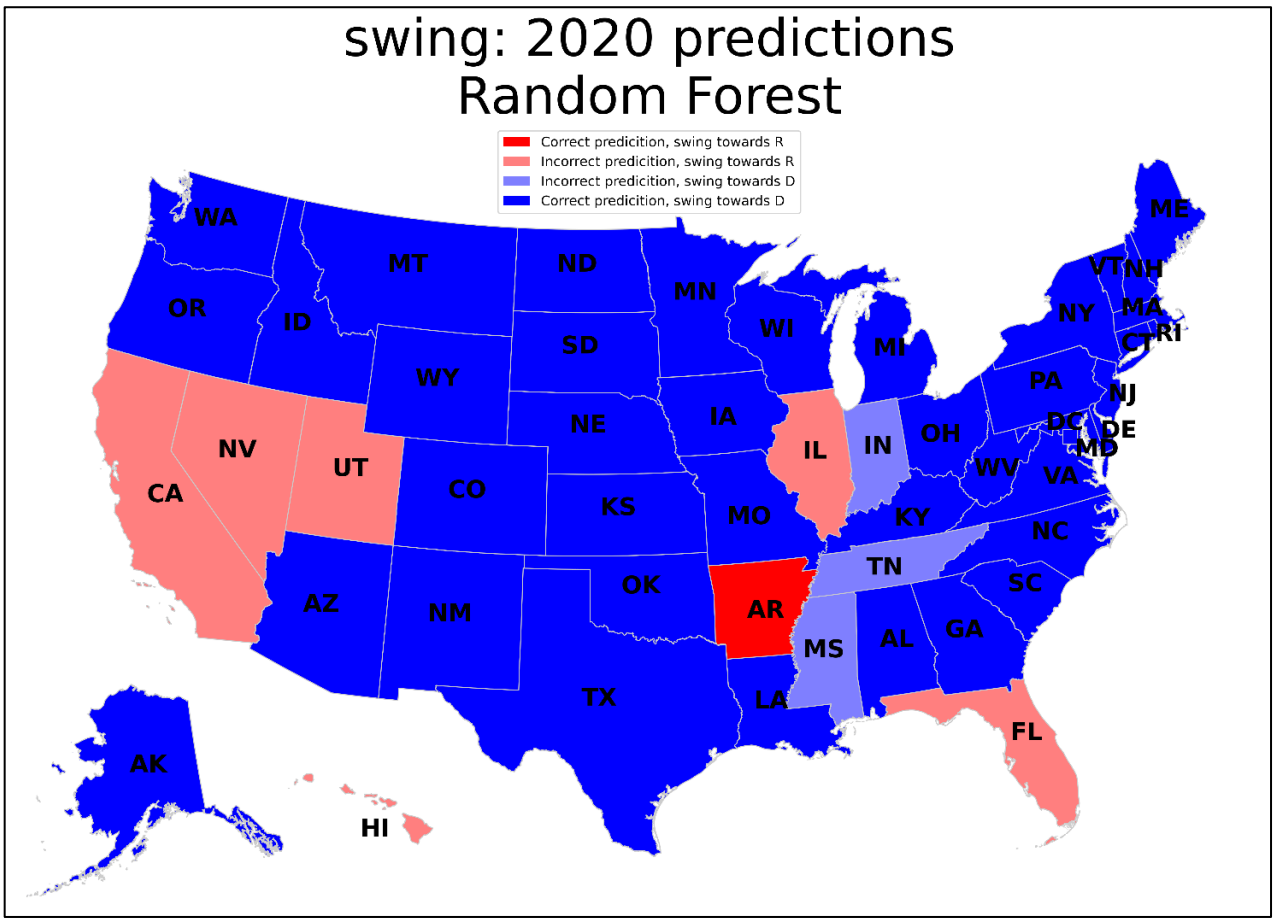
As can be seen on the graph above, for all targets optimal performance is achieved at around 400 trees. Note that resulting out-of-bag scores are significantly different from one target to the other. However, these differences do not exactly match the differences in Random Forest model performances on test sets.

For **winner**, best Random Forest performance on the test set was already achieved at 300 trees.



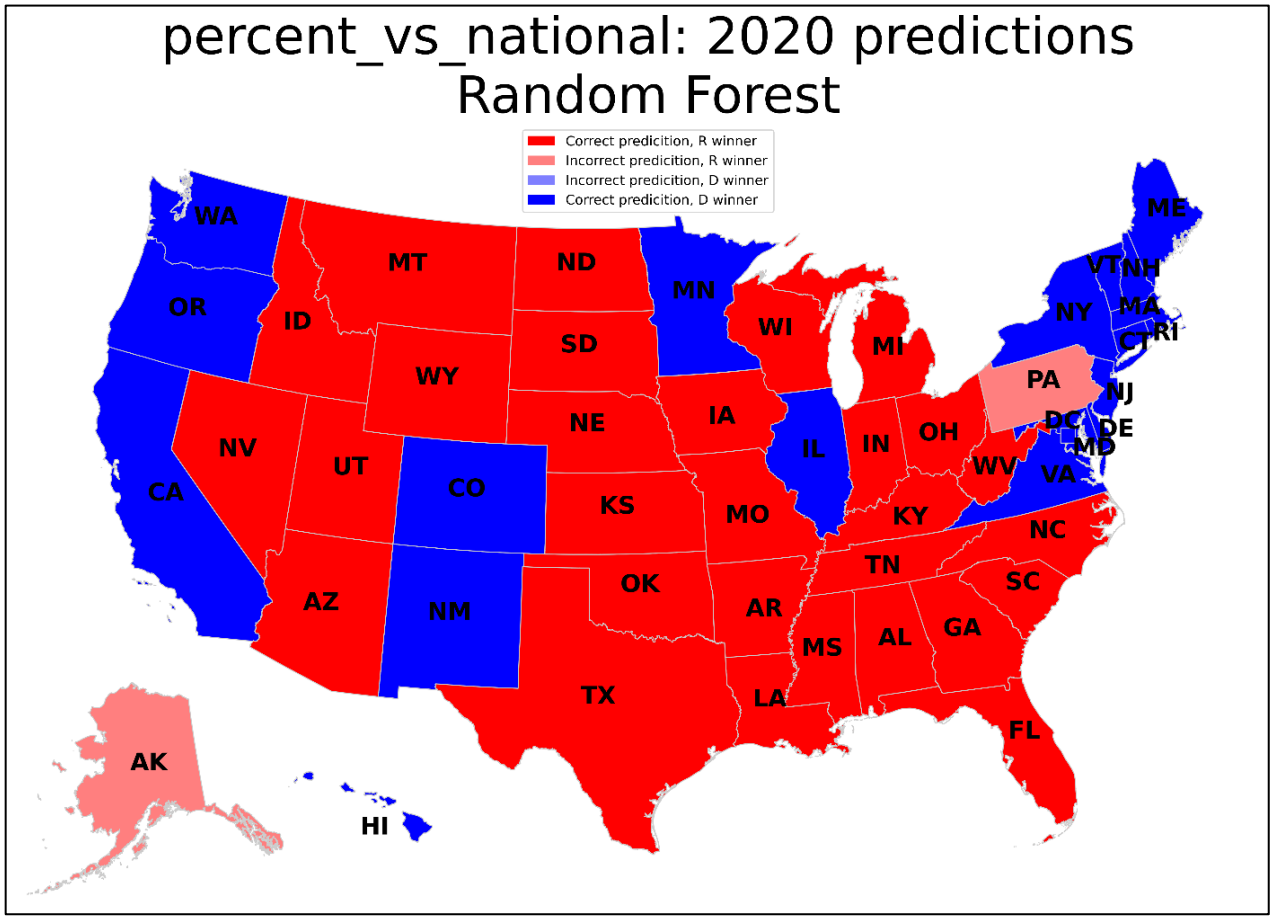
This model achieves accuracy of 47, which is definitely a good result, especially considering the states that the model got wrong. Georgia (GA) and Arizona (AZ) were the states with the two closest results in the 2020 presidential election. Democratic candidate Joe Biden won them by 0.24% and 0.31% of votes respectively. Nevada (NV) was also one of states with very close results, as Biden won it by 2.39% of votes. Only Maine (ME) was won by Biden with a relatively large margin of 9.07% of votes.

For **swing**, Random Forest models were unable to achieve good performance on the test set, even though their out-of-bag scores on the train set were low.



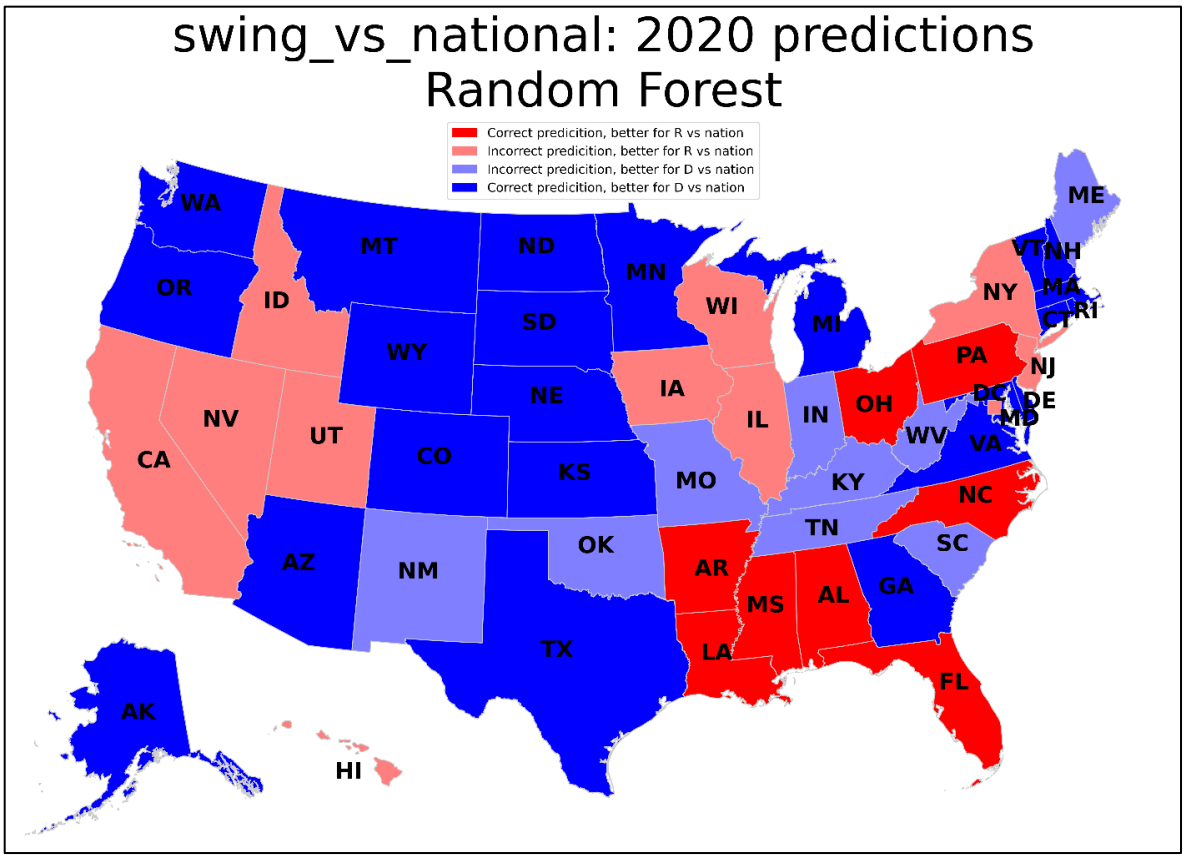
This model predicted 1 out of 7 class 0 instances and 41 out of 44 class 1 instances correctly, which amounts to a balanced accuracy score of 0.54. This is not a particularly useful model.

