Predicting the US 2020 Presidential Elections

I. Problem

The principal objective of this project is to predict the outcome of the 2020 presidential election. Indeed, On November 3, U.S. voters will decide who will be their next president for the next four years. The two Republican and Democrat candidates have been known for several months: Republican Donald Trump and Joe Biden, Democrat.

The problem is that there are many different variables to consider in such a prediction model: there are in fact 50 separate elections, one for each state, as well as the popular vote. In many cases, even though the polls are held by professionals, they may be not reliable because they are biased. We need to be very careful of that. Moreover, the Covid-19 crisis that began 8 months before the elections and the way the current President is managing this crisis may affect the presidential elections.

II. Approach

A. National popular vote model

With that in mind, we decided to first predict the popular vote taking into account macro data (how the country is performing economically, employment rate, etc.) and human data (professional polls). We are training the model on the 19 previous elections (since 1948) to predict the percentage obtained by each candidate for the national popular vote. This prediction for the national popular vote will allow us to include this national popular vote as a feature in order to predict the electoral college vote. One of the biggest issues of our model is that we think that we way people vote has drastically changed since WW2, because of the deep changes rooted in the American society, and the new way people interact with means of information. Therefore, we will only have 20 elections in order to train our model.

B. Electoral college model

The second step is to predict the Electoral College vote: we need to define the leaning of every state of the United States. One difficulty of such an approach is that some states have a 'loyalty' towards a party (Texas will always be Republican..). Therefore, in order to take into account this characteristic of every state, we thought about two strategies:

- Define a 'loyalty' feature towards Rep or Dem for every state. Then we would be able to define a single model for all of the states and predict on new states only based on the features we have defined for very state
- Define a model per state and train every model on the previous elections for this particular state. This would repeat the fact that we would only have 20 data points (from the previous last 20 elections) for every state. One solution to this issue is that states are somehow correlated on their predictions. Therefore, I can triain a state on all of the data from all of the states from the previous states, but weighting differently the observations from every observation based on the mutual correlation of the states. This allows to have 1000 training points for every per-state odel.

Then, after having tried these two models, we would select either the best one, or combine their strengths in an Ensemble Meta Model (*Mixture of Experts* model).

III. Data collection

A. How did we collect the data?

In order to predict the National Popular Vote, we need to include some macro-indicators that would reveal the state of the US during years before the election. It has been studied that macoeconomic indicators could influence widely the outcome of a presidential election. In our work, we decided to focus on:

- Current president's popularity: the *Net Approval Rating* (NAP)
- Economic indicators: GDP, Real Disposable Income for citizens (RDI), Payrolls of citizens
- Stock Indicators: Evolution of the Dowjones (Stock)

The sources for this data were: **GallUp Polls** and **Federal Reserve Economic Data**. We extracted aggregated (mean or median) values for every indicator according to different frequencies (for instance, the NAP was extracted every month 6 months before the election, whereas the GDP was extracted 6 times, 1 per year without election and 3 per election year). The created features were highly correlated (as we will see later). Last, another macro indicator of the state of the election was whether the current president was seeking for reelection, or if the previous president had accomplished his second mandate.

Special Election 2020 Note: Since all the economic indicators have been shaken for the last 6 months because of the covid19 crisis, we decided to 'lower' the impact of the crisis in order for the model to reflect the real behavior of the citizens. This would allow to treat the 2020 election point as in-distribution and not Out of Distribution.

This allowed to construct **26** features per election year.

We extracted the polls data from the Five-Thirty-Eight github repository for elections from 1968 untill 2020 (https://github.com/fivethirtyeight/data/tree/master/polls). Five-thirty-eight gathered many different polls where each of them have a sample size, the type of population interviewed, and the percentage of people that answered yes to one question "Would you be likely to vote for candidate A" where A is either the Republican or Democrat candidate. They then adjust those percentages on two criteria:

- trend line adjustment: in states that haven't been polled recently, we make inferences about what's going on there using national polls or polls from other states that have been surveyed recently. Indeed, trend-line-adjusted averages have been quite a bit more accurate, historically
- Adjustment for house effect: when certain pollsters consistently show better results for certain candidates.

Then, Five-thirty-eight averages those adjusted polls on a daily basis.

