2016 U.S. Primary Elections

Capstone Project

Machine Learning Engineer Nanodegree

Trace Smith

April 31, 2016

Udacity

I. INTRODUCTION

PROJECT OVERVIEW

The objective of this work is to leverage machine-learning techniques to predict whom voters are most likely to elect as the party's nominee in 2016 Primary Election based on a series of demographics. A large database containing election results for both Democrat and Republican candidates from 28 primary state counties was queried and merged with demographic facts obtained from the United States Census Bureau for corresponding counties. This work examines patterns and trends in how voters cast their ballot given a larger feature space of county demographics spread throughout the United States. The objective is to generate a predictive model and fine tune it such that the algorithm learns from a series of training examples and generalizes to an out of sample testing set that correctly classifies a discrete label, the candidate winner of the respected primary county.

PROBLEM STATEMENT

In today's politics, candidates rely heavily on polls to make campaign decisions, but what if before hand we could identify which candidate is more likely to win a given county based on county statistics such as ethnicity, education, income, gender, retail sales, number of business, etc. The demographic dataset at hand does not tell who has voted, but rather who lives in the respected counties. Linking together the county facts with the winner of the 2016 primary elections of each county becomes a supervised learning problem as the features and the labels are the demographics and candidates (winners), respectively. Several learning algorithms will be explored and examined as to which classifier performs the best when predicting the class label. For example, a decision tree is constructed which goes through a series binary splitting criteria from the root node to the parent/child nodes and ultimately at the terminal leaf, attributes that were highly influential in the success of the respected candidate can be identified. Perhaps this information could provide valuable insight into how Democrats or Republicans campaigns should adjust their strategies moving forward in the remaining primaries. For instance, a candidate who is not predicted to perform well in heavily populated counties could increase spending on advertisement in order to enhance their support. As we navigate through the remaining primary states, any edge a candidate can have to boost their support base could potentially be the difference in accumulating the required number of delegates to become the party's nominee. In this work, analysis in voter trends for both parties will be discussed in detail, however the machine-learning algorithm only considers the Republican Party and the current nominees still present in the 2016 race for the White House. Those candidates will each be a class label and are as follows: Businessman Donald Trump, Senator Ted Cruz (R-Texas), and Governor John Kasich (R-Ohio). The same workflow outlined in this report could also be implemented for the Democratic Party.

METRICS

The aim of supervised machine learning is to generate a model that makes predictions based on evidence in the presence of uncertainty. The model will learn from the training set and generalizes to unseen instances (testing set) so that the model correctly determines the class label, candidate winner. Such instances would represent demographics for the remaining primary states. The performance of the classifier will be evaluated by examining the F1 score.

Being that votes are not distributed uniformly among the candidates, the labels are characterized as unbalanced. For instance, Donald Trump has won nearly 40% more counties than Senator Ted Cruz. Other performance metrics are available from Sci-Kit Learn such as Accuracy, but it is often the case that when the classes are unbalanced the accuracy score will be higher as it computed by taking the ratio of the number of correct predictions out of the total examples. Metrics such as precision and recall would be better choices for multiple, unbalanced classes. Precision is defined by taking all of the examples the classifier labeled as positive, and then determines what percentage was correctly predicted. As the precision score increases, this will result in a lower false positive rate. Alternatively, recall is the ratio of the number of true positives to the number of true positives plus the false negatives. A higher recall score yields a lower false negative rate. The F1 score is just the harmonic mean between precision and recall and is computed by the following equation: 2*[(precision * recall) / (precision + recall)]. To output a single F1 score, a weighted average of the scores are calculated using the 'weight' method. This means that the more instances a class has, then the more importance the F1 score is for this class (i.e. Trump). Therefore, the F1 score will be the basis to determine how well the classifiers performs. Included in the code for both classifiers is a classification report, which lists the precision, recall, and F1 score for all three classes. Additionally, a confusion matrix heatmap is included in the script that allows one to visually observe the accuracy of the classifier.

II. ANALYSIS

Data Exploration

The 2016 Primary Election for POTUS dataset is obtained from Kaggle, which includes a large database of election results (CNN) and demographics (U.S. Census Bureau). The 'primary_results' table in the 'database.sqlite' file is 9x13212 and the 'county_facts' is 51x3143, with 9151 Republican and 4060 Democrat elements. The county facts table contains demographics such as age, race, sex, economic status, education, income, and may other attributes; the complete list of the feature abbreviations and descriptions can be found in the Appendix. All of the features are sampled from a population per each individual county and are recorded as integers (non-categorical) with units ranging from dollars, square miles, and percentage; there are no missing values in tables. To query the database and create additional tables, a relational database management system SQLite is utilized. Calling the `sqlite3` function from the python standard library provides the interface for accessing the SQLite database. In order to use the structured query language in the Ipython notebook, SQLite magic command is required (pip install ipython-sql). Additionally, 'pandas' is also used several times in the script to read a SQL command and stores the query into a DataFrame. Listed in the code are the queries were implemented in the command line to create tables and output queries in .csv format. The first table in the database consists of the primary results from 28 primary, with the most recent results from the Nevada caucus. The number of votes each candidate received, for both the Republican and Democrat parties, is listed in this table with respect to the county and state the votes were allocated. The fraction of votes refers to the percentage of votes the candidate received out of the total number of votes casted in the corresponding county. After several queries two tables were created for each party, which consisted of selecting the candidate who was allotted the highest fraction of votes per each primary county. Once a table of the winner has been created (`Democrat_Winner`, `Republican_Winner`), each table is subsequently linked with the demographics of the analogous county. Shown below in Tables 1-3 are screenshots for each table stored in a

pandas DataFrame. Table 4 below shows a screenshot of the descriptive statistics for several of the demographics included in the analysis. From this table we can view statistics such as quartiles and the mean population, age, sex, etc. For example the average population with individuals above the age of 65 (AGE775214) is 17%. However, these statistics are rather skewed being that the computations are taking into account counties that consist of both large metropolitans such as Harris County (Houston, Texas) and may not be an accurate representation of the less populated regions across the country. As a clear conclusion is not drawn from the descripting statistics, the next section will discuss an alternative approach taken to explore relationships between the candidates and demographics.

	state	state_abbreviation	county	fips	party	candidate	votes	fraction_votes	county_state
0	Alabama	AL	Autauga	1001	Republican	Donald_Trump	5387	0.445	Autauga_AL
1	Alabama	AL	Autauga	1001	Republican	Ted_Cruz	2482	0.205	Autauga_AL
2	Alabama	AL	Autauga	1001	Republican	Marco_Rubio	1785	0.148	Autauga_AL
3	Alabama	AL	Autauga	1001	Republican	Ben_Carson	1764	0.146	Autauga_AL
4	Alabama	AL	Autauga	1001	Republican	John_Kasich	421	0.035	Autauga_AL

Table 1: Primary Results

PST045214	PST040210	PST120214	POP010210	AGE135214	AGE295214	AGE775214	...	SBO015207	MAN450207
55395	54571	1.5	54571	6.0	25.2	13.8	...	31.7	0
200111	182265	9.8	182265	5.6	22.2	18.7	...	27.3	1410273
26887	27457	-2.1	27457	5.7	21.2	16.5	...	27.0	0
22506	22919	-1.8	22915	5.3	21.0	14.8	...	0.0	0
57719	57322	0.7	57322	6.1	23.6	17.0	...	23.2	341544