For **percent_vs_national**, a Random Forest model with 400 trees achieved very good performance.



This model has an accuracy of 49, getting only 2 states wrong. In my opinion, this is a remarkably good result considering how little information the dataset actually uses.

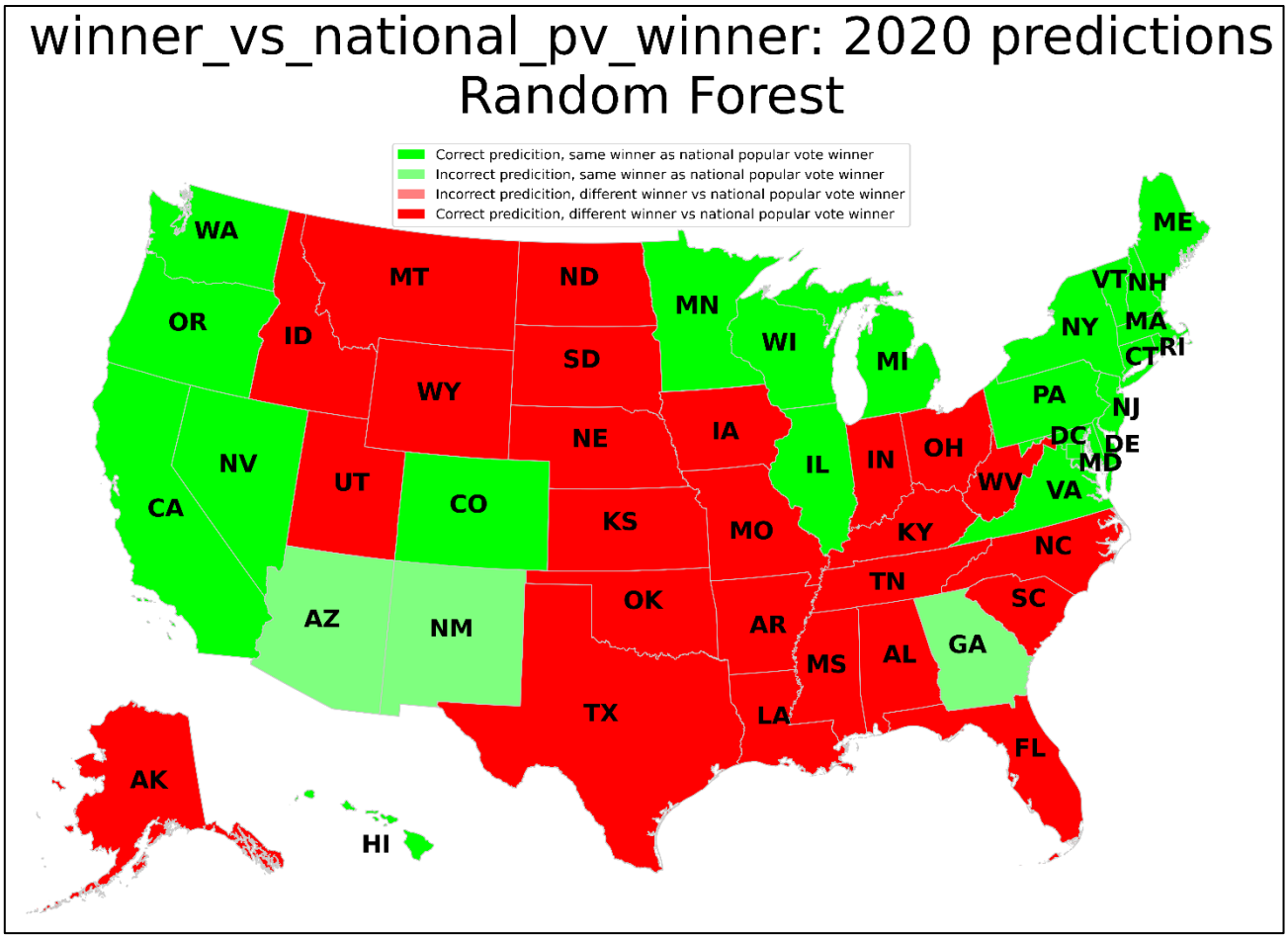
For **swing_vs_national**, the Random Forest model did not perform that well.



This model achieves accuracy of 31, which is not a great performance, however, it does better than Logistic Regression or SVM models for **swing_vs_national**, which either predict only one class or do as well as random guessing.

As for **winner_vs_national_winner**, the Random Forest model was unable to deliver useful predictions. Its performance was similar to the Logistic Regression model for **winner_vs_national_pv_winner** shown earlier, as it predicted one of the classes 50 times out of 51, despite the train set having a roughly equal class distribution.

However, very good performance was achieved on **winner_vs_national_pv_winner** using Random Forest.



This Random Forest model using 400 trees achieves the accuracy of 48. This is a very good prediction, considering that predicting **winner_vs_national_pv_winner** for every state requires that the model doesn't only capture the trends within the state, but the national trends as well (more on that later).

Overall, Random Forest models produced stable results in terms of hyperparameters for all targets, i.e. increases or small decreases in the number of trees either didn't change the output on the test set, or changed it minimally. For all targets, small out-of-bag error achieved on the train set generally meant good performance on the test set, with **swing** being the only exception.

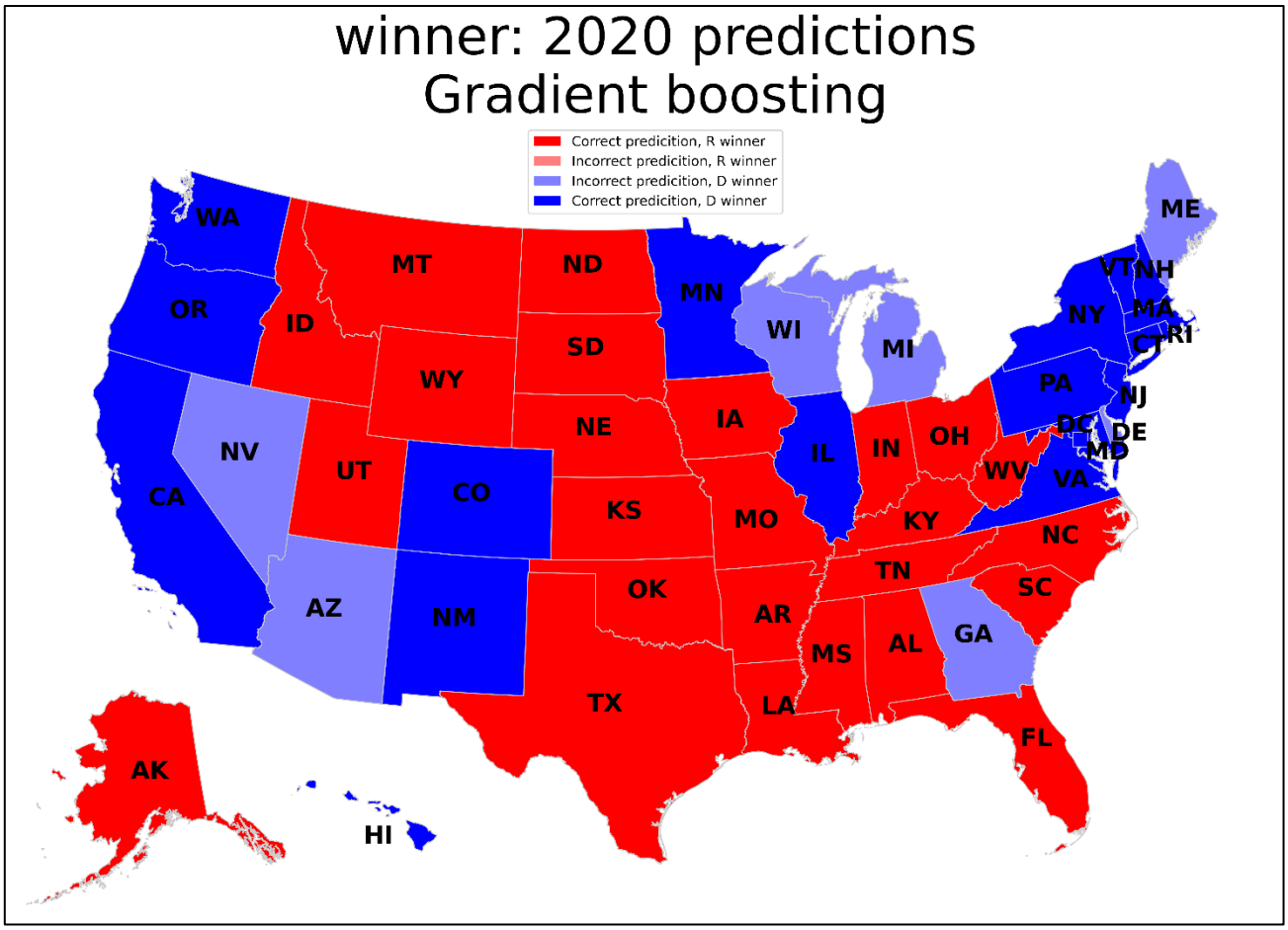
VII. Gradient Boosting models

In this project, Gradient Boosting was attempted for **winner**, **swing**, **swing_vs_national** and **winner_vs_national_winner**. As for **percent_vs_national** and **winner_vs_national_pv_winner**, very good results for these targets were already achieved via Random Forest. Since Gradient Boosting

models require significant computational resources, further improvements for these two targets were not attempted.

