Machine Learning Engineer Nanodegree

Capstone Project

Dimitri Theoharatos
June 27, 2018

Classification of Political Party Affiliation

Project Overview

Understanding the political values of humans and the underlying characteristics that shape these values is an important question in understanding humanity. In America, people tend to be grouped into two factions, Republicans and Democrats. Though there are gray areas along the political spectrum, these two parties have maintained their supremacy in American politics for over 150 years. The common wisdom to determine party affiliation is that an individual possesses a certain set of values and thus prescribes to the party in which she believes her values are best represented. This begs the question, what are the fundamental characteristics that shape these values?

Using data compiled by the American National Election Studies (ANES) organization [1], I attempt to answer this question. ANES has surveyed voters every four years since its inception in 1948, collecting a long list of characteristics defining each respondent. These surveys include demographic-related questions, values-based questions, and party identification-related questions.

Problem Statement

Equipped with decades of compiled data from ANES, I will build a supervised binary classification to determine the party preference of an individual. To do so, I will extract features I deem important through the exhaustive list of nearly 1,300 features and use these features as inputs into my classified algorithmic pipeline. My output, party preference, is also a column in the ANES dataset, and I will use this as my target variable.

Given the dynamic nature of political parties and their ideologies, I have decided to break this up into two problems to enhance the predictive capabilities of my algorithm. For data exploration purposes, I will compile two sets of data, one from modern times that includes the 2004, 2008, and 2012 surveys, and the other a historical set that includes the data from 1952-1960. It might seem more reasonable to use all the data to train my model to make it more robust, but consider the following geographical election maps where the blue and red coloring represents the Democrat and Republican presidential candidate winning that state, respectively:

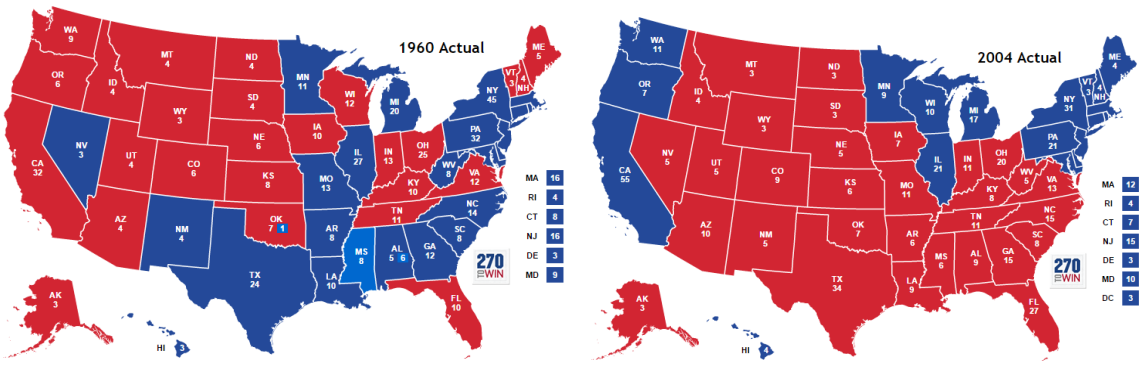

Figure 1: Election maps for the 1960 (left) and 2004 (right) elections [2]. Notice the major geographical differences between the two elections.

The south completely shifts (see [3] for a possible reason why) from Democrat to Republican and so does the west coast but from Republican to Democrat. Considering geographical location represents several inputs of the feature set, it seemed sensible to split the problem into separate epochs. Thus, after reducing the feature space and forming two separate datasets, I will predict which party an individual prescribes.

Evaluation Metrics

I will use accuracy and the F1 Score metrics to evaluate my algorithms, defined below:

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})$$
$$\text{F}_1\ \text{Score} = 2 \cdot (\text{Precision} + \text{Recall})/(\text{Precision} * \text{Recall})$$

where:

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$
$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

and TP, TN, FP, and FN represent true positive, true negative, false positive, and false negative respectively. I chose to include the F_1 Score as a metric since there is a slight class imbalance in my data (60-40 split among Democrats and Republicans) and this imbalance makes accuracy not as effective of a metric.

Analysis

Data Exploration

The ANES dataset has nearly 1,300 features and I manually sifted through the documentation of each variable to decide on what to include in my initial data exploration phase (this was as tedious as it sounds). I wanted the focus of the feature set to be demographic in nature, rather than implementing some self-annotating feature that would obviously produce the right output (e.g. the question "Who did you vote for?" would almost certainly indicate the party preference of the individual). With this attempt to avoid circular logic and hone in on the demographic qualities of the individual, I narrowed down the feature set to the following features:

age_range: the age range in which the individual falls under broken into seven categories.
gender: the gender of the individual.
race_3: the race of the individual broken down to 'white', 'black', or 'other'.
race_7: the race of the individual with additional categories including 'Asian' and 'Hispanic'.
education_level: the education level of the individual broken into four categories.
geo_region: the geographical region in which the person resides.
south: a binary feature that determines whether the individual resides from the south.
hh_income_percentile: the household income of the individual categorized into five bins.
occupation: the occupation of the individual among five categories.
occupation_several: the occupation of the individual with several more categories.
union_membership: whether a member of the household is a part of a union.
religion: the religion of the individual using four categories.
parents_native_born: whether the parents of the individual were born in the United States.
marital_status: if the individual is married, divorced, or has never been married.