Table 2: Demographics (county_facts)

candidate	votes	fraction_votes	county_state	PST045214	PST040210	PST120214
Donald_Trump	5387	0.445	Autauga_AL	55395	54571	1.5
Donald_Trump	23618	0.469	Baldwin_AL	200111	182265	9.8
Donald_Trump	1710	0.501	Barbour_AL	26887	27457	-2.1
Donald_Trump	313	0.565	Bullock_AL	10764	10915	-1.4

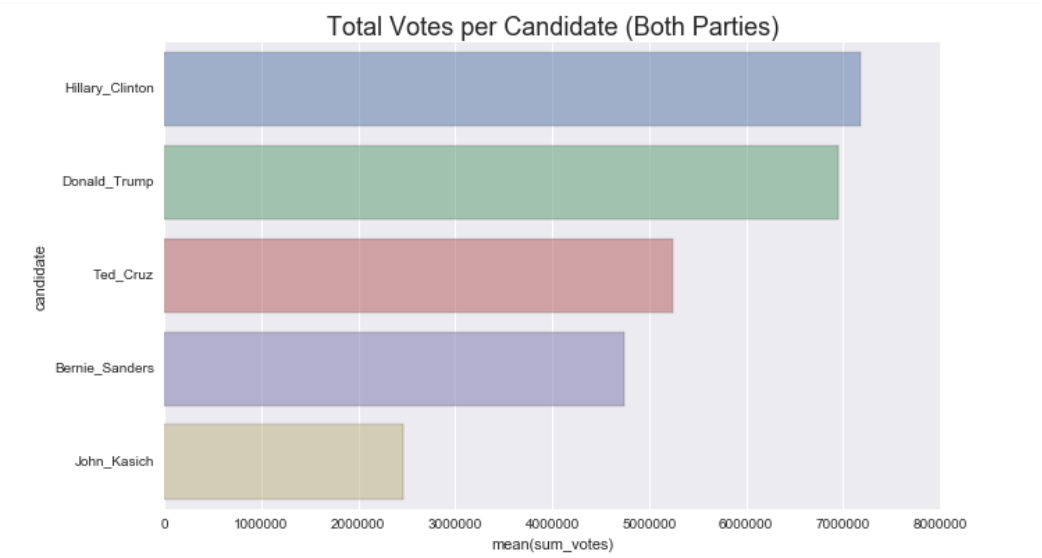
Table 3: Join Republican Winner with Demographics

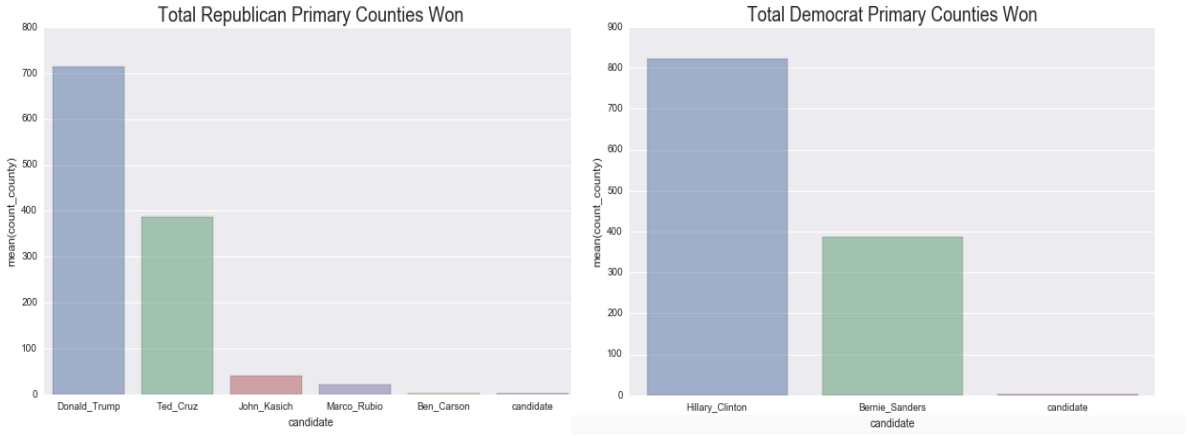
	PST040210	PST120214	POP010210	AGE775214	SEX255214	RHI125214
count	1.131000e+03	1131.000000	1.131000e+03	1131.000000	1131.000000	1131.000000
mean	1.036587e+05	0.847480	1.036472e+05	17.146950	50.066313	83.006278
std	2.649025e+05	4.200434	2.648904e+05	4.622498	2.342683	15.877681
min	8.200000e+01	-10.800000	8.200000e+01	4.100000	30.100000	22.200000
25%	1.433000e+04	-1.900000	1.433150e+04	14.200000	49.700000	75.350000
50%	3.273500e+04	0.000000	3.273500e+04	16.700000	50.600000	89.300000
75%	8.610150e+04	2.900000	8.610150e+04	19.200000	51.200000	95.000000
max	4.093011e+06	22.400000	4.092459e+06	52.900000	54.500000	98.800000

Table 4: Statistics of Republican Winner & Demographics

Exploratory Visualization

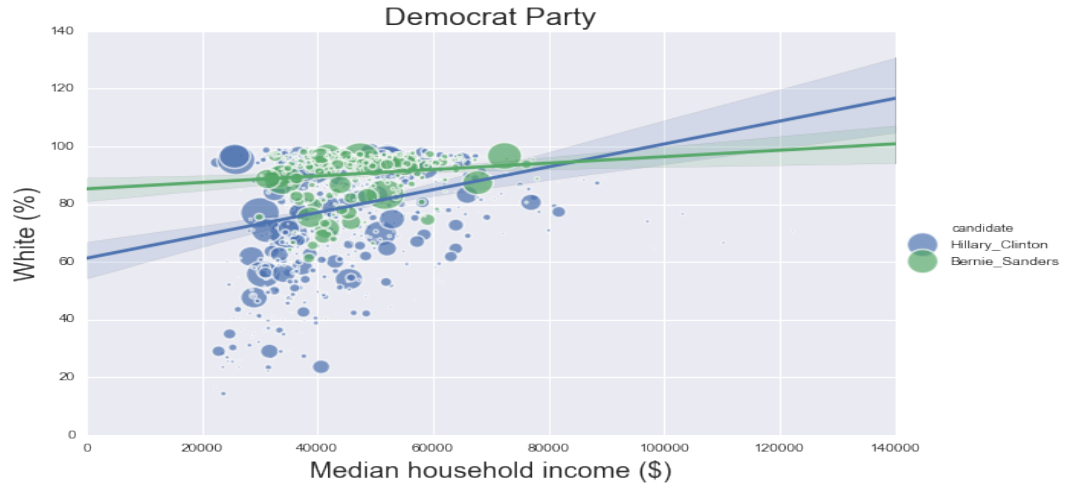
In summary of the available data, Hillary Clinton is currently the leading nominee for the Democratic Party, while Donald Trump remains in first place for the Republicans. Comparing all of the candidates from both parties thus far, Clinton maintains a slight edge over Trump in the total number of votes accumulated. The two frontrunners maintain significant leads in the vote count in contrast to their respected counterparts as shown in the plot below. Moreover, both Clinton and Trump hold a convincing lead in the number of county victories where they received the highest number of votes. The plots provide an overview of how the candidates have performed thus far in the 2016 Presidential Primary race.