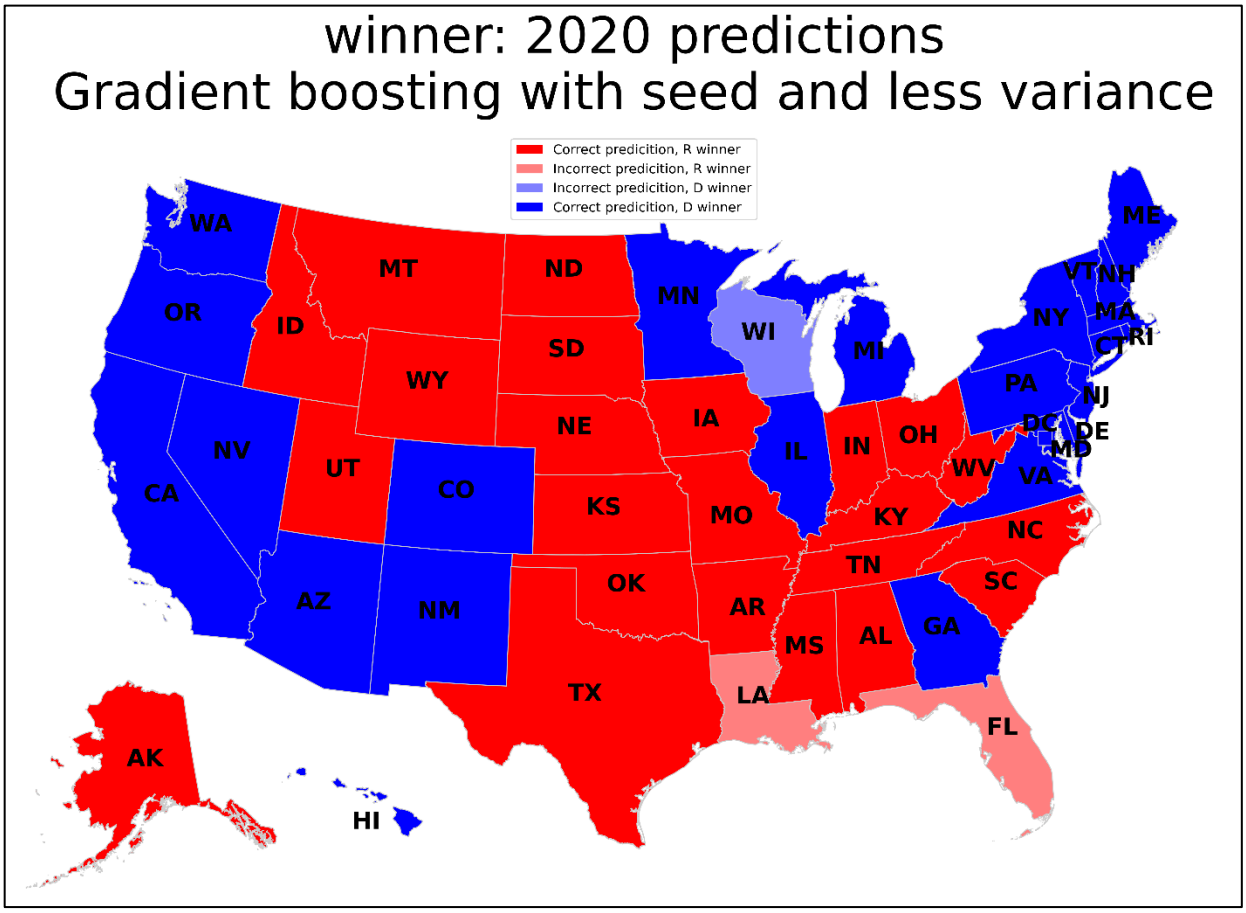
Scikit-learn's Gradient Boosting Classifier provides a large number of hyperparameters to tune. Due to computational limitations, grid search was attempted only over values of learning rate, number of estimators, and maximum depth of base estimators (trees). Overall, Gradient Boosting models produced stable results in terms of hyperparameters for all targets, i.e. or small changes of hyperparameter values either didn't change the output on the test set, or changed it minimally.

With hyperparameter values found during grid search, Gradient Boosting produced the following result for **winner**.

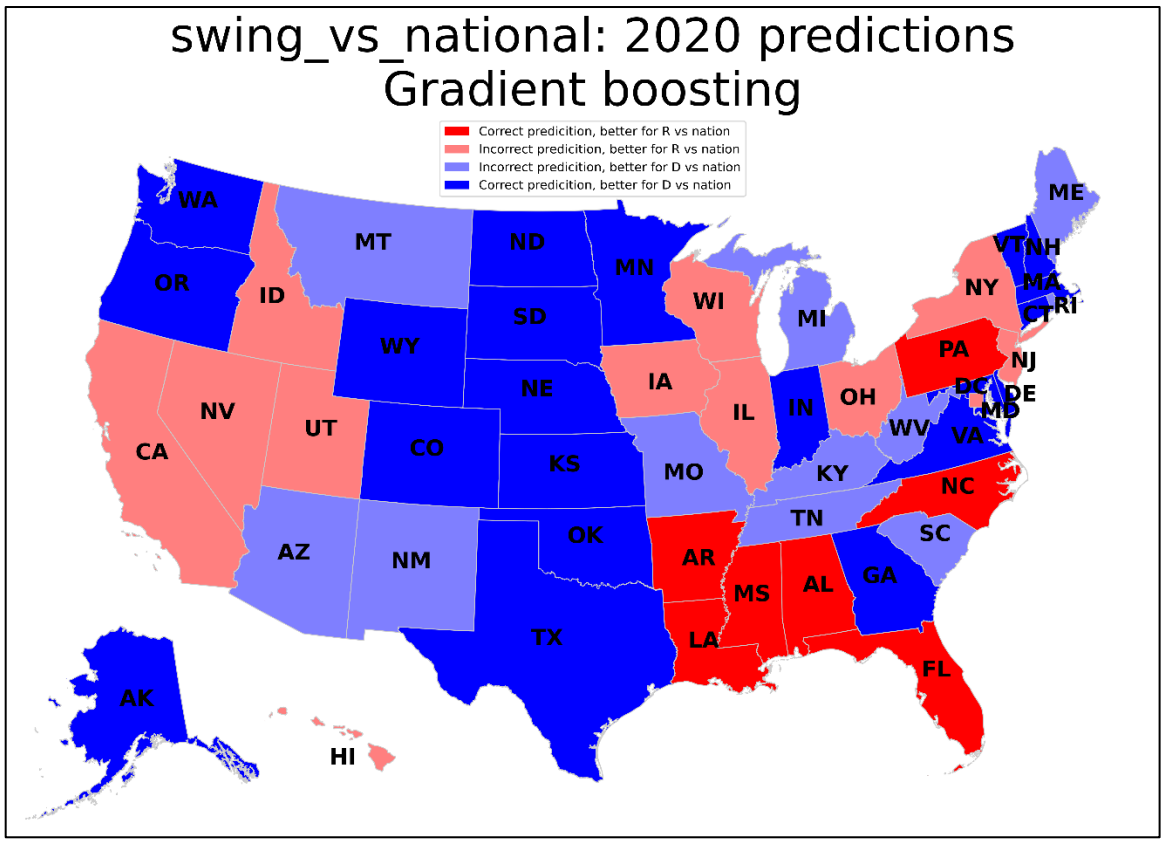


This model achieves accuracy of 44, which is lower than the Random Forest model for winner. This is not a very good result: the model misclassified 7 blue states as red, including states such as Maine (ME) and Delaware (DE), that voted Democratic by a large margin in 2020. If the states voted as predicted by this model, the Republican candidate would win the Electoral College as well.

Gradient Boosting is supposed to improve results of models such as Random Forest. In order to achieve this, values of some hyperparameters were changed to decrease model variance: learning rate was lowered, maximum number of features to consider during splitting in trees was lowered, and only a share of the train set was made available to every tree. To balance this out and decrease model bias, number of trees was increased. Also, initial predictions were made with the best SVM model for **winner** shown earlier, not with a dummy classifier, as is done by default. Through trial and error, the following model was found.

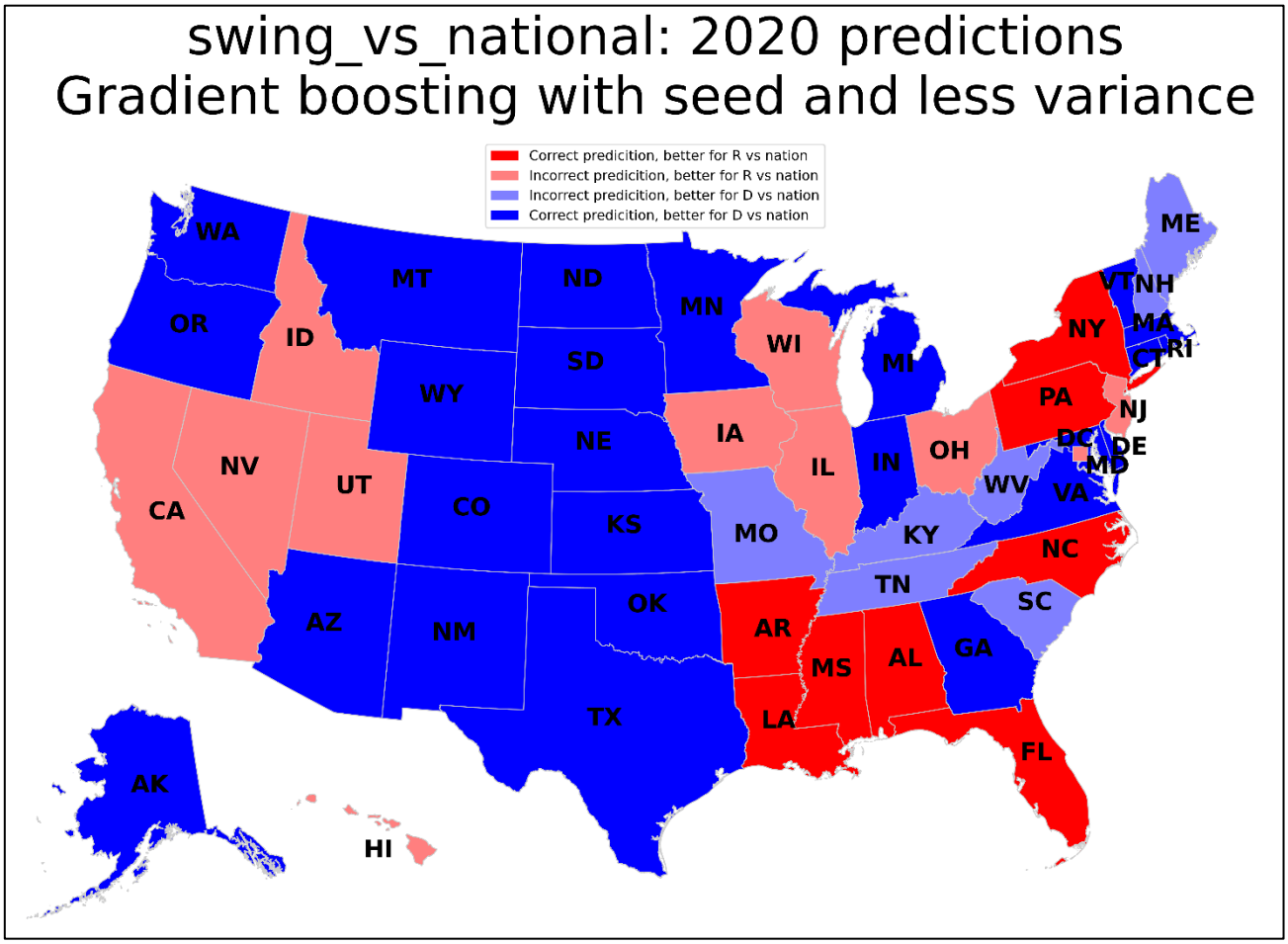


This model for **winner** achieves accuracy of 48, which is higher than any previous model for this target. It makes mistakes only on 3 states. While Wisconsin (WI) and Florida (FL) had close results in 2020, Louisiana (LA) didn't, since it was won by the Republican candidate with a big margin of 18.6% of votes. This in an outlier prediction for an otherwise good model.



