Analysing Demographics and Predicting The Winning Political Party In US Presidential Elections

Aunsh Chaudhari
Northeastern University
aunshchaudhari@ccs.neu.edu

Akshay Singh
Northeastern University
akshaysingh1520@ccs.neu.edu

Abstract

In this project, we predict the political party – Democratic or Republican, most likely to win in a particular county of a state. Using various demographics and the preference of certain sections of the population of the county as indicators, we train a machine learning classifier written in Python with the help of a number of important machine learning algorithms such as Decision Trees, Naïve Bayes and the Support Vector Machine. The performance of our implementation is analyzed using the average accuracy achieved by these algorithms in predicting the correct winning political party. We also focus on the impact of the feature selection on each of the algorithms by determining which feature set optimizes the performance of the algorithm.

Introduction

The recently concluded US Presidential Elections were followed enthusiastically by one and all, irrespective of their nationality. Eventually, Donald Trump from the Republican party emerged victorious over Hillary Clinton of the Democrats. This brings us to the question, which factors could be significant in this win? The voter's population is made of several factions and sections of the society. Each of them have similar mindsets, beliefs and noticeably, support a particular political party that promises to fulfill their needs and resonates with their ideals.

Factors like the percentage of educated people, Black, the Hispanic population as well as the average income can help us in predicting the eventual winning political party in the county of a state. On the other hand, it is important to remember that predicting an election on the basis of the demographics of a state must also consider that there is a possibility that the entire expected percent population may not vote on the day.

Given the complexity of the task at hand, prediction in this case is simplified to a classification problem – whether the Democratic party will win the election in the county or the Republican party will. We have used different machine learning algorithms to classify each county as a pro Republican or a pro Democratic county. In the upcoming sections of the paper, the methodology and algorithms used to solve this problem are described and the results are analyzed to obtain some conclusions regarding the relevance of the demographics selected to classify as well as to discuss any future work that could be done to improve the accuracy of the predictions.

Overview Of US Presidential Election Process

The election process begins with the primary elections and caucuses and moves to nominating conventions, during which political parties each select a nominee to unite behind. The nominee also announces a Vice Presidential running mate at this time. The candidates then campaign across the country to explain their views and plans to voters and participate in debates with candidates from other parties.

During the general election, Americans head to the polls to cast their vote for President. But the tally of those votes—the popular vote—does not determine the winner. Instead, Presidential elections use the Electoral College. To win the election, a candidate must receive a majority of electoral votes. In the event no candidate receives the majority, the House of Representatives chooses the President and the Senate chooses the Vice President. [2]

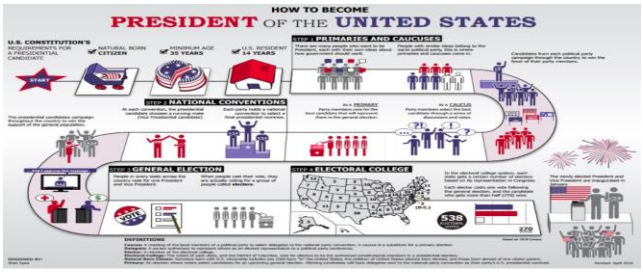

Figure 1. US Presidential Election Process [2]

Dataset

In this project, we examined a data set that consists of entries for each county that consist of the votes garnered by each political party in 2012, as well as the demographics, in terms of a percent or a number of the present population of the county at the time of voting. This dataset is provided by Kaggle [3] and includes the description of 3112 counties in terms of votes and population statistics. Specifically, each entry in the dataset includes the following attributes:

- County name
- Votes for Democrats in 2012
- Votes for the Republicans in 2012
- Votes for Democrats in 2016
- Votes for Republicans in 2016
- Persons under 18 years, percent
- Persons 65 years and over, percent
- Female persons, percent
- White alone, percent
- Black or African American, percent
- American Indian, percent
- Asian, percent
- Hispanic population, percent
- Foreign born, percent
- Bachelor's degree or higher, percent
- High school graduate or higher, percent
- Veterans
- Black owned firms,
- American Indian owned firms
- Asian owned firms
- Women owned firms
- Hispanic owned firms
- Median household income
- Persons below poverty level
- Persons per household
- Density
- Persons under 5 years, percent
- White alone, not Hispanic or Latino, percent
- Living in same house 1 year & over, percent
- Veterans
- Mean travel time to work (minutes)
- Housing units
- Homeownership rate
- Housing units in multi-unit structures, percent
- Median value of owner-occupied housing units
- Households
- Persons per household
- Per capita money income in past 12 months
- Private nonfarm establishments
- Private nonfarm employment
- Private nonfarm employment, percent change
- Nonemployer establishments
- Manufacturers shipments
- Merchant wholesaler sales
- Retail sales
- Retail sales per capita
- Accommodation and food services sales
- Building permits
- Land area in square miles
- Population per square mile
- Two or more races, percent

From the data entries of the number of votes for the Democratic and Republican party in 2016, we were able to compute the winning political party in the county, thus assigning labels to each county (Winner). This task involves binary classification where we assigned the county 1 if the Republican party won in the county and 0 if the Democratic party did.