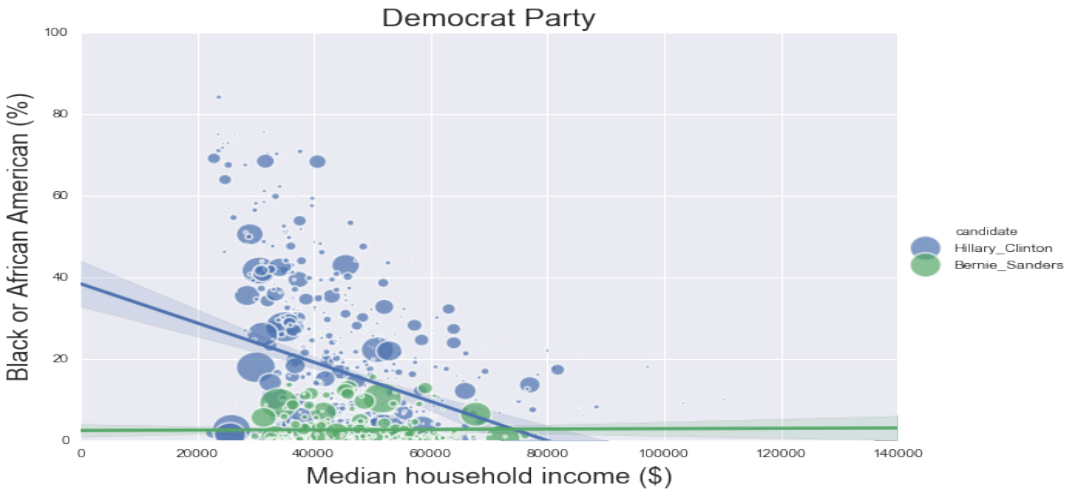




It's not the number of counties that a candidate wins to become the party's nominee, rather accumulating the required number of delegates that are allotted by the individual states. The Democratic candidate needs 2,383 delegates (1,931 remaining) to win the nomination while the Republican nominee needs 1,237 (852 remaining). Still a long way left to go for both parties, especially the Republicans as a potential brokered convention looms if a candidate does not reach the magical number of 1,237 delegates. The attempt now is to combine all of the election results filtered by the winner of each county with the marker size representing the magnitude of votes allocated. Being the vast number of pairwise combinations, several interesting features were explored to determine whether a strong or weak correlation exists among the candidate

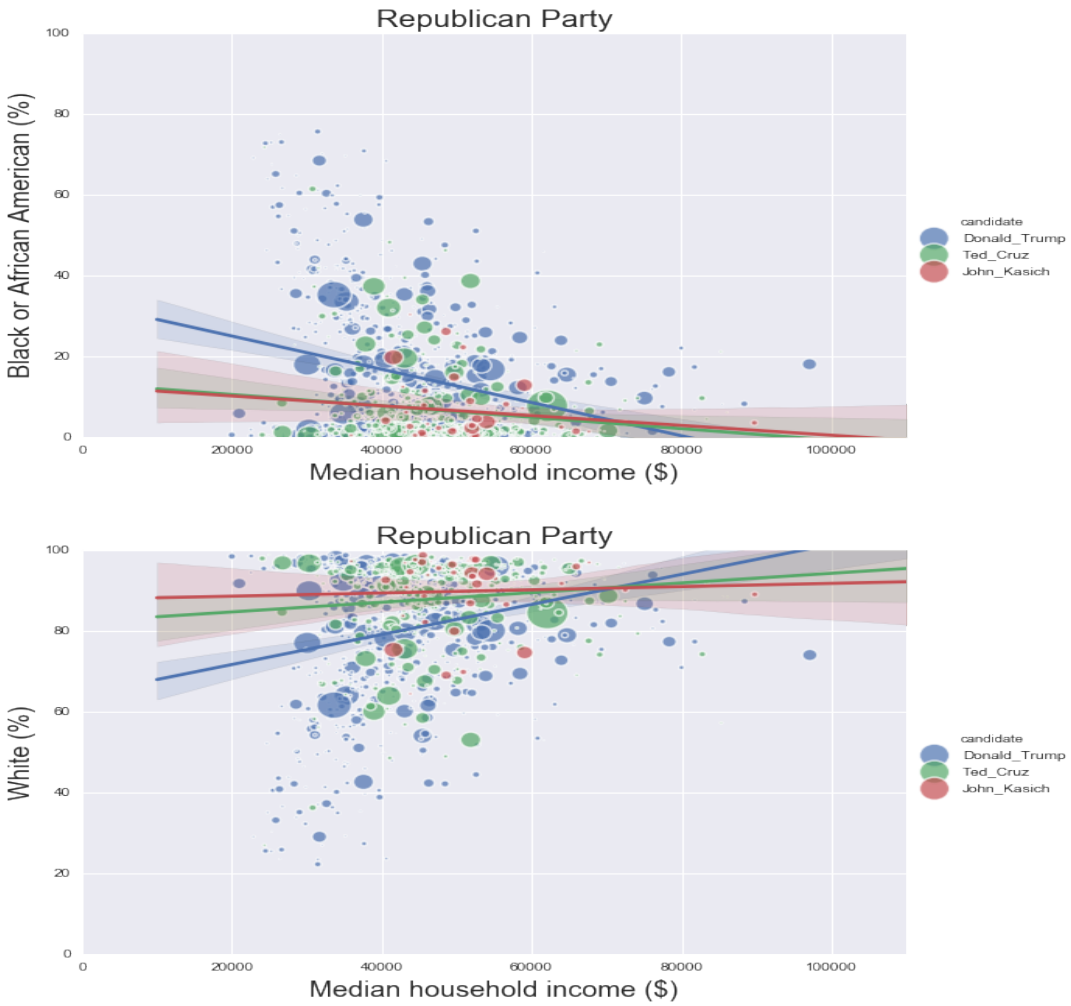
with the associated attribute. Given this knowledge, campaigns can then adjust their strategies to appeal towards certain demographics such that it enhances a candidate's chances of accumulating more votes. Receiving the highest fraction of votes in a primary state leads to the candidate increasing his/her delegate count and moving that much closer into securing the nomination for their respected party. The trends observed from the 28 primary states thus far provide a benchmark on how citizens will potentially elect a nominee throughout the remaining primaries/caucuses, baring any political shakeup. The scatter plots below compare several of the demographics with the markers representing the magnitude of votes the winning candidate received for the resultant county. Hence, larger maker sizes references more densely populated areas (i.e. higher number of votes). In some cases, the medium size markers could still represent a larger populated area with a low voter turnout ratio; voter turnout information is not available in this dataset. There are numerous combinations of voter trends that can be observed based on the given attributes, therefore the following plots explore the some of the relationships such as education, income, race, and ethnicity. From the figures below, Clinton performs better in areas with lower percentage of whites and those who possess college degrees compared to Senator Sanders who appeals to a population with higher percentages of college degrees and whites. It is also observed that Clinton does significantly better in regions with a higher percentage of African Americans with a median household income ranging between $20-50K.





On the other side, it would be interesting to see how Trump performs compared to Senator Ted

Cruz with Hispanics, given their plans regarding illegal immigration, which has been one of the leading political topics in the 2016 race. Trump outperforms Cruz in lower populated Hispanic regions (larger percentage of votes), however the trend does indicate Cruz performs slightly better in higher populated Hispanic regions with an annual median income ranging between $30-50K. Finally, comparing the percentage of whites versus the median income, Trump appeals more to areas with less household income and percentage of whites. Majority of Cruz's success ranges with populations receiving $40-60K with a significantly higher white population that ranges from 70% to nearly 100%. Additional correlation plots can be found in the `analysis.ipynb` file.