This makes a total of fourteen features. Notice that every single one of the features is categorical. Though a few are ordinal in nature (e.g. income, education, age), the rest cannot be quantified. The target variable is defined below:

party_preference the preferred party of the individual. This includes people that define themselves as a strong partisan, a weak partisan, or an independent that leans a certain way.

For the modern dataset (2004-2012), there are a total of 7,506 samples. For the historical dataset (1952-1960), there are a total of 5,716 samples. The same features and target variable are used in both datasets. To get a sense of the party distribution, refer to Table 1.

Party Preference	Count (1952-1960)	Count (2004-2012)
Strong Democrat	1451	2094
Weak Democrat	1535	1311
Leans Democrat	479	1220
Strong Republican	872	1086
Weak Republican	938	922
Leans Republican	441	873

Table 1: Distribution of party preference for both historical and modern datasets.

Exploratory Visualization

Let's do an initial investigation into our data to see which features of interest may be important in predicting the output. We'll start with race.

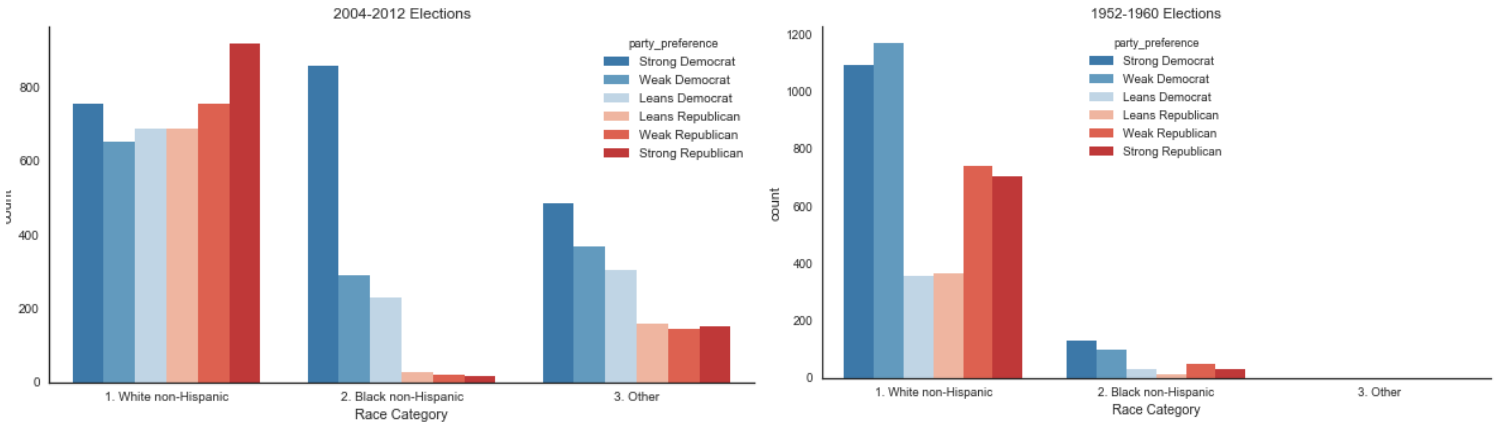

Figure 2: The impact of race on the party preference of an individual for both the modern (2004-2012) and historical (1952-1960) datasets.

A couple of things to note about race: black people overwhelmingly are Democrats in the modern elections, but are barely represented in the electorate in the historical elections. This suggests that race will be important in the modern elections but not as significant for the historical dataset. Let's visualize the impact of residing in the south.