For elections from 1948 until 1968, the polls data was more complicated to find. We extracted the monthly averaged polls from Gallup which was the first polling organization to conduct accurate opinion polling for United States presidential elections.
https://web.archive.org/web/20170630070844/http://www.gallup.com/poll/110548/gallup-presidential-election-trialheat-trends-19362004.aspx#4

B. Feature engineering

Once the data extracted, we would need to process the polls data. Some studies (*An Updated Dynamic Bayesian Forecasting Model for the US Presidential Election*) revealed that the only polls that were truly relevant for presidential elections are the ones that are issued for the last 6 months of a presidential campaign: those are the ones we selected. We subsequently aggregated these polls in order to get a *per-month* poll indicator for both Republican and Democrat candidates.

The final dataframe consisted of **36** observations for **35** features.

Data Cleaning + Feature Engineering: In order to clean our data, we first imputed the missing values by interpolation. We then checked the types of the columns and the consistency of values.

A crucial issue we needed to deal with, as mentionned earlier, was that we only had a few observations for a lot of features. Therefore, we tried to engineer this features. We created two new dataframes:

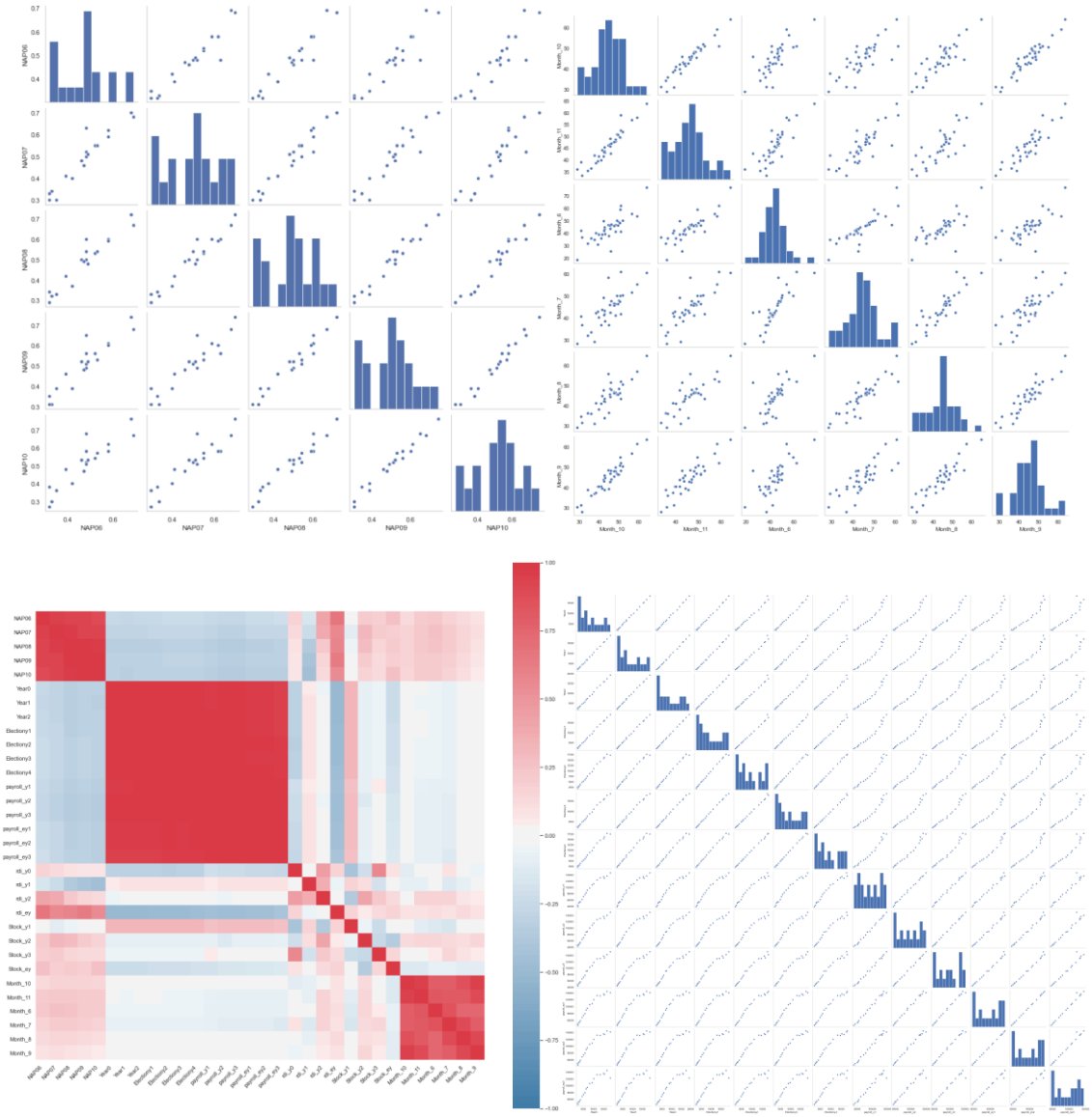
- The first one would consist in computing, for every section of our features (GDP, polls, NAP ..) a weighted sum of increases rate: $NAP = \sum_{\{i=1\}}^{5} w_i * \frac{NAP_{\{i+1\}} - NAP_{\{i\}}}{NAP_{\{i\}}}$ where the weights would be more important as we are getting closer to the election. The resulting dataframe would consist of 36 obesrvations and **9** features. We call it the difference dataset
- The second one would use the features from the difference dataset and combine them with 'weighted average' features. In the same fashion as above, we would aggregate the features from a same indicator in a weighted sum, with weights getting more important as we are getting closer to election year. The resulting dataframe would consist of 36 obesrvations and **15** features.This is the sum/difference dataset.