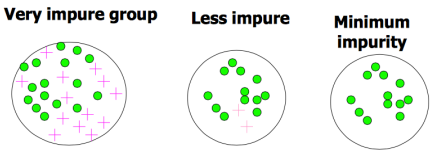


Algorithms and Techniques

In this work, the following learning algorithms were included: DecisionTreeClassifier, Support Vector Machines, and Random Forest; the classifiers call the respected functions from Sci-Kit Learn. The objective of the models are to predict the value of a target variable, in this case is the Republican candidate, by learning decision boundaries inferred from the list of attributes. The DecisionTreeClassifier is included because classification trees are simple to interpret and provide a useful visualization, but is not the best supervised learning approach for predicting accuracy compared to other methods such as Random Forest or boosting. Classification trees can become bushy and quite noisy, thus needing to be pruned back to an

optimal size. The DecisionTreeClassifier can handle instances when multi-classes are present, as in the case of this target data set where the number of classes is three; meaning the classification algorithm makes the assumption that each sample is assigned to one and only one label, but not both at the same time. The decision algorithm begins at the top of the tree (root) and is a top down approach that is known as binary splitting (i.e. True/False) where the data samples are split on the feature that results in the largest information gain. Information gain is a mathematical way to capture the amount of information one gains by picking a particular attribute. Computing the difference between the entropies before and after the split yields the information gain. Entropy is the measurement of impurities of a collection of examples and is calculated as follows:

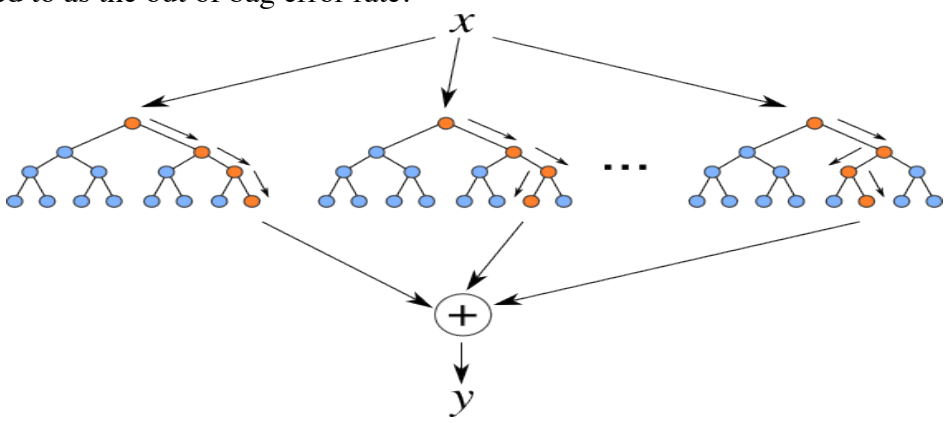
$$Entropy: -\sum_i p_i log_2 p_i$$



Source:http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf

Where p is the probability of class i and is computed as the proportion of class in the set. The attributes that maximize the information gain the most is selected as the root node and then the samples are split into the respected branch, True or False. A new node is created at for each branch and this iterative process and splitting procedure is continued at each child node until the leaves are pure.

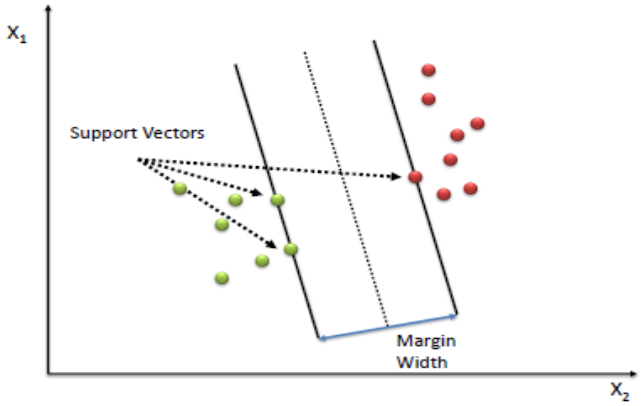
Random Forest follows the same methodology as decision tress, however this algorithm uses a large number of decision tree models from bootstrapped aggregated datasets, which improves the predictions by reducing the variances and bias of the estimates, especially for unbalanced data. This method grows multiple trees, which are then ensemble to yield a single consensus prediction as shown in the figure below. The algorithm is similar to that of the BaggingClassifier, however the Random Forest provides improvement to the model by de-correlating the decision trees. The algorithm first begins boostrapping the dataset of observations and generating a large number of trees, with each one being different. The model entails that each time a tree is split, a random selection of "m" observations per feature is chosen as split candidates from the full dataset. For every tree that is grown on a bootstrap sample, the error rate for observations that are left out of the bootstrap sample is monitored and is referred to as the out of bag error rate.



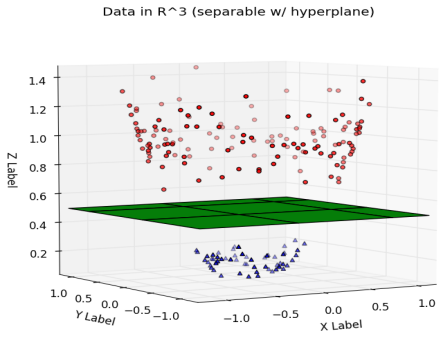
Furthermore, the runtimes are relatively fast when just running the Random Forest classifier without including GridSearch; run times significantly increase when adding the optimization method to roughly 5-10 minutes with only 30 trees. The algorithm can be used with large or

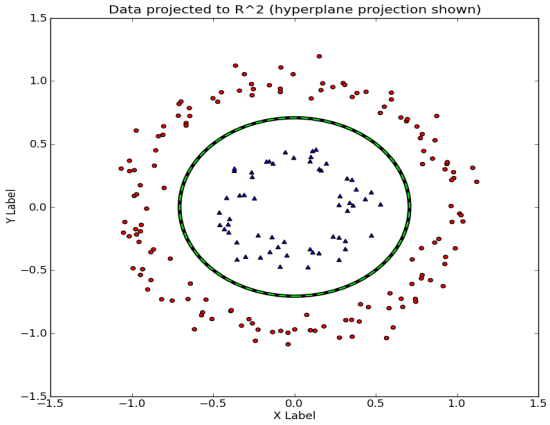
small datasets and with a larger number of attributes; Republican DataFrame (`df_rep`) consists of 51 different features and 1131 elements. Compared to the DecisionTreeClassifier, Random Forest almost always has a lower classification error and better F1-scores.

The final classifier included in the analysis is SVM, which is a classification algorithm that constructs hyperplanes in a high dimensional space and separates two different class labels. There are many possible solutions on where to draw the decision boundary, such as the solid line on the left or to the right. Both of these lines are too close to the existing observations and if new observations were added, it would be unlikely they behave precisely like the initial data. Therefore, SVM's reduces the risk of selecting the wrong decision boundary by choosing the line that has the largest distance from the bordering data points of the two classes. Having the additional space between the groups reduces the chance of selecting the wrong class. As shown in the figure, the largest distance between the two groups of students is called the margin. The margin width is determined by the points that are present on the limit of the margin, known as the support vectors. The dashed line in the middle of the margin width would be where the decision boundary would be.