For **swing_vs_national**, the model with the hyperparameters found during grid search did not improve results either. This model achieved accuracy of 28, which is barely better than random guessing, and worse than the Random Forest model for **swing_vs_national.**

In order to improve performance for **swing_vs_national**, values of some hyperparameters were changed to decrease model variance: minimum Gini impurity decrease for a split to occur was raised, and only a share of the train set was made available to every tree. To balance this out and decrease model bias, number of trees and maximum tree depth were increased. Also, initial predictions were made with the best Random Forest model for **swing_vs_national** shown earlier, not with a dummy classifier, as is done by default. Through trial and error, the following model was found.



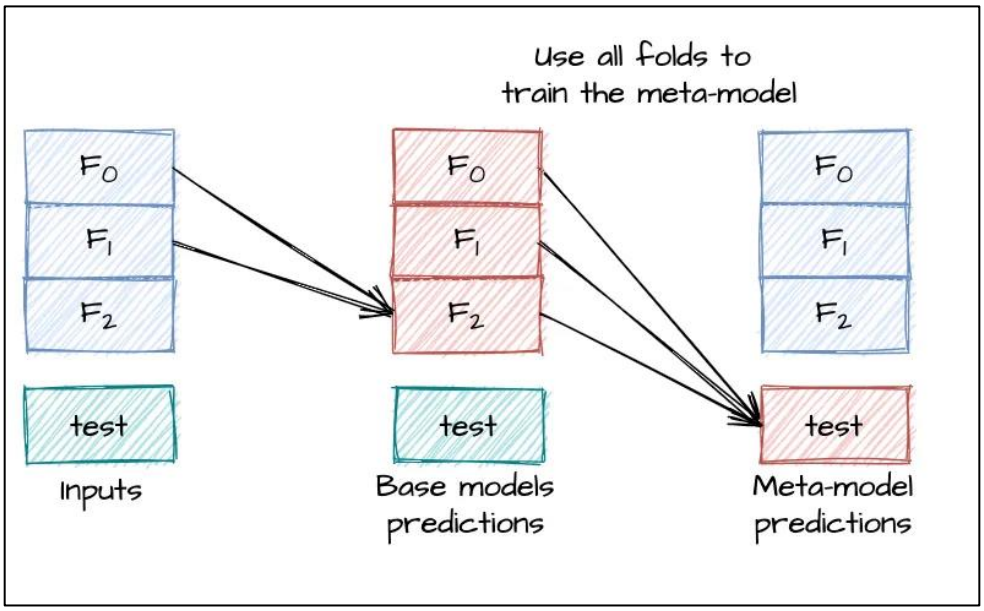
This model achieves accuracy of 33, which is slightly better than the Random Forest model for **swing_vs_national**. While this value of accuracy is not very high, perhaps this is still a good result, considering that the out-of-bag errors for Random Forest models were the highest for this feature.

As for **winner_vs_national_winner**, use of Gradient Boosting did not produce a meaningful model. The model predicted class 1 only on 3 instances out of 51, and on 1 of those 3 instances it was incorrect. This performance is only marginally better than with other types of models, all of which predicted at least 50 out of 51 instances to be of the same class. Attempts to alter hyperparameters manually did not lead to significantly better results for **winner_vs_national_winner**.

In case of **swing**, Gradient Boosting models with all hyperparameter combinations, either from grid search or picked by hand, did worse than Random Forest models or even the best SVM model.

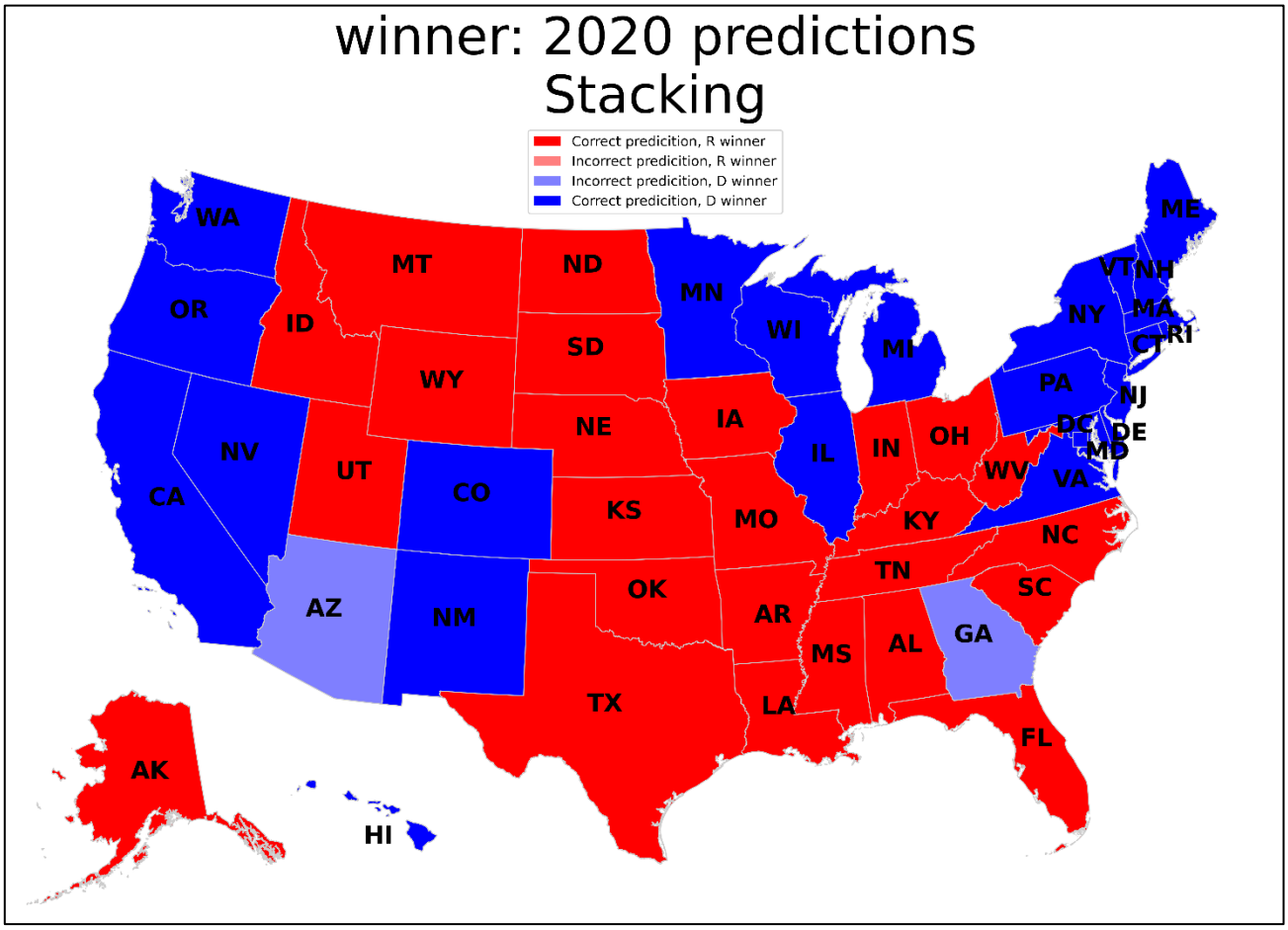
VIII. Model Stacking

In this project, model stacking attempted only for the two targets where performance improvements seemed necessary and possible: **winner** and **winner_vs_national_winner.**



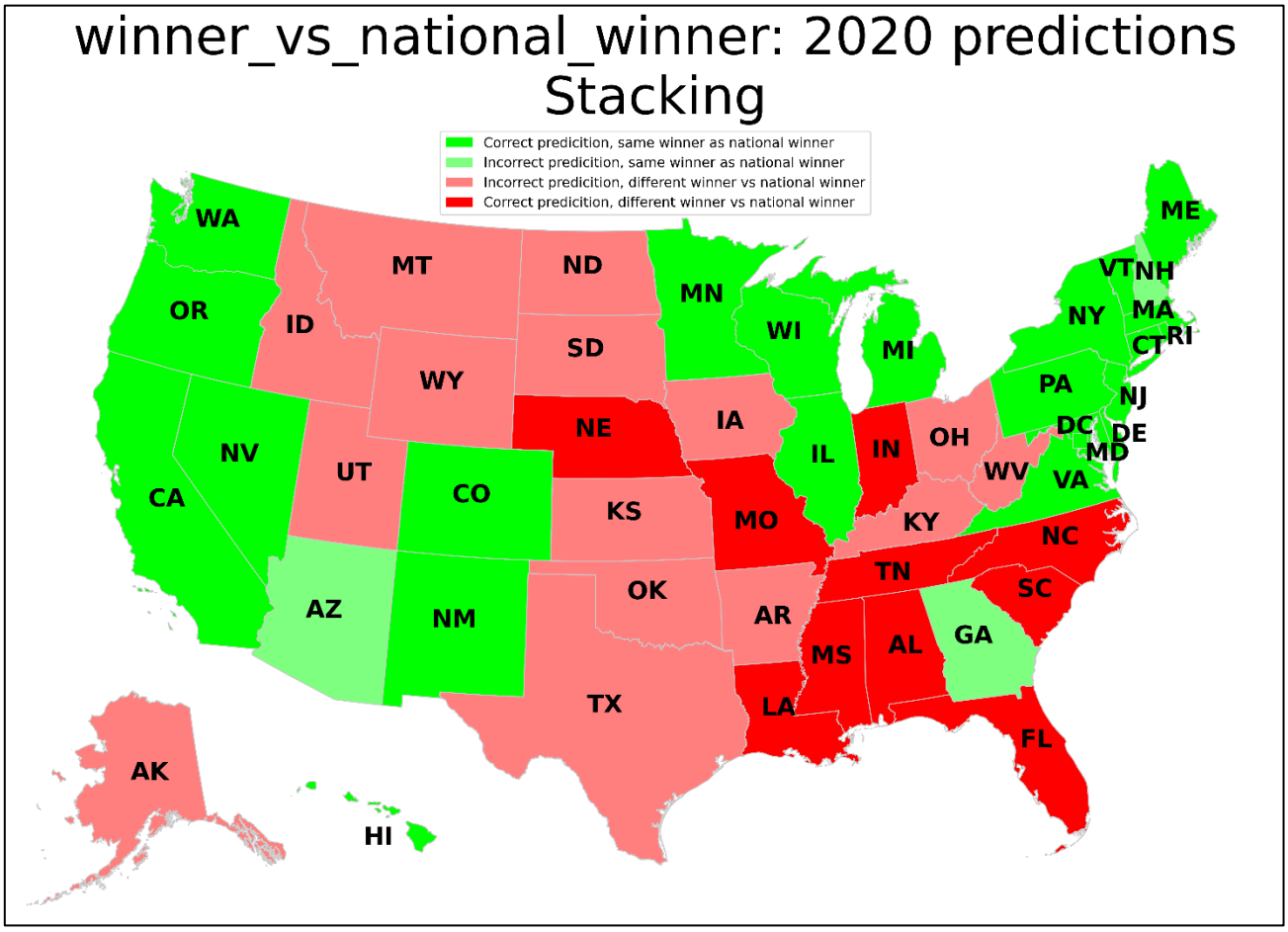
In order to decrease the chance of overfitting, model stacking was done as shown on the above image. First, a number of base models were selected among those already built earlier in the project. For both targets, the train data was split into 3 stratified folds. Each fold was predicted by multiple base models trained on the other two folds. Then, the meta-model was trained on the predictions of the base models for all 3 folds. Then all the base models were trained on all the train data and predicted the test data. Finally, the meta-model used those predictions to predict the test data, producing the final output of model stacking. Unfortunately, due to computational limitations, no cross validation of the meta-model was done.