C. EDA on the final dataset

Once the Data Cleaning and Feature Engineering phases were completed, we went on to perform preliminary EDA on the initial dataset, the difference dataset and the sum/difference dataset. In the interest of brevity, this section will only focus on the preliminary EDA performed on the initial dataset (more on why we chose this dataset in the sections below).

We began the preliminary EDA with the .describe() statistics to explore any possible outliers we may have missed during the Data Cleaning process and to identify any possible outliers in our dataset. The extreme values from the .describe() statistics seemed well behaved and we did not discover any outliers, so we proceeded to pairs-plot and correlation matrix in order to identify possible correlations and dependencies between features.

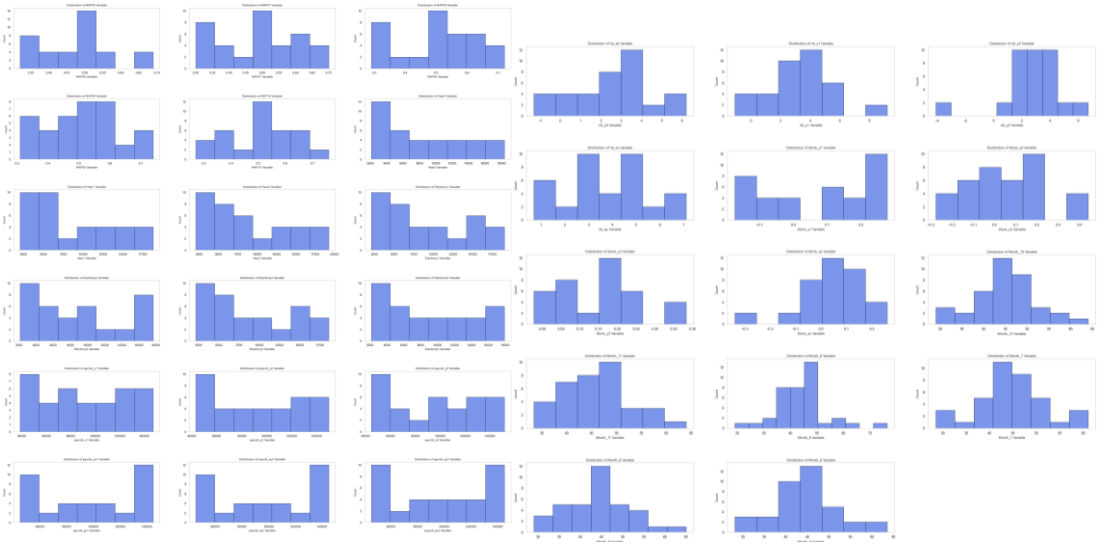
The pairs-plot excludes some categorical features, such as 'Democrat_Republican', as it seemed redundant to seek correlation between the candidate's political orientation and the numerical features in the initial dataset. The pairs-plot is a 'messy' 35 x 35 grid of plots. Consequently, in addition we will only show pieces of the pairs-plots for the correlated features, identified from the correlation matrix (the full pairs-plot is included in the Appendix).