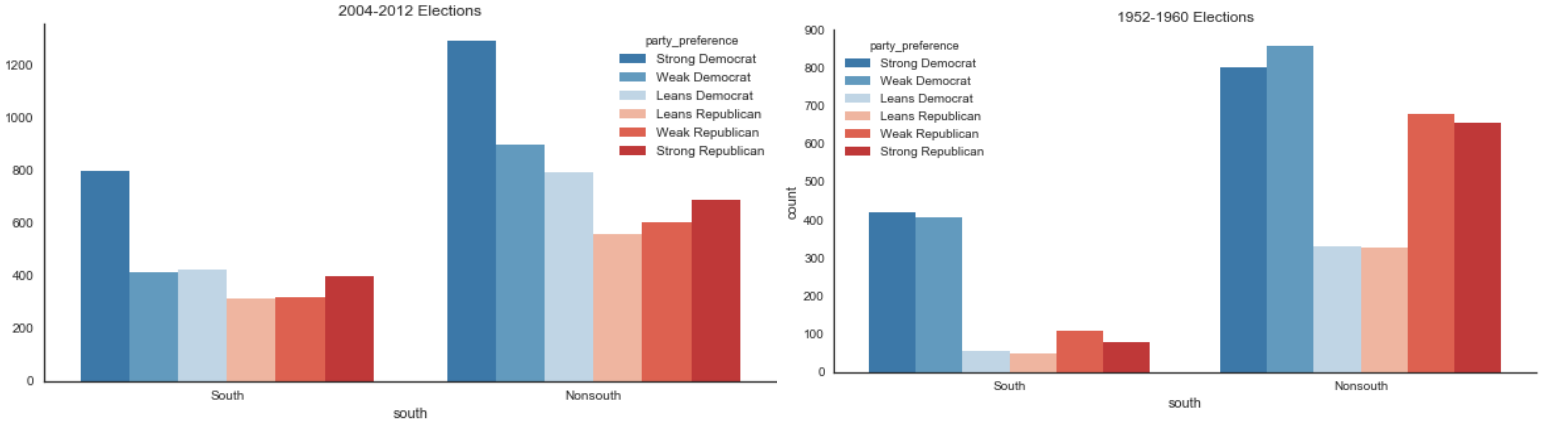

Figure 3: The impact of south residency on the party preference of an individual for both the modern (2004-2012) and historical (1952-1960) datasets.

To my surprise, there were more Democrats than Republicans in the south in both time frames, but the extent in which the Democrats outnumber the Republicans is vastly different. Residing in the south is a clear indicator that the individual is Democrat in the 1952-1960 dataset. Let's visualize the impact of income on the distribution of party affiliation.

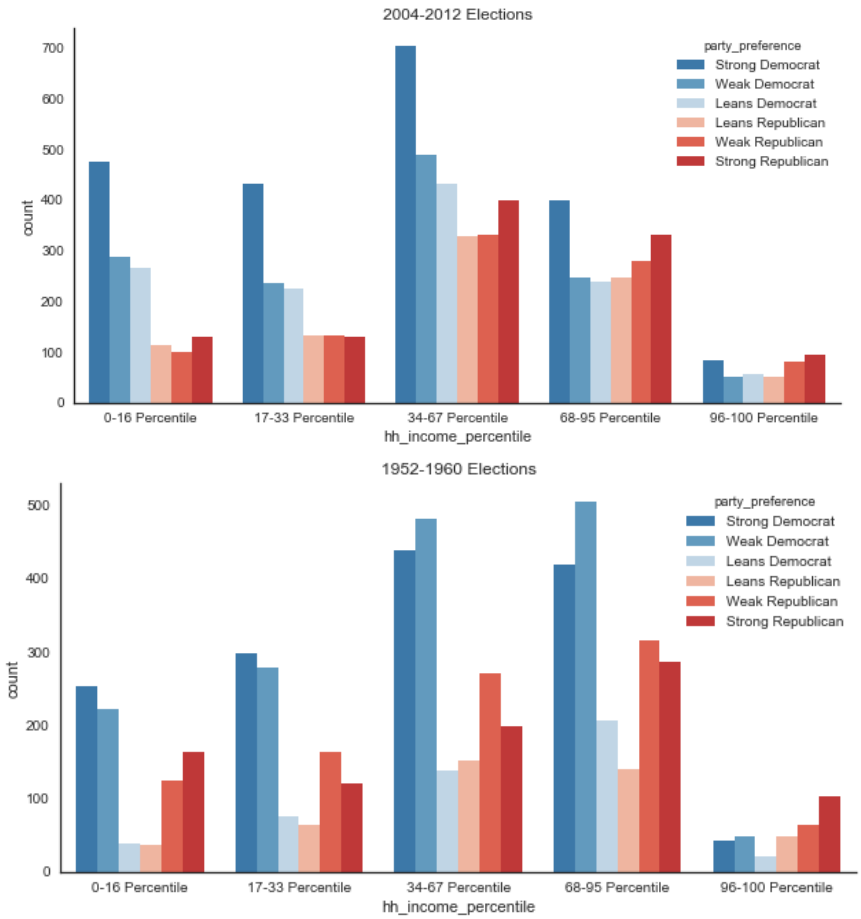

Figure 4: The impact of income on the party preference of an individual for both the modern (2004-2012) and historical (1952-1960) datasets.

The trends are relatively similar here, with the largest differentiation among the richest in the historical dataset and the lowest income in the modern dataset. There were a lot more Republicans in the lower income bracket in the historical data than there were in the modern data. Gender may have some interesting revelations too, let's investigate.