Not all data behaves linearly and therefore a linear separating hyperplane will not work. A special technique can be applied so that transforming the input variables can separate the non-linear datasets, which results in the shape of the dataset to become more linear. This is known as the kernel trick. A kernel is essentially a mapping function in which we take the input variables and transform into a much higher dimensional space. After separating the data using complex separating plane known as hyperplanes (shown below on the left), you take the solution and return back to the original space, and now you have a non-linear separation. SVM's can use a complex separating plane known as hyperplanes as shown below.





Benchmark

Due to the large dataset, running optimization techniques like Grid Search can result in significantly longer run times due to tuning multiple combinations of hyperparameters. It is expected that Random Forest would result in longer run times than SVM, due to the quantity of specified trees which can range upwards of 300 in some cases. Finding the optimal number of trees (n_estimators) will be examined in the analysis in order to address this issue and reduce the run time. The tradeoff between which model ultimately performs the best will be the F1 score and the computational run time. These two measurements will be the threshold in evaluating the models.

III. METHODOLOGY

Data Preprocessing

Initially the tables is the database were not assigned with a primary or foreign key, thus as a quick fix instead of recreating the tables, a single column was added that consisted of merging the county and state abbreviation columns together to give a unique ID name to both the primary results and county facts table (i.e. `county_stateabbrev`); the two tables can now be linked together. Furthermore, this predictive model will focus only on candidates who are currently still in the race: Donald Trump, Ted Cruz, and John Kasich. Those candidates whom have suspended their presidential campaign were removed from the 'republican_data' DataFrame: Marco Rubio, Ben Carson, Jeb Bush, Rick Santorum, Rand Paul, Chris Christie, Carly Fiorina, and Mike Huckabee. An additional step in the data preprocessing was to remove columns that were repetitive when joining the Republican_Winner and county_facts table (i.e. state, state abbreviation, county). A final step is then to designate the target column and feature columns. Being that Republican_Winner table contained columns that included state, county, etc. that are not part of the features, some additional work was required to reorganize the data such that two separate DataFrames were constructed; one containing the features and the other contains the target column Using the concat function in pandas allows the two separate DataFrames to then be re-joined together in the desired format. After slicing the newly created DataFrame, `df_rep`, X_all stores all of the features and y_all stores the target column.

Implementation

As previously mentioned, the dataset is unbalanced, thus is important to address splitting the data into training and testing when initially calibrating the model. Therefore, after preprocessing the data, applying the cross-validation technique `StratifiedShuffleSplit` helps deal with the imbalance number of classes in the dataset by allowing the training and testing set to have roughly the same ratio of candidate winners; the ratio will not be evenly distributed for John Kasich due to the low number of counties won (39) so far in contrast to his counterparts Trump and Cruz who have won 714 and 387, respectively. The data is also shuffled to remove any bias in the order of candidates. `StratifiedShuffleSplit` will randomly select training and testing sets multiple times and then average the results over all the tests. Seventy-five percent of the dataset has been assigned as the training set and the remained 25% will be used as the testing set (training samples = 848, testing samples = 283). After splitting the features and target as training and testing data, the DecisionTreeClassifier, Random Forest, and SVM classifiers are implemented and fine-tuned so that the optimal performance

is achieved. In the following section, the results obtained from each model will be discussed in detail.

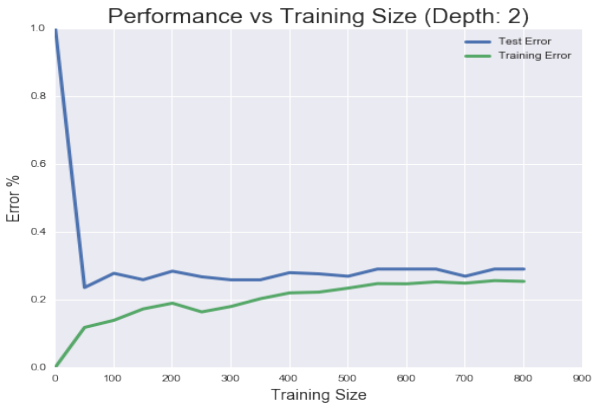
Refinement

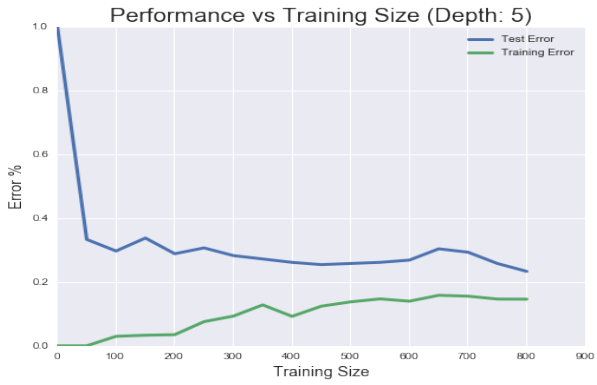
Decision trees tend to overfit the data, but determining the minimum number of terminal leaves, minimum number of splits, and max_depth can reduce the variance of the estimates. As it is desired to optimize the model and increase the performance of the learning algorithm when generalizing to unseen instances (testing set), one method is to implement the Grid Search algorithm, a hpyerparameter optimization technique. Grid Search is an exhaustive search that trains and evaluates a model for all possible combination of hyperparameters that produce the best model by tuning the parameters of the Random Forest model such as leaves/splits and max_depth. When discussing max_depth, this refers to maximum number of binary queries that can occur before a leaf is reached. The other two hyperparameters tuned are the `min_samples_leaf` , the samples required to be at a leaf node, and `min_samples_split` minimum number of samples required for splitting an internal node. Grid Search is also used in the refinement for the SVM model and several of the parameters tuned are C and Gamma. The value of C controls the tradeoff between a smooth decision boundary and classifying training points correctly. The higher the C value, the more intricate the decision boundary becomes. Second, Gamma defines how far the influence of a single training example reaches; low values meaning 'far' & high values meaning 'close'. If the value of gamma is too small then the model is too constrained and cannot capture the complexity or "shape" of the data.