From the correlation matrix and the pairs-plots above, we observe that we can combine the highly correlated features in our dataset into three categories:

- NAP[6-10] - net approval rating of the current president, election year, months June through October
- Year[0-2]; Electiony[1-4]; payroll_y[1-3]; payroll_ey[1-3] - measures of GDP and the payroll during off years and election years.
- Month[6-11] - national poll results, election year, months June through November.

The correlations observed above are very logical, given the nature of the problem at hand and the data collected. Nevertheless, as we will see in the modeling section below the strong correlation between the features could severely deteriorate the performance of some models (ex. Linear Regression). On the other hand, when performing PCA the correlation between the features could indicate strong signals in the data. As a result, we could expect models, such as Boosting or Lasso Regression reinforced with PCA to perform well on this dataset.



Next we went on to explore the distribution of the variables in our dataset. The intent of this section was to further explore our data and to perform a refined search for possible outliers. Note, we do not plot the distribution of the categorical features, as their distribution is known, and no interesting ideas can be drawn from studying it.

The distributions above seem pretty reasonable, with a good spread of values. One possible outliers are the '-4' observations for the 'rdi_y2' variable. Thus, we decided to explore this observation more closely. The observation is from 1948 and the value is substantially smaller that the values for 'rdi_y0' and 'rdi_y1'. We cannot guarantee the accuracy of the data collection process, especially for more outdated observations. Consequently, we conclude that the 'rdi_y2' observation for 1948 is a possible outlier.

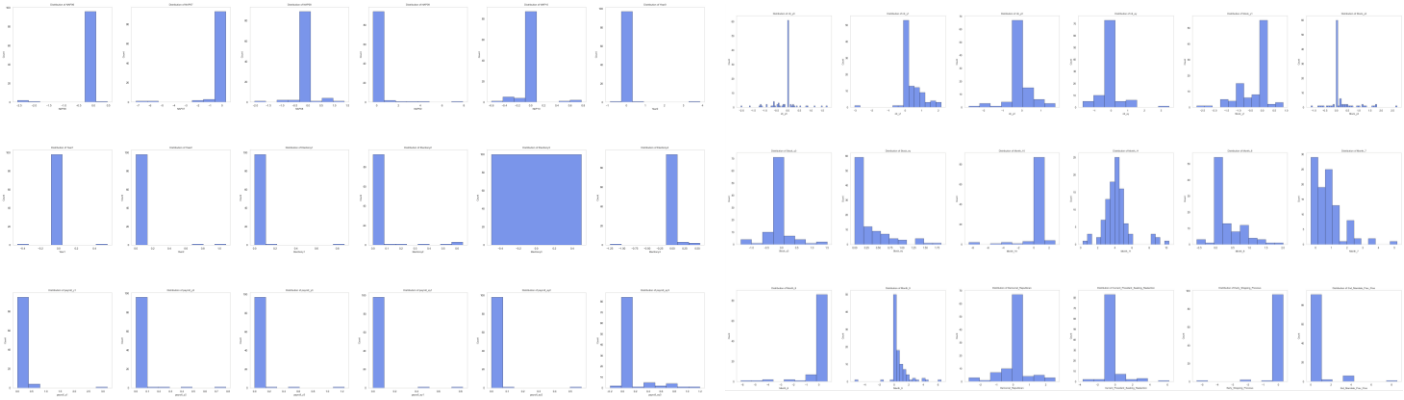
D. Which dataset should we use?

Having performed the initial EDA, we went on to explore how adequate the features in the difference dataset, the difference/sum dataset and the initial dataset are to predict the 'popular_vote_percentage'. To do so, we performed Bootstrapped LassoCV. The idea behind this approach was to assess how many times would a feature retain a non-zero coefficient (making it therefore more adequate to predict the 'popular_vote_percentage'). We then utilized the results obtained to determine the most adequate dataset, which we later used to model the 'popular_vote_percentage'. We present the results from our analysis in addition:

- In the difference dataset, we observed that in the overwhelming majority of the bootstrapped samples the values of all the coefficients were zeroed (see Appendix for the coefficient values distribution). Therefore, we concluded that the intercept, the mean prediction is a better predictor for modeling the 'popular_vote_percentage' than any of the features in the difference dataset. As a result, although we performed the baseline modeling on this dataset too, we decided not to include the difference dataset as our final dataset for predicting the popular vote.
- In the difference/sum data set, we observed that in the majority of bootstrapped samples the values of the coefficients were zeroed (see Appendix for the coefficient values distribution). Although we observe that some features, such as 'rdi', 'national_polls' and 'national_polls_dif' are somewhat adequate for modeling the 'popular vote_percentage', after performing the

baseline modeling on the difference/sum dataset to further reinforce out beliefs, we decided not to include this dataset as our final dataset for predicting the popular vote.