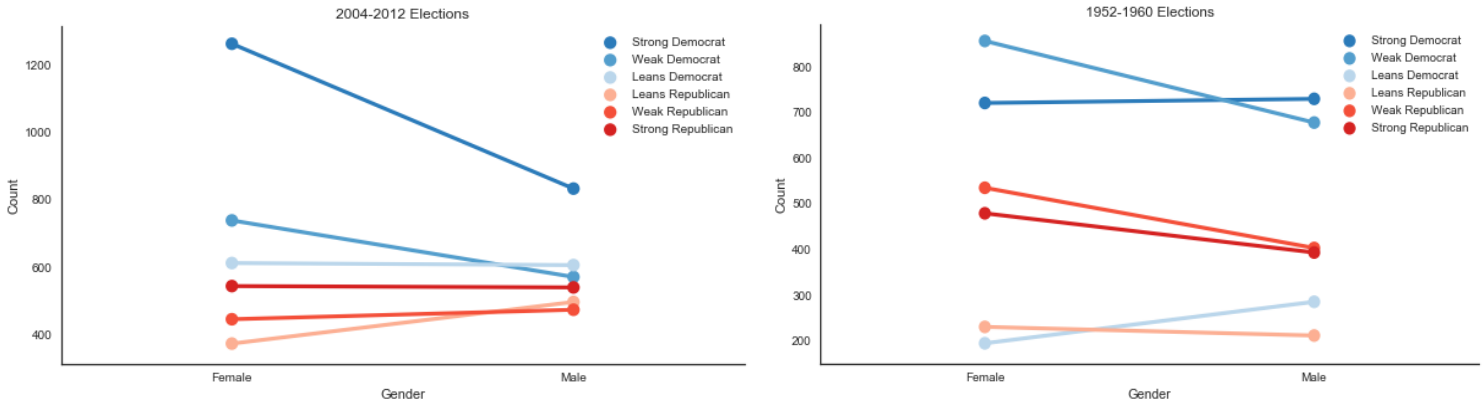

Figure 5: Point plot depicting the impact of gender on the party preference of an individual for both the modern (2004-2012) and historical (1952-1960) datasets.

In the modern elections, it looks like there are proportionally more men that are Republican than Democrat and vice versa for women. There is no strong correlation either way in the older elections.

After investigating some features of interest based on intuition, the entire feature space was evaluated for their relative important by integrating a random forest classifier that has a built-in feature importance ranking mechanism.