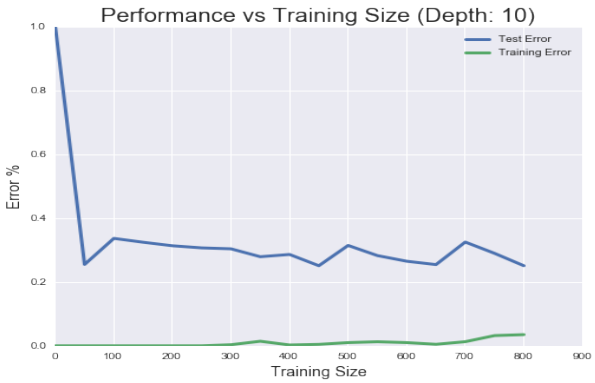
IV. Results

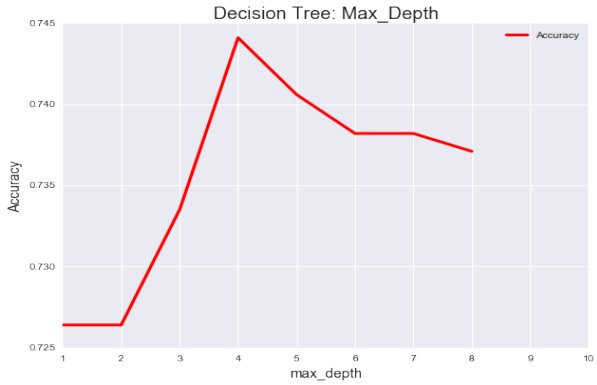
Model Evaluation and Validation

DecisionTreeClassifier: As the decision tree grows and becomes more complex the issue of overfitting arises. In this scenario, the model has virtually memorized the training data but will not be expected to perform well with out-of-sample data. On the other hand, if the tree is too simple then this could result in underfitting as the learning value is restricted to one level of the decision tree and does not allow the training set to learn the data adequately; a lower complexity decision tree results in high bias. To address the effect of either over or under fitting, a plot of the maximum depth of the tree versus the accuracy error (1-accuarcy) was generated. As observed form the figures below, the training error is virtually zero as the model has basically memorized the small training set. However, as more data is added to the model, the training error begins to increase. When the training size increases, error begins to increase and the high error translates to underfitting the data; likewise, the low training error would indicate overfitting the data. The testing error is relatively high initially because it has not seen enough examples, but begins to decrease as more training examples are given. With that said, the best max_depth to consider for the classifier is 5. Additionally, a plot was created of the max_depth vs the cross-validation score (scoring = 'accuracy') located in the bottom right (red line). Cross-validation is the process of dividing the data into folds (cv = 10) and hold out one of the folds as a test set and the remaining folds are used as the training set. This process is repeated until each fold is a test set and the score is then averaged. This graph shows that by increasing the complexity of the tree beyond a max_depth of 4-5 reduces the classifier's performance. As a note, when implementing CV, the cross_val_score performed the best when taking the training set (X_train, y_train) as an input, which had already been split using Stratified_Shuffle_Split(). In summary of the DecisionTreeClassifier, the corresponding F1 score after applying GridSearch was 0.739 and the run time took 19 seconds for five different simulations, averaging roughly 4 seconds per run.

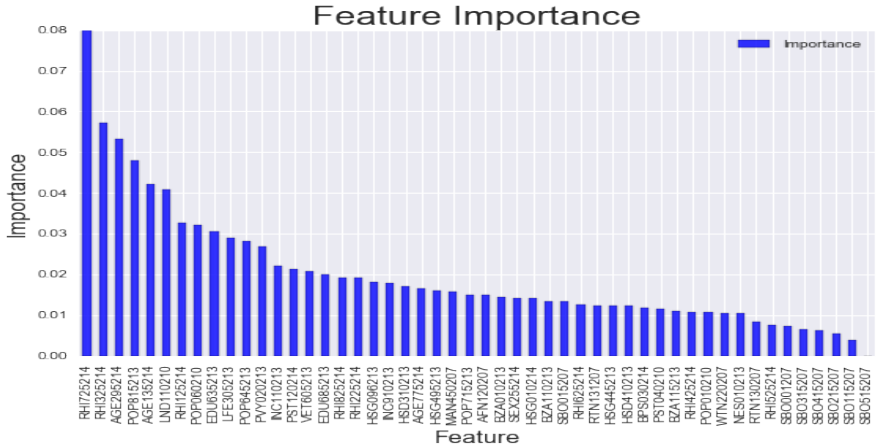








Random Forest: Predictions of decision trees tend to have a higher variance which is why the Random Forest algorithm was included in this work. The addition of more trees can result in reducing the overall variance and does not tend to overfit as easily. Also, being that our dataset contains a large set of features, a useful approach to determine which features are the most relevant is to use Random Forest by calling the `clf.feature_importances_ function` from Sklearn. From the bar plot below, the top three most relevant features are Hispanic population, age population under 18 years old, and total land area (square miles). A larger land mass feature makes sense, being that a larger magnitude of votes would be expected to come from that region, thus being more influential than less populated areas. The remaining features decline exponentially and remain relatively flat after the first 4-6 features. It is implicit that irrelevant features can reduce the accuracy and efficiency (run time) of the classifier. In the Random Forest model, it was decided to include all of the features; preliminary runs with only the first 10-15 features did not result in significantly improving the F1 score.



Initially running the Random Forest model without fine-tuning and with 100 trees yielded an average F1 score of 0.79 on the testing samples after 10 iterations, which is as slight improvement to the DecisionTreeClassifier. Next, in an attempt to optimize the performance of the RF classifier, GridSearch was applied and the following hyperparameters were tuned:

'Max_depth' [3,4,5,6,7]
'Min_samples_leaf': [2,4,6,8,10,15,20,25,30]
'Min_samples_split': [2,4,6,8,10,15,20,25,30]

Once tuning the hyperparameters with 100 trees (n_estimators), the F1 score was 0.78, slightly better than the DecisionTreeClassifier, with a run time of 18 minutes. This result is for a single iteration, so the average of say 10 different iterations could be slightly higher and closer to 0.80. Reducing the number of trees down to 50 would drastically reduce the run time. Note: if we take a look at the OOB error rate vs the number of trees, the error rate begins to plateau when more than 50 trees is added to the forest, hence reducing the n_estimators to 50 could cut the computation run time down quite a bit without affecting the performance.

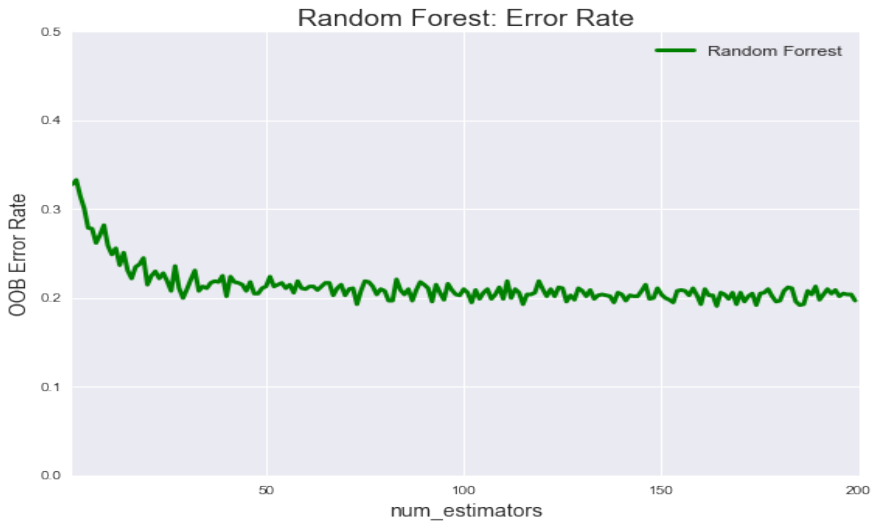
Support Vector Machines: The SVC classifier performed the worst in comparison to the decision tree and random forest. The run time was relatively fast, 1 a total completion time of 1 minute for 10 iterations; however a resulting F1 score of 0.48 was observed. The following tuning parameters were included in the GridSearch:

'C': [1,10,50,100,200,300,400,500,1000]
'Gamma': [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1]
'Kernel': ['rbf']

Since linear SVC attempts to separate the classes by finding appropriate hyperplanes in Euclidean space, perhaps the data is not linearly separable which is resulting in the poor performance of the classifier. Therefore the SVM classifier would not be a valid model to use with this dataset.