- In the initial dataset, we observe that in the majority of bootstrapped samples the values of the coefficients were not zeroed. The algorithm only zeroed part of the highly correlated features (the part it picked-up first), improving therefore the adequacy of the dataset to model the 'popular_vote_percentage' by retaining only the most relevant features. As a result after we performed the baseline modeling on the initial dataset, we realized that it is the most adequate dataset for predicting the popular vote. From this point onwards, the modeling part of the popular vote will be based on the initial dataset.



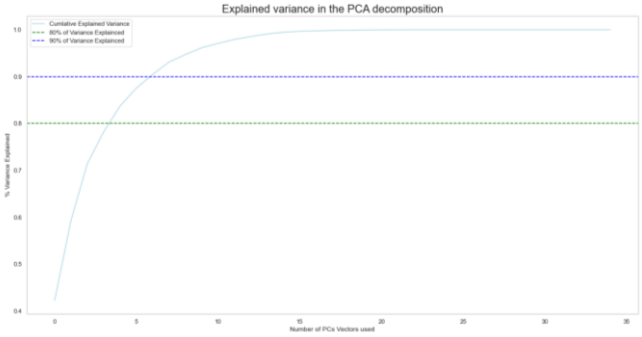
IV. Modeling the national popular vote

A. Data preprocessing: scaling the data and perform PCA

The first step of our modelling part was to scale our data. Indeed, the various different scales present in our dataframe would make a very difficult interpretation and scoring for our different predictions. We chose to standardize our data because the different continous features might have a shifted Gaussian Distribution (letting go of negative values).

On top of this standardized data, we computed the PCA decomposition of our dataframe. There were many reasons for PCA to work well with our dataframe:

- This would handle the high correlation among our predictors
- It would enable us to work with less features and have a better training of our model



Therefore, according to the model we use, we expect our models to perform well when they use between **3** and **10** Principal Component Vectors.

Therefore, we had two dataframes to play with: the scaled features and the PCA vectors.

B. Models explored

The models we have used are: Linear Regression, Lasso Regression, Regression Trees, Random Forest Regression and AdaBoost Regression. The scoring metrics we used is the Negative Mean Square Error. We performed Hyper-Parameter tuning + Model selection via Cross-Validation. The cross-validation strategy was 9-fold Cross Validation. Please see the appendix for an exhaustive list of the Hyperparameters we tuned for every model, and their range of values.

A priori, we expect the different models to perform very differently.

- We expect Linear Regression on scaled data to behave poorly because of the correlated features: the Gram matrix will not be invertible.
- We expect Tree models to perform poorly because they are to be fit on the whole features and we only have a few training points: high variance of the tree predictions
- We expect Random Forest to have a considerable increase in performances when compared with trees. There are several reasons for that. First, they grow trees with a small amount of features: less variance per tree, and then they average the predictions over trees: less resulting variance
- We expect penalized models to perform well: Lasso on scaled features and PCA vectors should perform well.

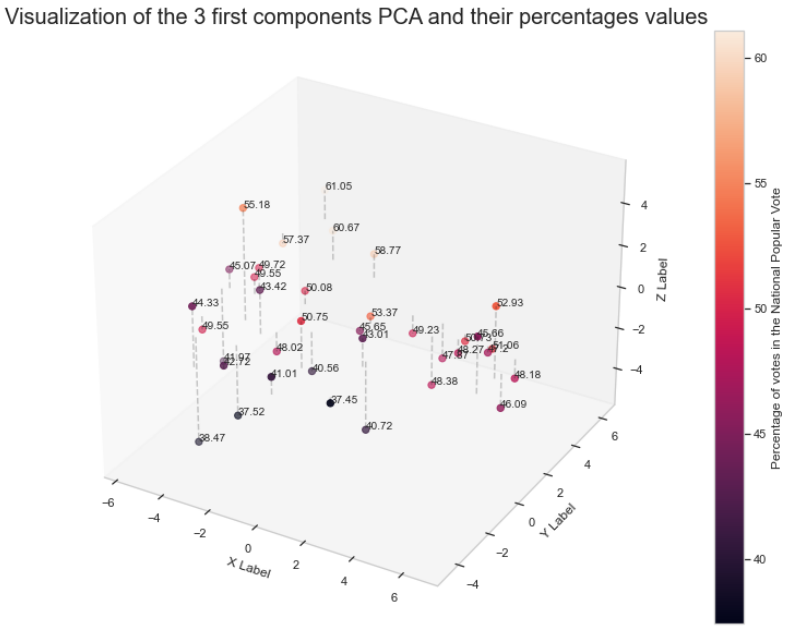
In order to see the complete results of our models + Hyperparameter tuning, please refer to the Appendix.