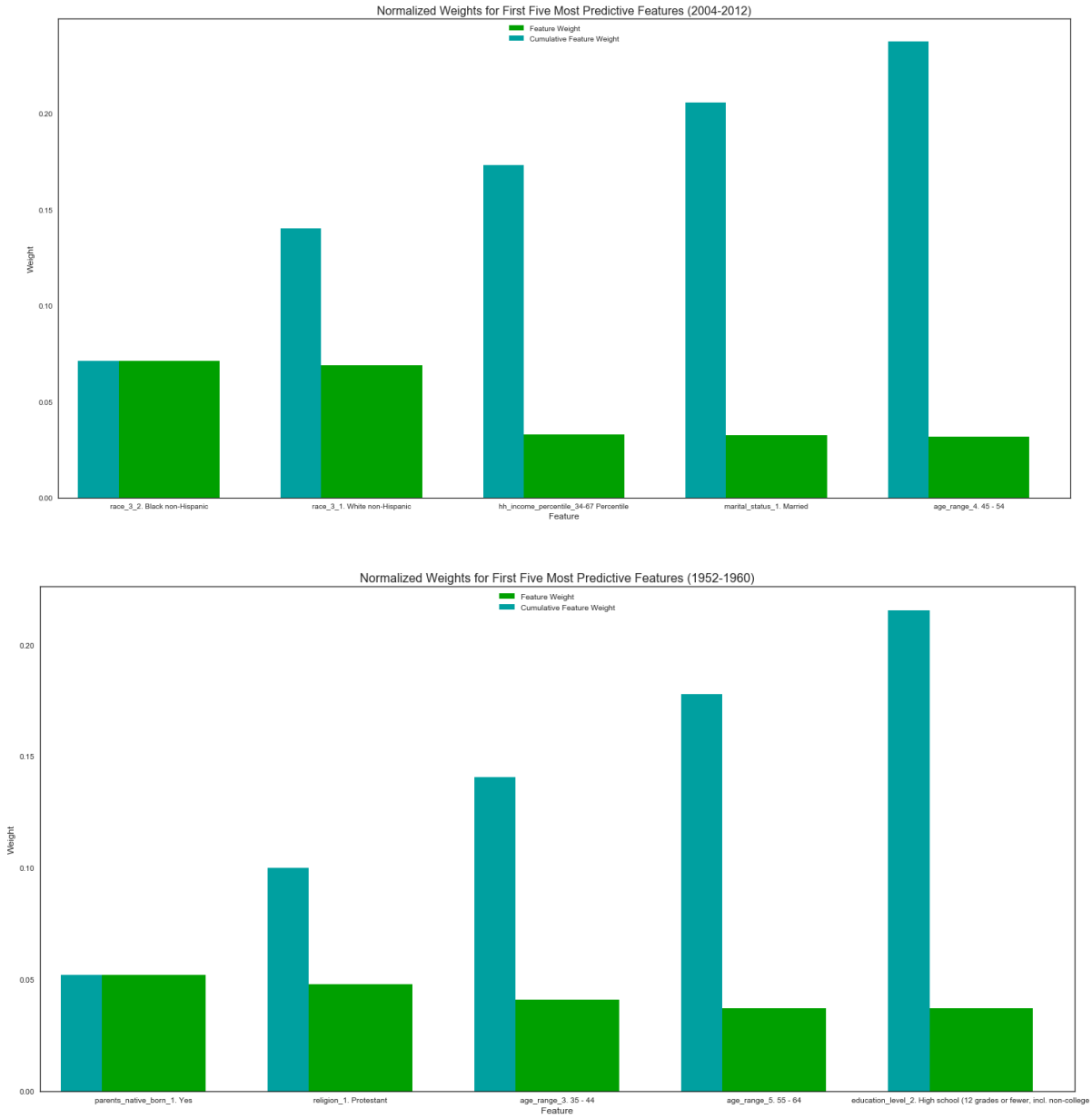

Figure 6: The five most important features for both the modern (2004-2012) and historical (1952-1960) datasets using a random forest classifier.

The features of importance have no overlap between the two datasets implying that the factors that best predict party affiliation change over different epochs. Furthermore, notice that nearly all of the top features hover around a weight of 0.05 suggesting that each feature in the feature space has similar predictive power.

Algorithms and Techniques

The problem posed is a supervised classification problem, allowing for several algorithms to be compared. The following algorithms will be used: logistic regression, decision trees, random forest, AdaBoost, and support vector machines (SVMs). Logistic regression and decision trees are both great candidates for initial profiling because of their interpretability and ease of use. Due to the tendency of decision trees to overfit the training data, ensemble methods such as Random Forest or an AdaBoost model will be used to alleviate the overfitting and hopefully increase the model's performance. Finally, SVMs are also used due to their effectiveness in high dimensional spaces. This suite of models represents a varied class of supervised models ranging from easy implementations with high interpretability to more complex models with low interpretability. Such a diverse set of algorithms will ensure an optimal model is realized. Let's further investigate each algorithm to better understand how the data is being treated.

In logistic regression, predictions are conditional probabilities of a sample belonging to a specified class after a transformation function is applied. This function is referred to as the logistic function and generates an S-shaped output constrained within the values 0 and 1. By taking the logarithm of the odds ratio of this conditional probability, a parametric, linear predictive function can be constructed with weights applied to each feature. These probabilities are fed into a maximum likelihood function to determine the prediction of each sample belonging to a particular class.

Unlike logistic regression, decision trees are both not parameterized and nonlinear. Instead, decision trees classify based on the information gain at each decision. More concretely, at each step of the decision tree, the decision tree maximizes the information gain meaning that the children will have less entropy than the parent node. Entropy is a measure of disorder. Therefore, if a node has equal parts of two classes, the entropy will be at a maximum whereas if a node only has one class within it the entropy will be at a minimum. By ensuring the children at each decision node have lower entropy than their respective parent, the purer the leaf nodes will be resulting in more accurate predictions.

Since decision trees often overfit, it would be useful to use an ensemble method such as random forests to reduce the error due to overfitting. Instead of training the entire training dataset using every feature, the following steps can be performed instead:

1. *Randomly select a subsection of the feature space to train on a bootstrapped sample of the training data.*
2. *Train each bootstrapped training set on these subsections of the feature space to create fully-formed decision trees.*
3. *Determine the prediction by aggregating the decision of each decision tree, in this case using majority-vote since it's a classification problem.*

Similarly, another ensemble method, boosting, can be used that will also improve upon simply relying on one weak learner for classification. Unlike random forests, boosting utilizes the entire feature space for prediction, but instead of running several weak learners in parallel to fit various trees, it instead does so sequentially. It benefits from this sequential method since each successive algorithm fits the residual error of the previous algorithm, increasing the penalty for misclassifying these observations and thus making it less likely to continue to misclassify these respective samples.

The last algorithm that will be used in this project is a support vector machine (SVM). The goal of the SVM is to separate the training samples via a hyper-plane while attempting to minimize misclassifications and maximize the margins between the clusters of data. The margin is defined as the distance between the hyper-plane and the closest data point. Separating the data with a plane becomes very difficult especially when the data is nonlinear. To deal with this, SVMs employ a kernel function, which maps the data to higher-dimensional spaces, making it easier to separate the data and creating a separation boundary. This requires a lot of computational power and thus takes longer to train than any of the other previously-mentioned algorithms.

Benchmark

A naïve benchmark that can be used as performance for this test is as follows:

1. Determine the majority party (Democrats in both cases)
2. Always predict that party
3. Output the accuracy and F_1 Score

Implementing the above results in the following benchmark results:

Dataset	Accuracy	F_1 Score
Modern	**0.621**	**0.672**
Historical	**0.619**	**0.671**

Table 2: Benchmark results for accuracy and F_1 Score using a majority-class predictive model.

This majority-class naïve predictor will be used as a baseline to compare all other predictive models to. Ideally, the final, optimized model (as well as the intermediate models) will greatly improve on both the accuracy and F_1 score of this naïve model.