Justification

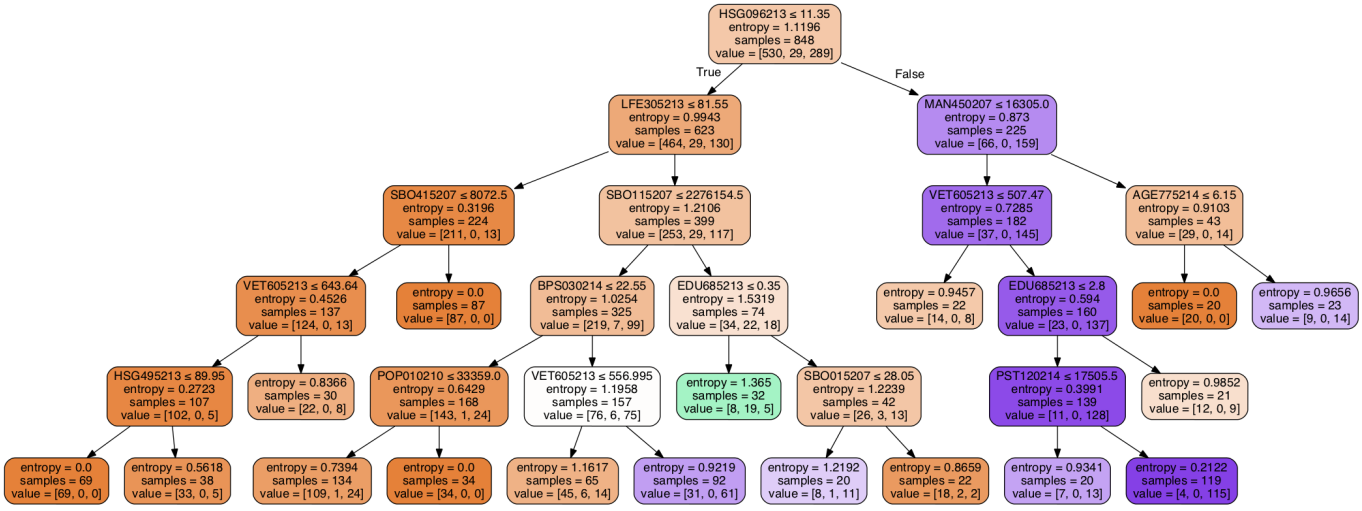
To validate the Random Forest model, we will take a look at the out-of-bag (OOB) error versus the number of trees. First, we do not need to apply any type of separating test (i.e. cross-validation) to get an unbiased estimate of the testing error; the random forest algorithm already addresses this. As each tree is constructed from a different bootstrap sample of the original dataset, each tree is trained with about 2/3 of the total training set. The next step is to predict the classification of the remaining 1/3 using the tree, and calculate the misclassification rate (out of bag error rate). The OOB error from the entire forest is then aggregated to determine the overall error rate. From the plot below, we can observe that the as the number of trees grows beyond 50, the OOB error levels out around 20%. Therefore, the RF model correctly predicts the Republican candidate 78-80% of the time.



V. Conclusion

Free-Form Visualization

To execute the visualization of the decision tree, installing the Graphviz package is required in addition to PyGraphviz, which is a Python interface to the Graphviz graph layout and visualization package. The script to create the tree can be found under the `Construct Decision Tree` heading in the analysis.ipynb. Also, the `tree.pdf` file in the main directory contains the decision tree. A description of the features shown in the tree can be found in the Appendix. As shown below, "the number of housing units in multi-unit structures" (i.e. the percentage of apartments, townhomes in a county) is the most influential feature and is the root of the decision tree based on the DecisionTreeClassification using the splitting criteria of entropy as previously discussed. We can start at the root of the tree and move down the branches until reaching a terminal leaf in order to determine which Republican candidate is associated with demographics of a county located in the United States. The terminal or 'leaf' node at the bottom of the tree is referred to as the decision node and is dominated by one class and the value is ordered as follows [Trump, Kasich, Cruz].

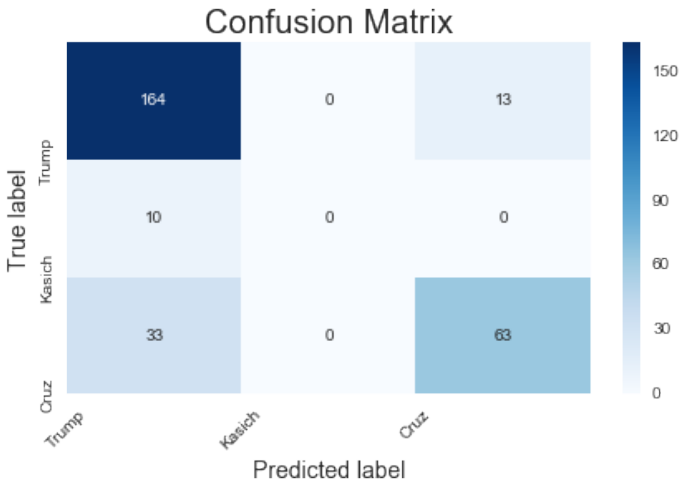


Given that the best performing model is the Random Forest learning algorithm, a confusion matrix was incorporated in the code to give a visual representation of the model's performance. The confusion matrix below is a table that summarizes how many times the classifier predicted the correct class label. Each column in the matrix represents the instances in a predicted class and the rows represent the instances that are the true class (actual). The table reports the numbers of true positives, true negatives, false positives, and false negatives; this matrix gives a more detailed analysis than calculating the accuracy, which is just computing the proportion of correct guesses. Accuracy can give some misleading results, especially when the dataset is unbalances and is not a reliable metric when examining the performance of a classifier. In the case of a 3x3 confusion matrix, the diagonals are the "true positives". We can see from that the classifier does not perform well at all when predicting John Kasich as the correct label. This is part to the limited number of examples (39 counties won) compared to Trump or Cruz. The precision for Cruz and Trump is relatively high with scores of 0.83 and 0.79, respectively; Trump has the highest F1 score, 0.85. Illustrated in the confusion matrix below, we can see instances where Cruz was incorrectly labeled as Trump 13 of 63 times and hence bias exiting towards Trump and lowers the F1 score. What stands out from the matrix is the percentage Trump was correctly predicted 80% of the time (164 correctly labeled).

```
Classification Report:
             precision    recall  f1-score   support

Donald_Trump       0.79      0.93      0.85       177
John_Kasich        0.00      0.00      0.00        10
   Ted_Cruz        0.83      0.66      0.73        96

avg / total        0.78      0.80      0.78       283
```



Reflection

The goal of this project is to initially provide correlations among Democratic/Republican candidates with respect to a large number of demographic features. As pointed out in the analysis, we can observe trends in the data such as Senator Sanders performing better among college students where there is high population of whites. Or other trends such as Trump performing better in areas with a higher percentage of African Americans with median household incomes ranging between $30-60K than Senator Cruz. This analysis is intended to provide the viewer with an understanding of the dataset at hand, such as trends in the demographics as a whole and how candidates could expect to perform with similar characteristics in other future primary counties. We can also observe which statistics yielded