C. Final model chosen

The model that performed best according to cross-validation was:

- Lasso Regression on 3 PCA vectors with a regularization coefficient C = 0.1. The CV MSE was 7.535. The resulting MLE weights parameters of the Model are [-0.118, 1.857, 1.377]. The CV R2 score is 0.7268.

Let us visualize, for the sake of interpretation, the distribution of the repartition of the votes against the first 3 PC vectors.



As we can see, we have successfully captured a pattern in predicting the national popular votes (increasing from bottom right to far left). This is what is captured by the weights of this model: negative in the first component (X) and positive in the two others.

Therefore, this is the model we will be using in order to make the predictions for the National Popular Vote.

We computed the final prediction for the National Popular Vote and computed CI via Bootstrapping for this method. The final results are:

- For Donald Trump: NPV = 46.525 +/- 1.39
- For Jobe Biden: NPV = 52.46 +/- 1.88

Note: The final actual result (47.3 for D. Trump, 51 for J. Biden falls right into our Confidence Interval).

V. Conclusion

Therefore, we might say that the model we built for Predicting the National Popular Vote is robust:

- The features selected are consistent (they perform better than the 'average baseline' model, coming from the other engineered dataframes)
- We dealt with Correlation by performing PCA and training models that we robust to Correlation (Lasso on PCA)
- We got Confidence Intervals that made a lot of sense regarding the actual results

But overall, we successfully being biased towards Joe Biden because of the polls and the coronavirus crisis, which is not the case for all of the other predictions models.