To test the algorithm we used $1/5^{th}$ of the training set as a test set. We performed 5-fold Cross Validation, meaning that we separated our dataset into 5 buckets, and at each iteration we used one bucket as the test set and the remaining buckets as the training set.

Data Preprocessing

The features in the dataset of counties, present great variation with respect to their range, most of features are continuous and the label is categorical. In order to be able to use the features reliably and improve the efficiency, per-

formance of the classification algorithms, we preprocessed the data before the actual classification task.

Rescaling

Feature scaling is a method used to standardize the range of independent variables or features of data. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work without normalization. Since our data is comprised of attributes with varying scales, they are rescaled into the range between 0 and 1.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

General formula for Rescaling

Where x is the original value and x' is the normalized value

Feature scaling is helpful for a number of algorithms such as:

- K-nearest neighbors with a Euclidean distance measure
- Support vector machines
- Neural networks
- Logistic regression

We have used the Min-Max scaler to rescale the attributes in our dataset. [4]

Feature Selection

Some features on the dataset are correlated to the winning political party in the county. We used two techniques for feature selection from 48 features of the dataset:

Univariate Selection

We have used statistical tests to select those features that have the strongest relationship with the output variable (Winner). The chi squared (χ^2) statistical test is constructed from a sum of squared errors, or through sample variance. The chi-squared distribution arises from an assumption of independent normally distributed data. [5]

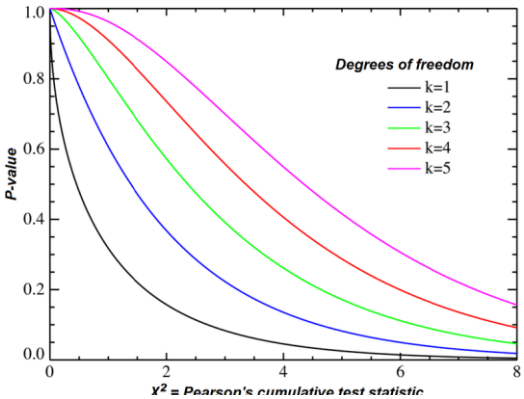

Figure 2. Pearson's chi squared test

Random Forest Feature Importance

Bagged decision trees like Random Forest and Extra Trees can be used to estimate the importance of features while selection from a large dataset. The two straightforward methods for feature selection are mean decrease impurity and mean decrease accuracy.

When training a tree, the Gini impurity or the entropy can be computed by how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged. After testing our dataset with sklearn's random forest implementation, we received features ranked according to this measure.

The other method is to directly measure the impact of each feature on the accuracy of the model. For the unimportant variables, permutation has little or no effect on accuracy whereas permuting important variables decreases the accuracy. [5]

In order to examine the correlation between the features and the impact they have on classification, we tested them by plotting them with the number of votes Donald Trump received as well as Hillary Clinton garnered to be able to understand whether any feature has a specific relation to either of the political party winning in the county.

From the two methods of feature selection above, 14 features were selected by ranking. The selected feature set is:

% White

Higher the percent of the white persons in the population of the county, higher was the vote share of the Republican party, in this case the votes for Trump. This shows the preference of the White persons for the Republicans.

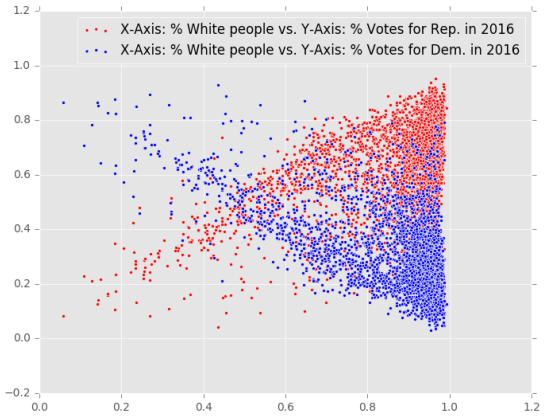

Figure 3a: Graph depicting effect of percent of white persons on each party votes

Table 3a: Correlation values for White persons

White people Vs Republicans	0.529662115927
White people Vs Democrats	-0.593048979509

% Black

Higher the percent of the black persons in the population of the county, higher was the vote share of the Democratic party, in this case the votes for Clinton. This shows the preference of the Black persons for the Democrats.