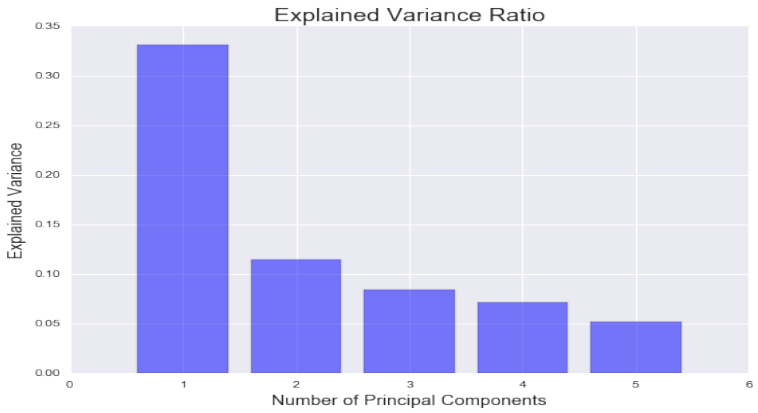
the highest magnitude of votes, which could be key indicators as to which features are highly influential for that candidate's success. From there, the goal is to generate a predictive model that learns from previous data (training set) and generalizes to the future out of sample data that yields the lowest classification error as possible. Before using the model presented in this work in a general setting in which campaign leaders of the respected candidates could draw predictions from in regards to adjusting their strategies for the remaining 2016 race to the White House, I would like to see the performance of the classifier increase; this could entail implementing another classification algorithm other than the ones described in this analysis. Additionally, adding more features such as the percentage of males, various age groups, or unemployment rate just to name a few, could shed more insights into how a given region, state, or county would tend to elect as the next party nominee and ultimately the next President of the United States.

Improvement

One aspect of this project that could be improved would be extending the analysis to the Democratic Party and also draw correlations between demographics and the likelihood of a particular party, either Republican or Democrat, winning a given county; this would be useful especially in the general election. Another step that could be added in the code is to run multiple iterations when optimizing the Random Forest model and find an average F1 score. Only a single iteration was considered due to run times lasting nearly 20 minutes. This could give a more accurate representation of how the model performs rather than running a single iteration, which has a small degree of variance for each run. In regards to producing the optimal model that generalizes to the unseen instances, implementing a boosting classifier, such as AdaBoost or Gradient Boosting Classifier could potentially perform better with this dataset in contrast to the current Random Forest model. In Boosting, the examples that are incorrectly predicted by the previous classifiers in the sequential trees are chosen more often than those examples that were correctly predicted. Thus boosting attempts to produce new classifiers that are better able to predict examples for which the current ensemble's performance is poor. Perhaps this improves the model classification performance, especially for John Kasich, which was incorrectly labeled 100% of the time.

It should also be mentioned that reducing the high dimensional space using PCA was explored. As in the case of this dataset, in the presence of a large number of attributes, it may be useful to reduce the dimensions with an unsupervised learning method such as PCA prior to the supervised steps. The idea here is to determine whether or not the performance of the Random Forest learning algorithm improves by applying dimensionality reduction. The first step is to standardize features by removing the mean and scaling to unit variance since the data is comprised of various units. The eigenvectors of the covariance matrix represents the principal components (directions of maximum variance and determined the direction of the new feature space) and the eigenvalues (scalar) correspond to the magnitude of the eigenvectors. This means that the eigenvalues explains the variance along the new feature axes. From the table below, it can be observed that PST040210 and POP010210 have the high magnitudes and seem to be more influential. The first two principal components only account for 45% of the total variance and the first 5 accounting for nearly 65%. The training data features were projected onto the top several principal components and then the transformed dimensional space (n_components = 4) was passed to the Random Forest classifier. The resultant F1 virtually remained the same as the fine-tuned RF model with a score of 0.78.

```
Principal Component Analysis:
     PST040210  PST120214  POP010210  AGE775214  SEX255214  RHI125214  \
PC
0     0.242181   0.107170   0.242179  -0.078238   0.037342  -0.036180
1    -0.017193   0.072816  -0.017205   0.112771   0.077200   0.193108
2     0.077383  -0.255310   0.077381   0.165810   0.115741  -0.197318
3     0.088180  -0.085285   0.088187   0.204078  -0.113436   0.336093
4    -0.017352   0.030162  -0.017359  -0.031587   0.069262   0.039556
```




```
Classification Report:
              precision    recall  f1-score   support

Donald_Trump       0.80      0.93      0.86       177
John_Kasich        1.00      0.10      0.18        10
   Ted_Cruz        0.78      0.64      0.70        96

avg / total       0.80      0.80      0.78       283


F1 Score: 0.783
```

Principal Component Analysis:

APPENDIX:

Feature Name	Description
PST045214	Population, 2014 estimate
PST040210	Population, 2010 (April 1) estimates base
PST120214	Population, percent change - April 1, 2010 to July 1, 2014
POP010210	Population, 2010
AGE135214	Persons under 5 years, percent, 2014
AGE295214	Persons under 18 years, percent, 2014
AGE775214	Persons 65 years and over, percent, 2014
SEX255214	Female persons, percent, 2014
RHI125214	White alone, percent, 2014
RHI225214	Black or African American alone, percent, 2014
RHI325214	American Indian and Alaska Native alone, percent, 2014
RHI425214	Asian alone, percent, 2014
RHI525214	Native Hawaiian and Other Pacific Islander alone, percent, 2014
RHI625214	Two or More Races, percent, 2014
RHI725214	Hispanic or Latino, percent, 2014
RHI825214	White alone, not Hispanic or Latino, percent, 2014
POP715213	Living in same house 1 year & over, percent, 2009-2013
POP645213	Foreign born persons, percent, 2009-2013

POP815213	Language other than English spoken at home, pct age 5+, 2009-2013
EDU635213	High school graduate or higher, percent of persons age 25+, 2009-2013
EDU685213	Bachelor's degree or higher, percent of persons age 25+, 2009-2013
VET605213	Veterans, 2009-2013
LFE305213	Mean travel time to work (minutes), workers age 16+, 2009-2013
HSG010214	Housing units, 2014
HSG445213	Homeownership rate, 2009-2013
HSG096213	Housing units in multi-unit structures, percent, 2009-2013
HSG495213	Median value of owner-occupied housing units, 2009-2013
HSD410213	Households, 2009-2013
HSD310213	Persons per household, 2009-2013
INC910213	Per capita money income in past 12 months (2013 dollars), 2009-2013
INC110213	Median household income, 2009-2013
PVY020213	Persons below poverty level, percent, 2009-2013
BZA010213	Private nonfarm establishments, 2013
BZA110213	Private nonfarm employment, 2013
BZA115213	Private nonfarm employment, percent change, 2012-2013
NES010213	Nonemployer establishments, 2013
SBO001207	Total number of firms, 2007
SBO315207	Black-owned firms, percent, 2007
SBO115207	American Indian- and Alaska Native-owned firms, percent, 2007
SBO215207	Asian-owned firms, percent, 2007
SBO515207	Native Hawaiian- and Other Pacific Islander-owned firms, percent, 2007
SBO415207	Hispanic-owned firms, percent, 2007
SBO015207	Women-owned firms, percent, 2007
MAN450207	Manufacturers shipments, 2007 ($1,000)
WTN220207	Merchant wholesaler sales, 2007 ($1,000)
RTN130207	Retail sales, 2007 ($1,000)
RTN131207	Retail sales per capita, 2007
AFN120207	Accommodation and food services sales, 2007 ($1,000)
BPS030214	Building permits, 2014
LND110210	Land area in square miles, 2010
POP060210	Population per square mile, 2010