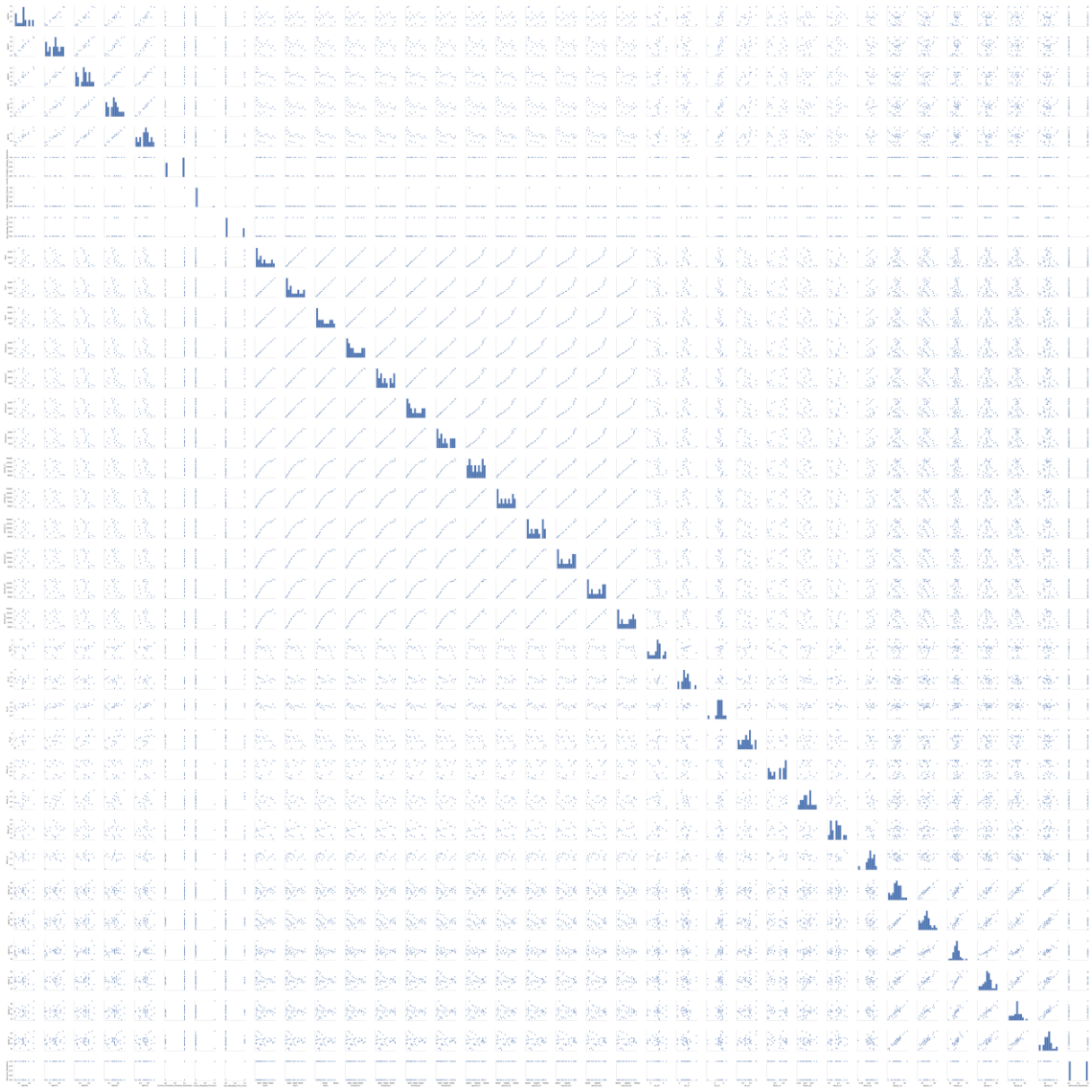
Forecast	Polls

Pollster	Date	Population	Mode	Sample size	Biden	Trump	Margin
Ipsos	OCT 31ST-NOV 2ND	Likely to vote	Online	914	54%	46%	D +7.2
YouGov (Economist)	OCT 31ST-NOV 2ND	Likely to vote	Online	1,363	55%	45%	D +10.4
Research Co.	OCT 31ST-NOV 2ND	Likely to vote	Online	1,025	54%	46%	D +8.7
AYTM	OCT 30TH-OCT 31ST	Likely to vote	Online	700	55%	45%	D +10.3
Redfield & Wilton Strategies	OCT 30TH-NOV 1ST	Likely to vote	Online	8,765	56%	44%	D +12.8
YouGov (Yahoo)	OCT 30TH-NOV 1ST	Likely to vote	Online	1,360	55%	45%	D +10.4
YouGov (Yahoo)	OCT 30TH-NOV 1ST	Registered voters	Online	1,501	55%	45%	D +10.4
Change Research	OCT 29TH-NOV 1ST	Likely to vote	Online	1,880	55%	45%	D +10.6
SurveyUSA	OCT 29TH-OCT 31ST	Likely to vote	Online	1,265	54%	46%	D +8.3
RMG Research	OCT 29TH-OCT 31ST	Likely to vote	Online	1,200	54%	46%	D +7.4