Methodology

Data Preprocessing

There was a clear need to preprocess the data as there were initially nearly 1,300 columns in the dataset, most of which would have no impact on the party preference. This was performed manually since I already had an idea as to which features I wanted to investigate. Also, if any of the values for each of these features were not available or missing, I removed these rows from my dataset.

In addition to the features being cleaned, the target variable was also transformed. The original target variable possessed seven categories ranging from "Strong Democrat" to "Strong Republican" and everything in between. I eliminated all entries that were pure Independents, and grouped strong partisans, weak partisans, and Independent-leaning partisans in either direction. Grouping Independents that leaned either way was a tough decision to make, as I knew this may lessen the performance of my model, but I determined it wiser to maintain a larger, more robust dataset. If any of the entries were missing or not available, I removed these individuals from the dataset.

In order to segment my datasets into separate time frames, I only included data from a certain range of time for each respective set. This was easy to do since there was a "Year" column in the dataset. Finally, for the sake of more aesthetic visualizations, I removed any unused categories in the dataset and renamed several features.

Implementation

After performing some initial profiling on the chosen algorithms using their default parameters, I selected the best-performing algorithms to optimize their parameters and squeeze out even more performance. The inputs into each of these models are the 14 features, with the output being the ***party_preference*** variable. In the initial profiling stages, I performed 5-fold cross-validation and found the mean and standard deviation for the accuracy and F_1 score. In addition, I plotted the ROC curves for each algorithm to provide a visual interpretation of the area under the curve.

Since all of the features were categorical in nature, I used a one-hot encoding scheme for each feature, drastically widening the feature space. In addition, the target variable was encoded as a binary output. After these steps were taken, the data could finally be fed into the models. The datasets were broken into training and testing sets using an 80/20 split. Then, using K-fold cross-validation with five folds, the data was fed into several supervised classification algorithms and plots were generated of their accuracies and F_1 scores. These are depicted below for the modern dataset (similar results were found for the historical dataset).

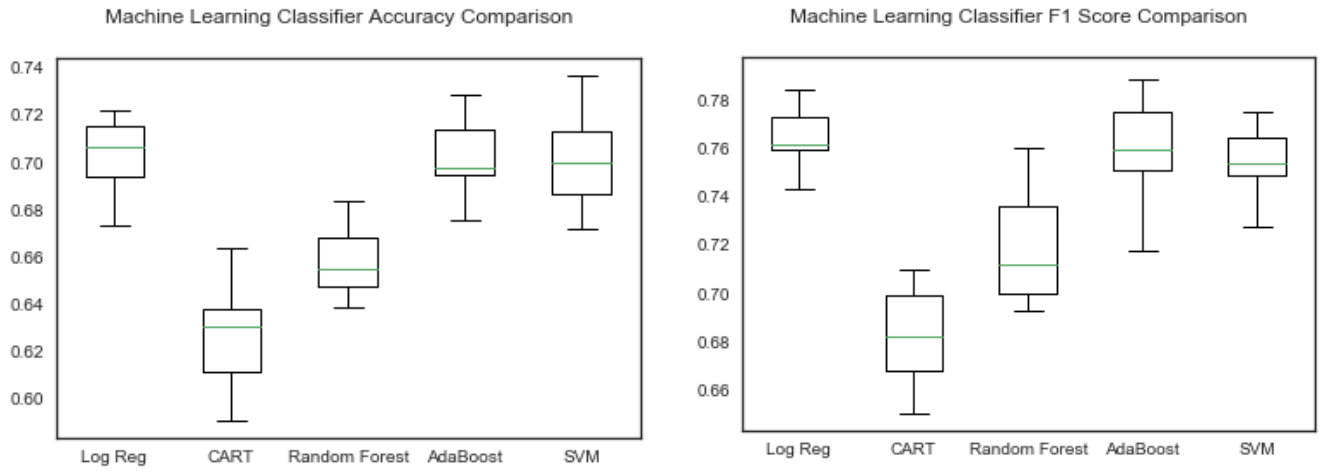

Figure 7: Boxplots using 5-fold cross validation comparing accuracy and F_1 scores for each algorithm in the modern dataset.

Based on the initial profiling, it was clear that the top models initially were logistic regression, AdaBoost, and SVMs. The parameters of these three were tuned to optimize each model during the validation process.

Since the dataset was already in a relatively clean format, there were no notable coding complications that arose. Apart from the tedious nature of cleaning the dataset and one-hot-encoding each categorical feature, the coding implementation was relatively straightforward.

Refinement

After running the data through a basic logistic regression model, I was not getting the accuracy I was expecting (67% accuracy with a 0.671 AUC) so I modified the dataset. Since the training set size was reduced with each additional feature added due to missing entries, I went through the feature list and removed any redundant features. The two features I got rid of were the redundant race (race_7) and occupation (occupation_several) features that had more missing values than their respective counterpart. This process alone ended up increasing the accuracy, F_1 Score, and area under the curve (AUC) using the same logistic regression model.