For **winner**, a total of 9 base models were selected (2 Logistic Regressions, 3 SVMs, 2 Random Forests, 2 Gradient Boosting models). This selection included all the best-performing models for **winner** in their own type, as well as some suboptimal models. For the meta-model, Linear SVM was used. The following results were achieved.



This model achieves accuracy of 49, getting only 2 states wrong. This is a remarkably good result. It is a better result than those achieved with any single base model. Furthermore, the two states that it got wrong were the states with the two closest results in the 2020 presidential election. The Democratic candidate Joe Biden won Georgia (GA) and Arizona (AZ) by only 0.24% and 0.31% of votes respectively. Had the votes been just slightly different, this prediction would be perfect.

For **winner_vs_national_winner**, a total of 9 base models were selected (2 Logistic Regressions, 3 SVMs, 2 Random Forests, 2 Gradient Boosting models). Since all the previously built models for **winner_vs_national_winner** predicted the vast majority of the instances in the test set to be of the same class, models of different types with roughly half of them predicting all 0s, and the other half predicting all 1s were selected. For the meta-model, Logistic Regression was used. The following results were achieved.



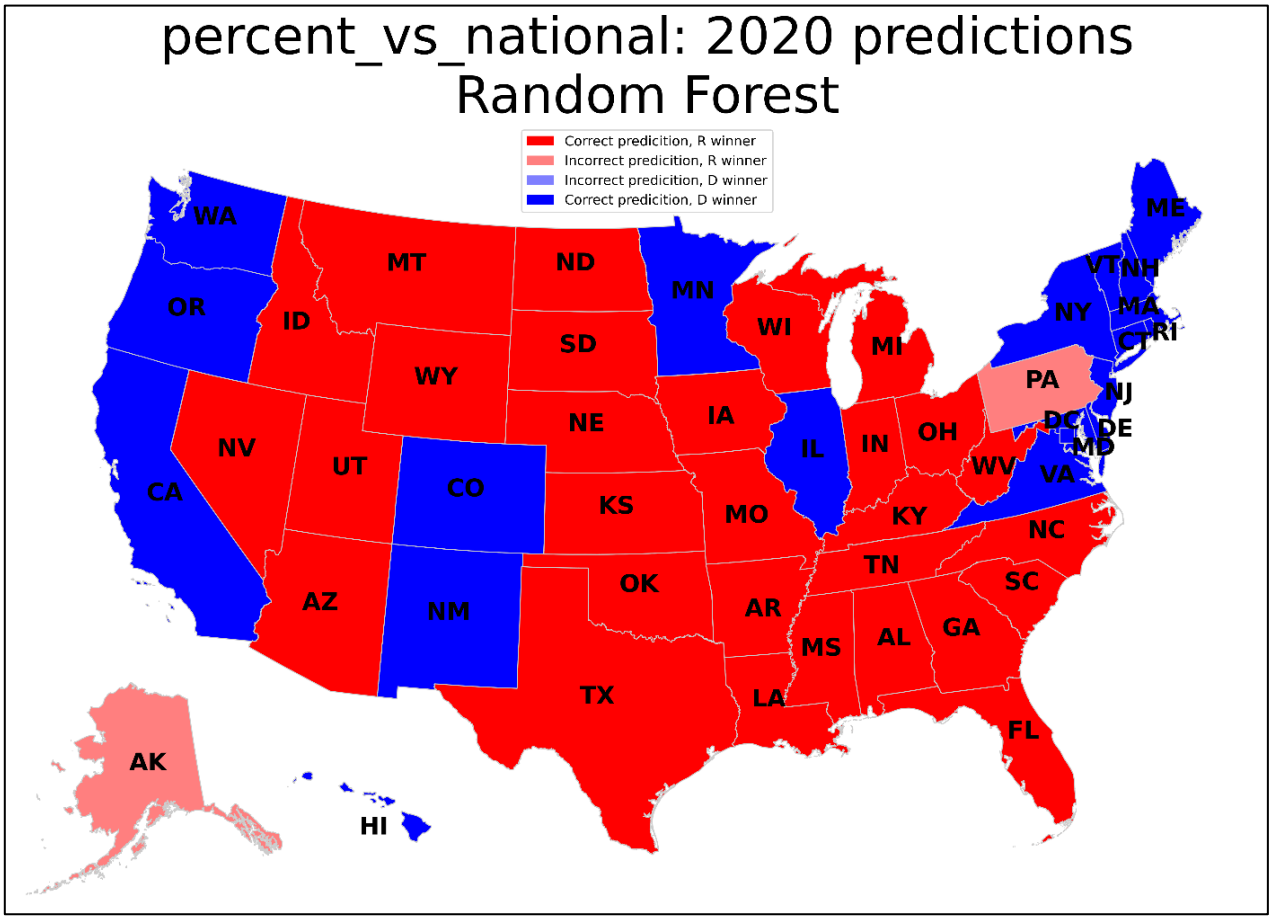
This model achieves accuracy of 33. While it still predicts mostly class 0, it gets many instances of class 1 right and it certainly does better than random guessing. This is a big improvement over the base models for **winner_vs_national_winner**, neither of which were informative by itself.

Overall, Model Stacking produced stable results in terms of selection of meta-model type for both targets, i.e. changes to the meta-model type or small changes to its hyperparameters did not lead to a big decrease in performance.

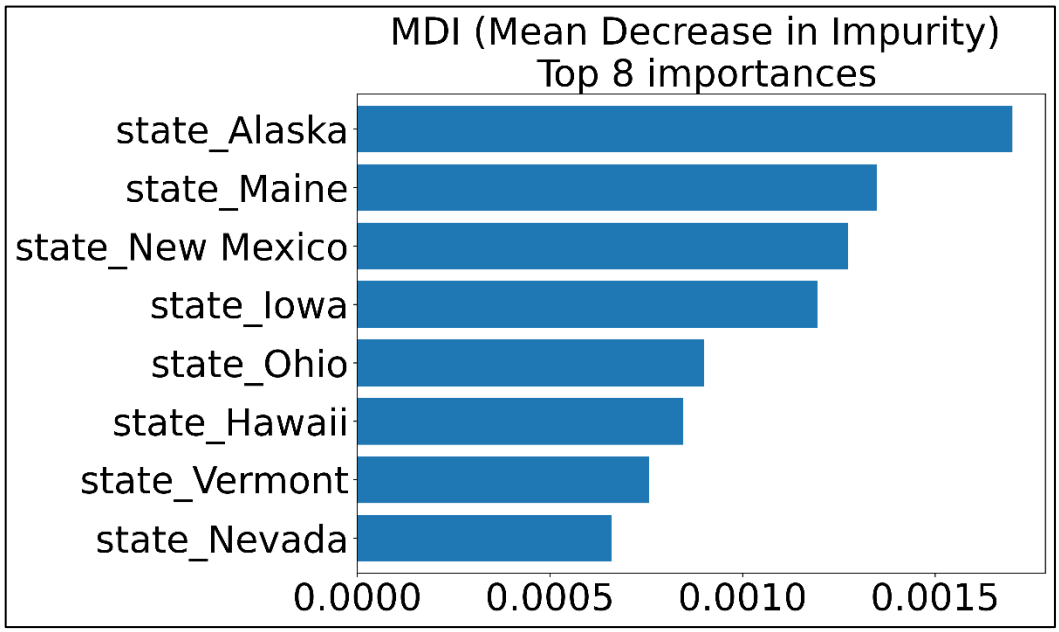
IX. Model Explanations

Many well-performing models were constructed during this project. It is interesting to ask how did they capture some relationships so well. However, this question is not easy to answer, since the best-performing models built in this project are Random Forests, Gradient Boosting models and model stacks, which can be considered black-box models. Also, the dataset contains many variables that are not independent from one another: obviously, economic features are all interdependent, as well as dependent on the state variables. This violates the feature independence assumption that is a part of most of the black-box model explanation methods. Keeping that in mind, these methods were attempted anyway.

For model explanations, the well-performing Random Forest model for **percent_vs_national** was selected. It achieves accuracy of 49 on the test set. Its output is presented below.