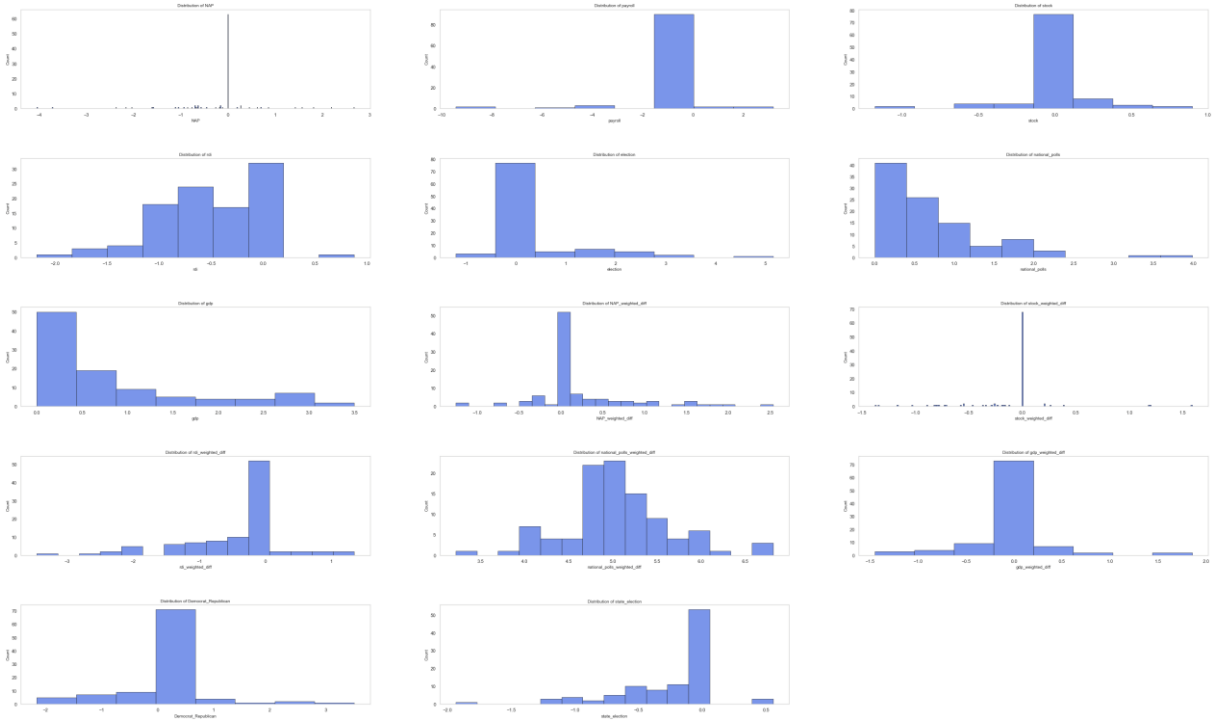
Now, the next step of our work is to leverage these predictions of the national popular vote and to construct new features for our per-state approach in order to build a model that would be able to predict the outcome of the Electoral College.

VI. Appendix

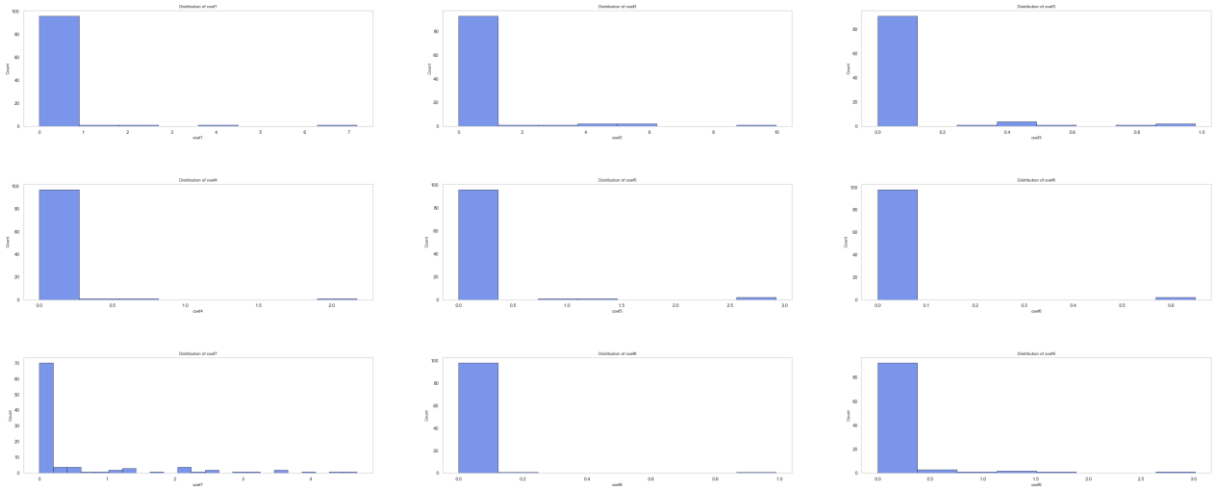
Full pair plots



Bootstrapped LassoCV with the 9-features dataframe



Bootstrapped LassoCV with the 15-features dataframe



Hyperparameters tuned for every model

- Lasso CV : regularization parameter : [1e-7,1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,0,1,10,100,1000]
- Tree: max_depth: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- Random Forest:

N_estimators: [50, 90, 110, 150, 200]	Min_samples_split: [1, 2, 3]
Max_features: ['auto', 'sqrt']	Min_samples_leaf: [1, 4, 5]
Max_depth: [1, 2, 3, 4, 5, 6, 7, 8]	Bootstrap: [True, False]

- Boosting:

-	Depth_Tree: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
-	N_estimators: [10, 20, 30, 40, 50, …., 200]
-	Learning_rate : [1e-3,1e-2,1e-1,1,10]

-