With a simple reduction of the feature space, the AUC score improved for the logistic regression algorithm by 0.03 and the accuracy by an additional 3%, demonstrating the importance of a large training set. Once the redundant features were removed, it was clear from all three metrics (accuracy, F_1 score, and AUC) that the top three models for both historical and modern datasets were logistic regression, AdaBoost, and SVM.

Results

Model Evaluation and Validation

Since several supervised machine learning models were implemented, it was crucial to compare performance between the models and to optimize the model parameters to enable a robust solution. Once the top-performing models were selected from the initial performance profiling (logistic regression, AdaBoost, and SVMs), GridSearchCV was integrated for hyperparameter tuning. The hyperparameters tested for each algorithm are depicted in the table below:

	C	Penalty	Number of Estimators	Gamma
Logistic Regression	logspace(0,4,10)	L1, L2	-	-
AdaBoost	-	-	[10,20,30,40,50, 60,70,80,90,100]	-
SVM	[0.1, 1, 10, 100]	-	-	[0.001, 0.01, 0.1, 1]

Table 3: Each top-performing algorithm and their respective hyperparameter values that were iterated through to optimize each model. Note that logspace(0,4,10) represents ten numbers evenly spaced on the logarithmic scale from 1 to 10000.

After running a GridSearch on using the above hyperparameter values, the following parameters were determined to be optimal for the modern and historical datasets depicted in Table 4 and Table 5.

	C	Penalty	Number of Estimators	Gamma
Logistic Regression	1.0	L1	-	-
AdaBoost	-	-	100	-
SVM	100	-	-	0.01

Table 4: The optimal hyperparameter values for each model using GridSearchCV for the modern dataset.

	C	Penalty	Number of Estimators	Gamma
Logistic Regression	2.783	L2	-	-
AdaBoost	-	-	40	-
SVM	1	-	-	0.1

Table 5: The optimal hyperparameter values for each model using GridSearchCV for the historical dataset.

Interestingly, the optimal parameters for logistic regression were its default parameters, meaning we have already reached its optimal performance using the out-of-the-box scikit-learn implementation.

In order to determine how robust the algorithms were, K-fold cross-validation was used with both five and ten splits. The accuracy and F1 scores were still within 5% of each other regardless of the number of folds, implying that the model is robust to the dataset regardless of the number of splits. Since the folds are randomized and two different-sized folds were integrated into the model evaluation, the model is likely to generalize well to unseen testing data. However, as seen in Figure 7, there is also clearly some variation of performance for each algorithm demonstrated by length of the boxplots. However, the variation is relatively small with all of the tests falling with 3 percent of each algorithm's median performance.

Justification

After optimizing each model's hyperparameters, the performance for each could be compared to the baseline benchmark of always predicting the dominant party. The results for both accuracy and F_1 score metrics are detailed in Table 6 and Table 7.

	Baseline	Logistic Regression	AdaBoost	SVM
Accuracy	0.621	0.721	0.718	0.73
F_1 Score	0.672	0.782	0.779	0.792

Table 6: Comparison of accuracy and F_1 Score metrics after optimization using the modern dataset.

Similar steps were taken for the historical elections (1952-1960) and the results are depicted in the table below:

	Baseline	Logistic Regression	AdaBoost	SVM
Accuracy	0.619	0.711	0.713	0.715
F_1 Score	0.671	0.756	0.756	0.753

Table 7: Comparison of accuracy and F_1 Score metrics after optimization using the historical dataset.

For both accuracy and F_1 score metrics, substantial improvements are made relative to the baseline measurements. The support vector machine ended up performing the best when using the modern dataset, though logistic regression and AdaBoost were close behind. Interestingly, the vanilla, out-of-the-box logistic regression model with no parameter optimizations essentially performed as well as its more sophisticated counterparts. All three models saw an improvement of performance ranging from 15-18% of its original performance for both metrics.

With regards to the historical dataset, none of the algorithms really stood out from one another, with all three models performing essentially the same. There is also improvement for each model relative to the baseline model, though not as significant as the corresponding improvements using the modern dataset.

There is clearly room for improvement in correctly predicting an individual's party classification, but with an F_1 score of 0.792 and accuracy of 0.73, the optimal SVM model is certainly a step in the right direction.

Conclusion

Free-Form Visualization

An important quality to any machine learning project is determining both the best metrics to implement as well as the best algorithms to use. Figure 8 encompasses both qualities.