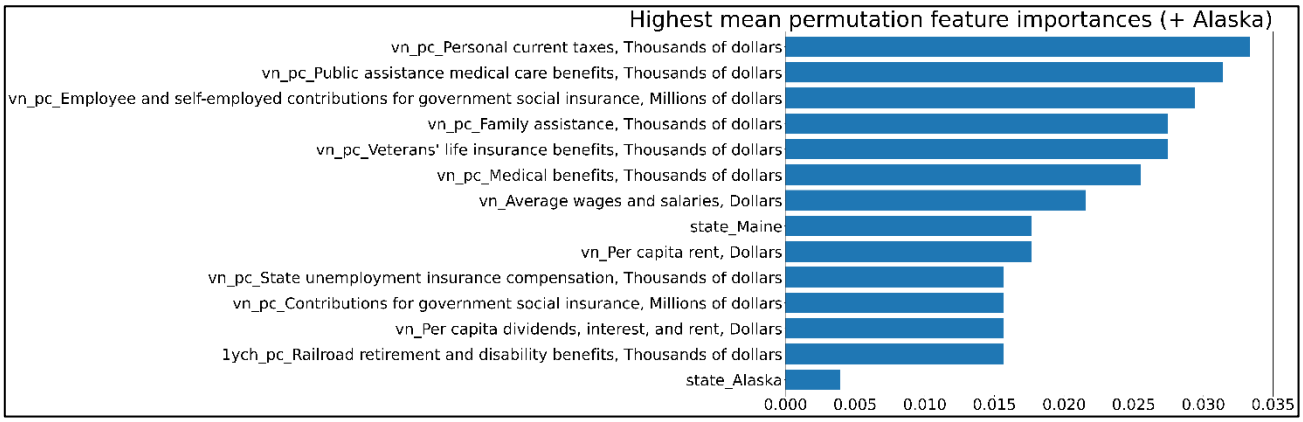
First, the Gini or Mean Decrease in Impurity (MDI) importance was used. According to scholars, MDI importance can be biased towards continuous features against categorical features. Therefore, only MDI importance of categorical features (state and political dummy variables) was analyzed. Most of these features had some importance, however, the dummy variables pointing out Florida, North Dakota, Alabama, New York, New Hampshire and Montana had MDI importance values of 0. That doesn't necessarily tell us much: maybe, the identities of those states were not important at all, maybe, the model learned the identities of those states via economic features, or maybe, those features were used in some splits that themselves didn't lead to impurity decrease, but really helped to make further splits better.



The top 8 most important categorical features are shown on the above chart. These high scores actually tell more about the features than low scores. High MDI importance means that the identities of these states were not learned by the model from economic features, at least not to the extent as other states were. Perhaps, these states have historically voted differently from the other states with similar economic policy or economy performance. It is curious that Alaska (AK) is on top of this rating, which is based on the test set, as well as one of only two states that the model got wrong on the train set.

As for political features, all of them had some importance, but not as much as I expected them to before attempting model explanations.

Also, permutation feature importance was calculated for all features using the test set. After 10 repeats, only 53 features had non-zero importance: 51 economic features and the dummy variables for the states of Alaska and Maine. This is interesting, since these two features were the most important categorical features according to MDI importance. These results don't mean that other features didn't matter for this model at all: with more repeats, more features appear to have non-zero permutation importance. However, it does point out the most important features for prediction for this particular model.

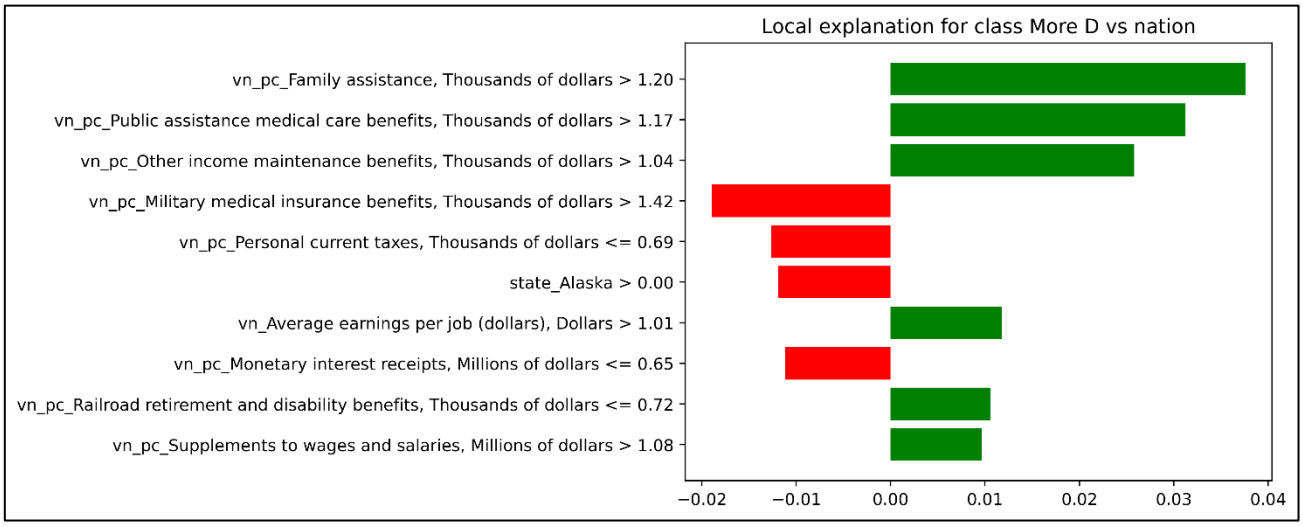


The highest permutation importance values, as well as the permutation value for the Alaska dummy variable are shown above. The Maine dummy variable was one of the most important variables by itself, while Alaska's dummy variable was not that important. As for the economic features, it is not surprising to see that things like personal taxes, medical care assistance benefits and family assistance used in this model, since they seem to be important to the voter.

It is worth noting that features with the highest importance for the model will always be features that separate Democratic-leaning and Republican-leaning states the best, since that is what impacts the **percent_vs_national** value the most. Therefore it's not surprising to see that the economic features with the prefix **vn_**, describing the state values in relation to the national value, are the best predictor here. It is expected that Democratic-leaning states generally have higher wages and higher tax rates, therefore, more taxes collected per capita, than Republican-leaning states.

It should also be noted that no single feature had a particularly big permutation importance value. It point to a large amount of correlation between the features in the dataset.

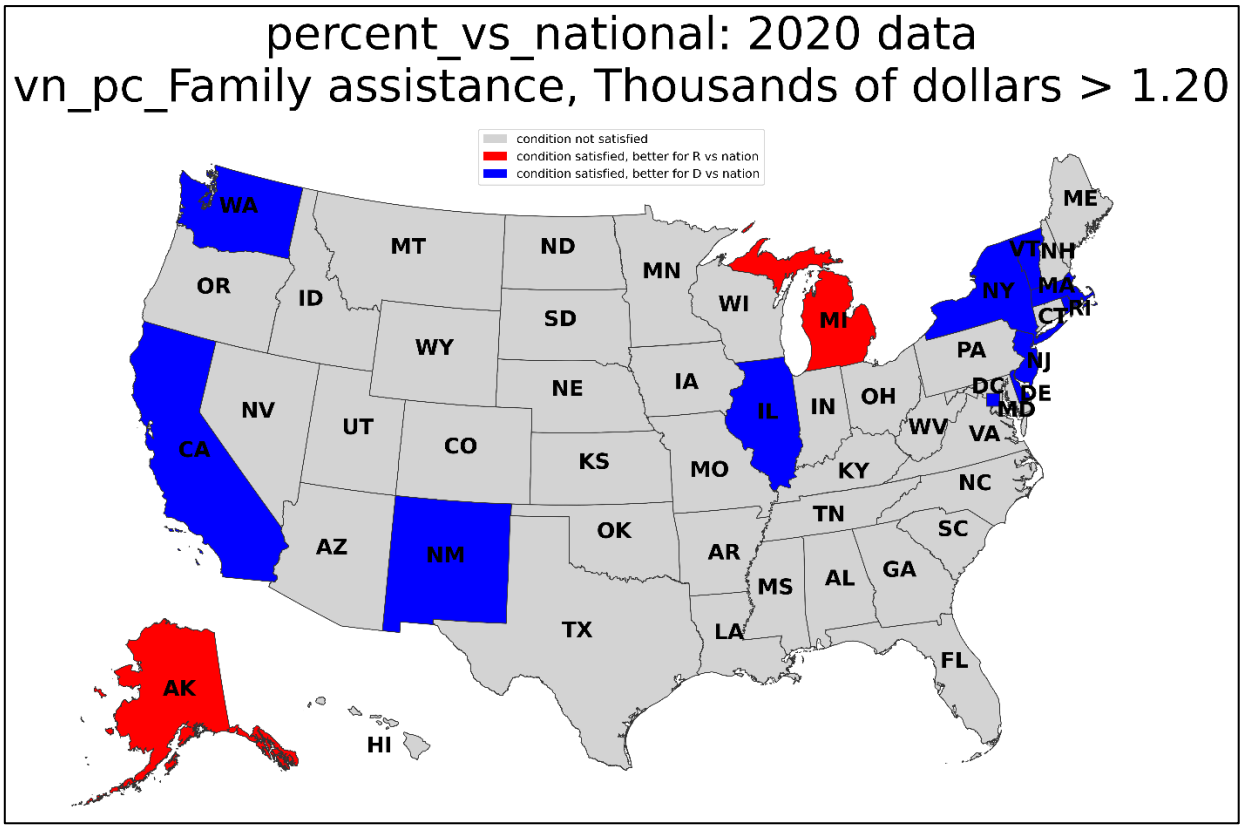
Finally, LIME was used to explain two particular predictions of the model: one correct and one wrong. For Alaska, which was predicted wrongly, LIME produced the following result.



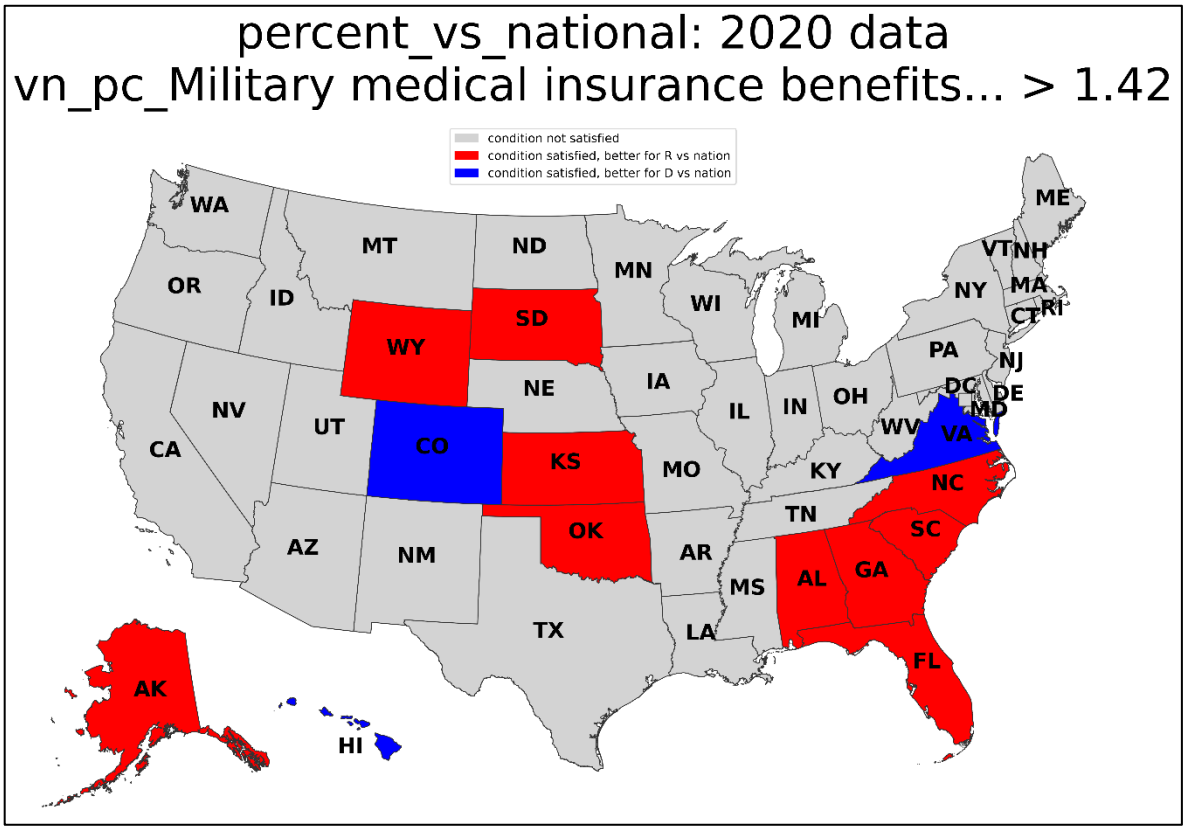
It is worth noting that LIME is unstable, and slightly different results can be expected when running the explainer again. However, the most important splits on top of the chart stayed the same throughout many runs of LIME.

Also, it is worth pointing out that the most important economic features in this LIME result also have high permutation importance values.

This LIME result partially explains why the model got Alaska wrong. This state has high levels of family assistance, medical care benefits etc. per capita in comparison to the rest of the U.S., which seems to be typical of a Democratic-leaning state. The Alaska dummy variable was important and "pushed" the prediction towards Republican-leaning, since the state of Alaska itself has been Republican-leaning historically. Even though the Alaska dummy variable was really important according to both MDI and permutation importance, perhaps, it should have been even more important, i.e. used in more splits, to compensate for Alaska's similarity to Democratic-leaning states in terms of economic features.

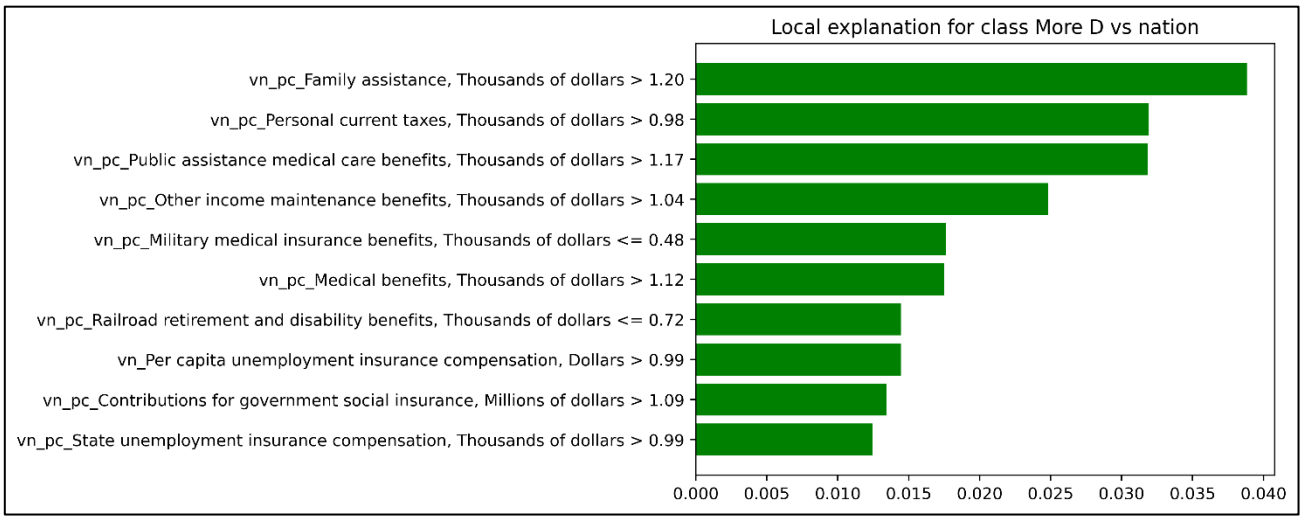


To further illustrate this point, the above map shows the states for which the top condition from the LIME explanation of the Alaska result is satisfied in 2020. This condition is satisfied for 11 Democratic-leaning states and only 2 Republican-leaning states, of which Alaska is one. Higher family assistance amounts per capita tend to be given out in Democratic-leaning states.

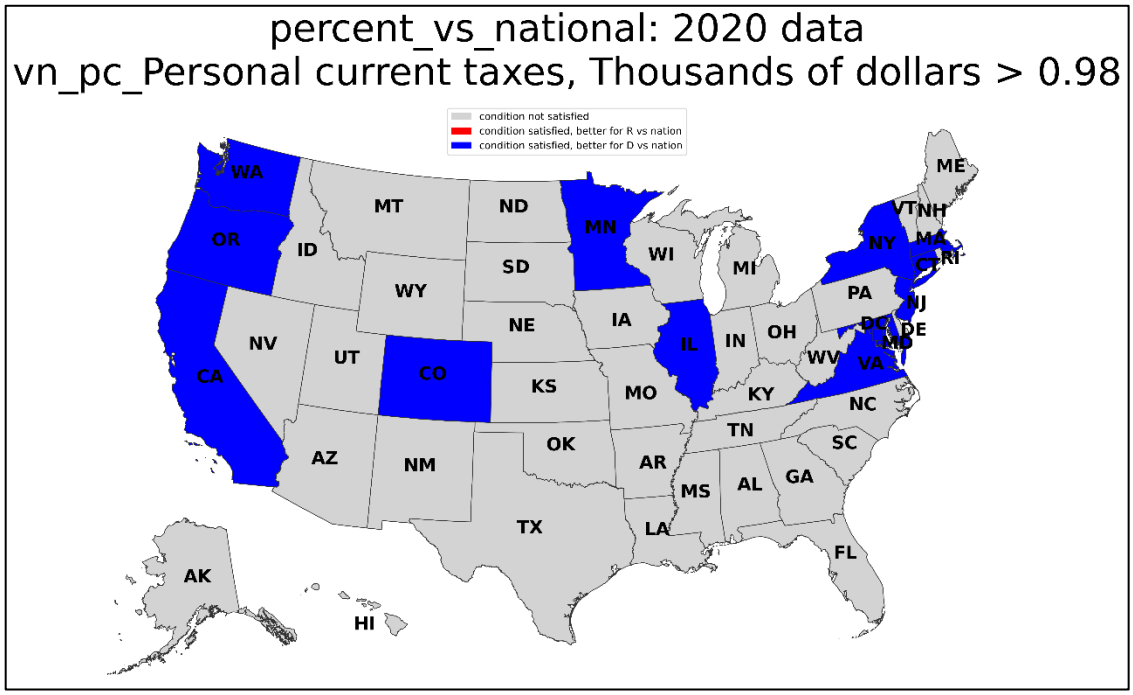


This map shows the states that satisfy the condition that pushes the LIME Alaska prediction towards Republican-leaning. Indeed, among states that receive much more military medical insurance benefits per capita than the U.S. in general, only 3 are Democratic-leaning and 10, including Alaska, are Republican-leaning. This could be explained as these states just having more members of the military living in them, and members of the military leaning towards Republicans.

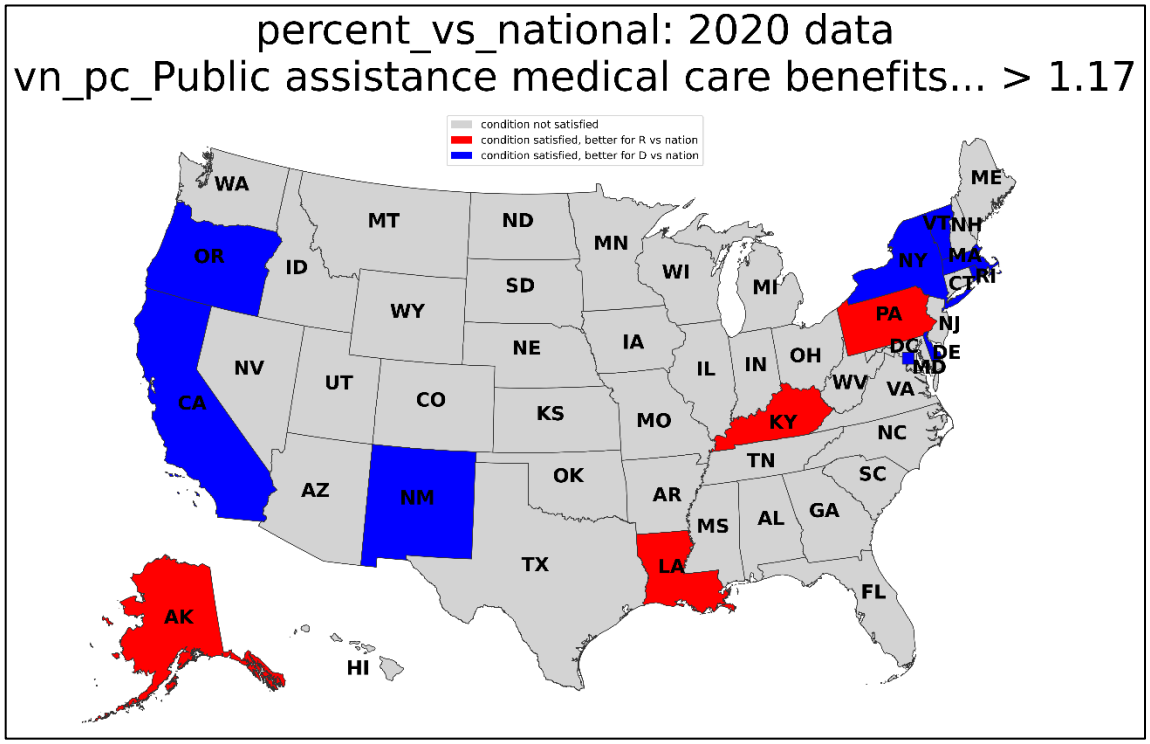
Another prediction that was explained by LIME was the District of Columbia. The LIME results for D.C. are shown below.