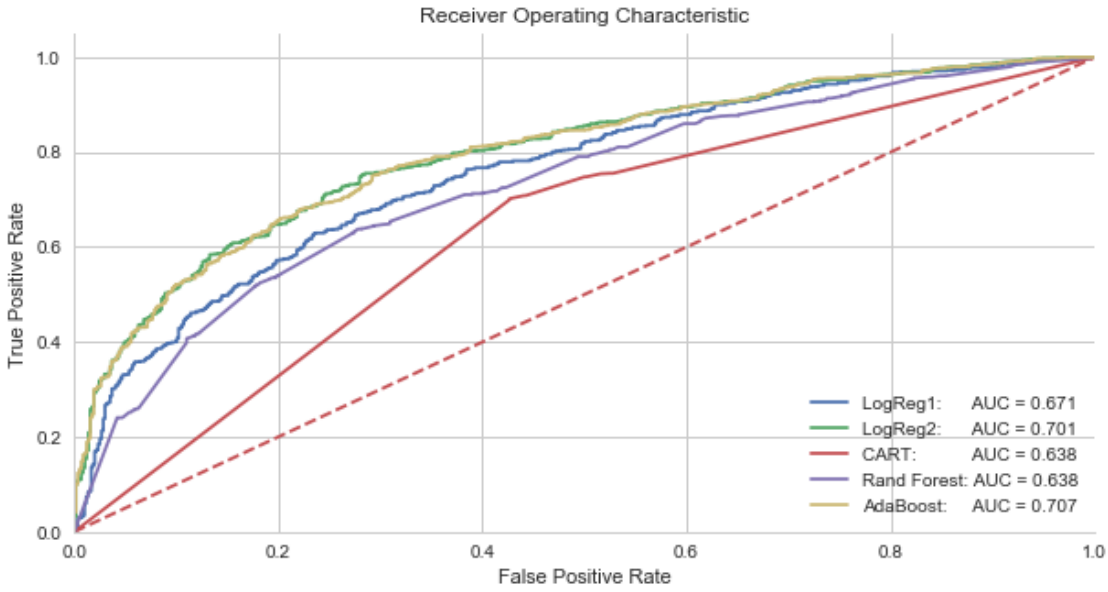

Figure 8: ROC for each algorithm using the modern dataset. Note that the LogReg1 and LogReg2 refer to logistic regression models performed before and after the feature space had redundant features removed, with the LogReg2 plot representing the updated feature space

Since the dataset had imbalanced classes, it was necessary to use alternative metrics to accuracy such as the AUC and F_1 score. Models that perform well tend to have higher areas under the ROC, suggesting that the updated logistic regression model (LogReg2) and AdaBoost were the top performing models (support vector machines do not have an ROC but were proven to be high-performing by comparing the F_1 scores). In order to build a robust machine learning pipeline, several parameters should be iterated through and optimized including metrics, algorithms, and hyperparameters for each algorithm. This iterative procedure underlies the practical, hands-on approach to machine learning. There is no black box, one-size-fits-all model to machine learning problems and this visualization is a testament to that concept. A basic algorithm such as logistic regression performs better than an ensemble model such as random forest, emphasizing the necessity of a thorough investigation of each model before coming to any conclusions based on preconceived notions of typical model performance.

Reflection

Determining political party affiliation based on the ANES dataset proved to be a rewarding and challenging experience. After sifting through massive amounts of data, the feature space was reduced from thousands to fourteen based on political intuition. Then, the dataset was further broken up into two distinct datasets- one that encompassed historical elections (1952-1960) and one that encompassed modern elections (2004-2012). Each dataset was explored to understand the important features in predicting political party affiliation and once the data was thoroughly understood and cleaned, it was fed into a machine learning pipeline. Using half a dozen supervised classification algorithms ranging from a basic logistic regression model to more sophisticated ensemble methods, the top three models were selected to further optimize the predictive capabilities. Finally, each top models' parameters were tuned to optimize performance and were compared to the baseline models.

An interesting aspect of this project was integrating personal political knowledge to determine the best features to use in the model. It stressed the importance of domain knowledge to create optimal machine learning solutions. One of the more difficult issues was cleaning up the massive dataset. Missing entries, redundant features, and determining how to best group variables was an extensive procedure, underlying the significance of data preprocessing in the overall machine learning process.

Improvements

One improvement derives from the fact that more of the ANES data could have been used to train the model, rather than restricting it to the 2004, 2008, and 2012 elections. As mentioned previously, I had decided to restrict it to these elections because party trends change drastically across several decades, and thus demographic information may lose its discriminative power. However, I could have been more judicious, and included 2000 as well to bolster the training set size while still maintaining the predictive abilities of each of the features.

In addition, the target variable could have been composed in an alternative manner. I had included strong partisans, weak partisans, and partisan-leaning independents and labeled them as "Republican" and "Democrat" accordingly. I could have instead only labeled a subset of these distinctions (e.g. only strong and weak partisans) and discarded the remainder of the data. This improvement once again underlies a tradeoff- enhance a feature's predictive power at the cost of reducing the dataset size. I erred on the side of the conventional wisdom that more data correlates to a more effective predictive model.

Sources

[1] http://www.electionstudies.org/studypages/download/datacenter_all_NoData.php
[2] https://www.270towin.com/1960_Election/
[3] https://rationalwiki.org/wiki/Southern_strategy
[4] https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/
[5] https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8
[6] https://ntguardian.files.wordpress.com/2016/08/cs6350project.pdf