D.C. is the most heavily Democratic-leaning state-level jurisdiction in the U.S. In 2020, it was won by the Democratic candidate Joe Biden with over 92% of the votes. Therefore, it's not surprising that all most important LIME explanations indicate that D.C. leans Democratic. Many of the features in this explanation also have high permutation importance values.



The above map shows the personal current taxes threshold. On 2020 data, this is a very good predictor of the result, since all 13 states where high amounts of personal taxes per capita were collected are Democratic-leaning.



The above map shows the public assistance medical care threshold. On 2020 data, this is a good predictor of the result, since among states with public assistance medical care 9 were Democratic-leaning and only 4 were Republican-leaning.

X. Results summary

Modeling results achieved in this project are summarized in the following table. Model performance for each feature is color coded. The numbers in the table are the amounts of correct predictions on the test set of 51 samples, apart from **swing**, for which the adjusted accuracy score on the test set is shown. Also included is the evaluation of hyperparameter stability, i.e. whether performance of models of these types stayed exactly or almost the same after small changes to their hyperparameters. Note that 'CV Boosting' refers to Gradient Boosting with hyperparameters found through cross validation, and 'Boosting' refers to Gradient Boosting with hand-picked hyperparameters.

Targets \ Models	Logistic R	SVM	RF	CV Boosting	Boosting	Stacking
winner	45	33	47	44	48	49
swing	poor	0.60	0.53	poor	poor	-
percent_vs_national	44	35	49	-	-	-
swing_vs_national	poor	poor	31	28	33	-
winner_vs_national_winner	poor	poor	poor	poor	poor	33
winner_vs_national_pv_winner	poor	poor	48	-	-	-
Hyperparameter stability	no	no	yes	yes	yes	yes

Color						
Performance	No predictive power	Minimal predictive power	Some predictive power	High predictive power, not the best model for the feature	High predictive power, the best model for the feature	Not attempted

As for model explanations, the well-performing Random Forest model for **percent_vs_national** was explained via MDI importance, permutation importance and LIME. Some features were found to have been important through multiple methods. It was found that economic features describing the state values in relation to the national value were the most important in predicting **percent_vs_national**. Some particular economic indicators that best described whether the state was leaning Democratic or Republican were personal current taxes, public assistance medical care benefits, family assistance, and military medical insurance benefits. At the same time, it was found that no single feature had a high importance metric.

XI. Results analysis and conclusions

Comparing types of models to each other, it is clear that Logistic Regression and Support Vector Machines are not suitable for any of the features. Vast majority of models of those types performed poorly on the test set, even if they had high cross validation scores. This is hardly surprising, since linear-based models or models that depend on the distance between data points in the feature space tend to do poorly when the feature space has a high number of dimensions, i.e. when the dataset has a lot of features, and the created dataset had 333 features. The few Logistic Regression and SVM models that seemed to have predictive power likely achieved relatively high scores by chance, since their results became much worse after a slight change of hyperparameters. This instability means that when faced with unseen data, these models will probably produce bad results, since the unseen data is likely to have a slightly different distribution due to relatively low sample sizes, which are inherent to the problem of predicting results of U.S. presidential elections by state.

All the other types of models were stable in terms of hyperparameters and therefore can be used for predicting the targets they did well on.

Random Forest models were clearly superior to Logistic Regression and SVM models and did very well on 3/6 targets and somewhat well on **swing_vs_national**. In fact, Random Forest results for **percent_vs_national** and **winner_vs_national_winner** were so good that further improvements via Gradient Boosting or Stacking were deemed unnecessary. Overall, for most targets Random Forest models proved to be good baseline models, since they required much less computational resources and produced only slightly worse results than Gradient Boosting or Stacking.

Cross validation for Gradient Boosting was the most computationally demanding part of the project and, ironically, it did not lead to any improvements over the Random Forest results. This could be due to either the scope or the method of cross validation being chosen wrongly. The scope of cross validation could certainly have been the problem. Scikit-learn's GradientBoostingClassifier has many possible hyperparameters, and only a minority of them could have been checked due to computational limitations. The method of cross-validation could have been a problem too: repeated stratified train-validation splits were used instead of the more common stratified K-folds. Different cross validation approaches are proposed in the last section of this report.

Gradient Boosting models with hand-picked hyperparameters did improve on the Random Forest results. For both **winner** and **swing_vs_national**, improvements were achieved after decreasing model variance via slower learning and making the model more stochastic (by making only a part of the dataset available to each base estimator), and then compensating for increased model bias by either increasing the number of estimators, or increasing the complexity of base estimators. Generally, it was observed that lowering the learning rate and increasing the number of estimators at the same time either improved or at least did not worsen the results.

The successful application of Model Stacking in this project shows that it can be used to fix bad predictions, such as in case of **winner_vs_national_winner**, or further improve good predictions, such as in case of **winner**. However, it has to be noted that in this project due to computational limitations the meta-model type was not found through cross-validation, but rather picked by hand (other meta-model types attempted would usually produce a result similar to the best-performing base model). Therefore, it is possible that these Stacking meta-models were just overfit on the particular test set. Since due to the nature of the problem no other test set was available, some method of cross-validation should be used to make sure that these Stacking meta-models really perform well.

Comparing targets to each other, it is clear that no really good results were achieved for the targets encoding the changes in voting patterns: **swing** and **swing_vs_national**. As for **swing**, this was always going to be difficult because of the mismatch between the class distributions in the train and test data. The unbalanced distribution in the test set is typical for this target: in most previous years, most states swung in one direction, usually away from the party of the incumbent president (although that was not always the case). Since the assumption that the target is distributed similarly in the train and test sets is essential to all machine learning models, I'm not sure that it is possible to construct a good model for **swing**. Perhaps, a different target, like "swing versus incumbent party" could have been tried, but I'm not sure how useful these predictions would be, since, again, most states (up to 49 in some elections) swing away from the incumbent party anyway.

On the other hand, **swing_vs_national** had a relatively equal distribution in both train and test sets, and still no model was able to capture the relationship well. Perhaps, adding more data to the model would have helped, however, it's not easy to find objective, quantifiable and relevant data by U.S. state going back to at least 1980. Prediction of this target could be improved by adding past election results as features to the model, which wasn't done in this project due to the possibility of overfitting on such data.

For both **winner** and **percent_vs_national**, good predictions were achieved. This is not particularly surprising, since among the 50 states and D.C., more than half of the jurisdictions are actually not competitive, which means that they are very likely to vote in favor of a certain party regardless of the economic situation. For modeling, this is useful, since this large amount of "obvious" cases helps to filter out the models that don't really capture the relationship. On the other hand, it diminishes the potential usefulness of the model: a person closely following U.S. politics would probably guess around 45 of the answers correctly before a presidential election without any model. Therefore, a model needs to predict these targets almost perfectly to be useful. It is remarkable that some models for **winner** and **percent_vs_national** in this project did achieve nearly perfect accuracy.

What was surprising however, is that according to various model explanation methods, at least the Random Forest model for **percent_vs_national** relied mostly on the economic features to achieve these very good results. When adding states' dummy variables to the dataset, I was worried that the models would mainly use them and not the economic features, but that probably wasn't the case, and the constructed Random Forest model for **percent_vs_national** was useful not only for prediction, but also for interpretation of the link between economic features and partisan lean of the states.

It was interesting to see how much easier it was to model **winner_vs_national_pv_winner** in comparison with **winner_vs_national_winner**. Random Forest models produced very good and stable results for the former, while the latter could only be predicted with some accuracy with the use of Stacking. It is worth noting that the only difference between these two features is that their values for 2000 and 2016 are opposite, since these were the two presidential elections (in the period from 1980 to 2020) where the winner of the popular vote did not win the Electoral College. It is worth noting that the Electoral College vote counts are known before the election, and the **winner_vs_national_winner** can be inferred directly from the targets such as **winner_vs_national_pv_winner** or **winner**. Therefore, it's not necessary to predict **winner_vs_national_winner** directly. However, it was still useful, as the discrepancy between these modeling outcomes clearly shows that national popular vote winners are more easily inferred from economic data than Electoral College winners.

It must also be noted that for a human, predicting targets such as **winner_vs_national_pv_winner** is much more difficult than just predicting **winner**, since not only the winner of the election in the state, but also the national popular vote winner needs to be guessed correctly (or the correct answer could be stumbled into by guessing both wrong). The large number of non-competitive states does not help here: their winner would be obvious, but the answer would still depend on who won the national popular vote. Due to this, the fact that the Random Forest model for **winner_vs_national_pv_winner** achieved accuracy of 48 is the most surprising result of the whole project for me. It would be very interesting to see how exactly did the model determine the national popular vote winner while all the economic features in the model were specific to states. Perhaps this could be achieved by applying LIME to non-competitive states.

XII. Suggestions for future work

As a result of this project, these ideas for further exploration are proposed.

1) **Using K-Folds for cross-validation**. Cross-validation for Gradient Boosting in the project was suboptimal. Perhaps using K-Folds for cross-validation instead of repeated stratified train-test splits would help to find hyperparameters for models to generalize better.
2) **Using data to predict earlier years for cross validation**. Perhaps, it would be possible to find better hyperparameters while trying to validate on previous elections already contained in the train set by using earlier data (2016 on 1980-2012 data, 2012 on 1980-2008 data etc.) as opposed to using conventional cross validation methods.
3) **Use cloud computing services.** In this project, for cross validation of Stacking was not performed, and cross validation of Boosting was performed over a relatively small scope due to computational limitations. Perhaps, with the use of cloud computing services proper cross validation for Boosting and Stacking could be done.
4) **Predict a different target instead of swing.** Since **swing** naturally has such a big difference between its class distributions in train and test sets, it would be better to predict a feature describing whether the state has swung away from the party of the incumbent president instead.
5) **Try regression instead of classification.** While the winner of the election matters more than the percentage margin, it would be interesting to try to predict some numerical target as well.
6) **Use deep learning.** Since the goal of the project was to use the methods taught in the particular online course, deep learning methods were not used. Maybe the prediction results can be improved by using more complicated methods such as deep learning.
7) **Use dimensionality reduction.** The dataset used in the project has 333 features, with many of them correlated. Perhaps after applying dimensionality reduction, simpler model types that were not successful for this dataset could be used.

Links:

- Project source and full code: https://github.com/vectorkoz/economic-presidential
- Online course: https://www.coursera.org/learn/supervised-machine-learning-classification
- Political data: https://doi.org/10.7910/DVN/42MVDX
- Economic data: https://apps.bea.gov/regional/histdata/releases/0921spi/index.cfm
- Stacking article: https://towardsdatascience.com/how-to-properly-validate-a-model-when-stacking-ad2ee1b2b9c
- Code for choropleth adapted from: https://github.com/ilmonteux/mapping/blob/master/US_elections/election_maps.ipynb