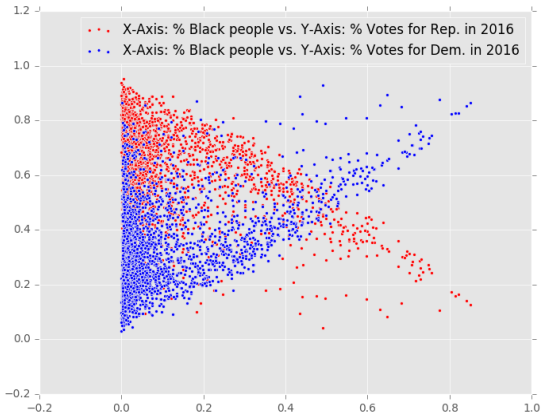

Figure 3b: Graph depicting effect of percent of black persons on each party votes

Table 3b: Correlation values for Black persons

Black people Vs Republicans	-0.425172944651
Black people Vs Democrats	0.50926898948

% Asian

Higher the percent of the Asians in the population of the county, higher was the vote share of the Democratic party, in this case the votes for Clinton. This shows the preference of the persons of Asian descent for the Democrats.

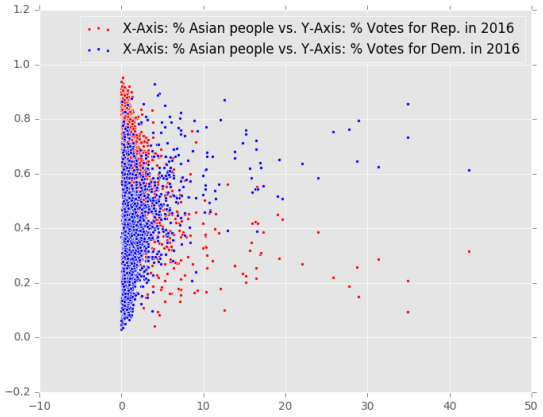

Figure 3c: Graph depicting effect of percent of Asians on each party votes

Table 3c: Correlation values for Asians

Asian people Vs Republicans	-0.446636600574
Asian people Vs Democrats	0.434614110329

% Hispanic/Latino

Higher the percent of the Hispanic/Latino in the population of the county, higher was the vote share of the Democratic party, in this case the votes for Clinton. This shows the preference of the Hispanics for the Democrats.

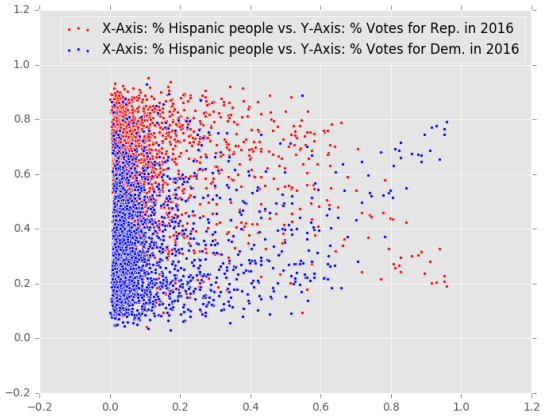

Figure 3d: Graph depicting effect of percent of Hispanic/Latino on each party votes

Table 3d: Correlation values for Hispanic/Latino

Hispanic people Vs Republicans	-0.188423041309
Hispanic people Vs Democrats	0.182939797015

% Foreign Born

Higher the percent of the people in the population of the county that were born outside the USA, higher was the vote share of the Democratic party, in this case the votes for Clinton. This shows the preference of the foreign born persons for the Democrats.

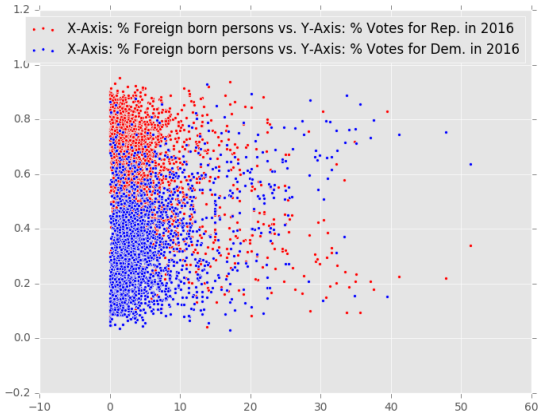

Figure 3e: Graph depicting effect of percent of Foreign born persons on each party votes

Table 3e: Correlation values for Foreign born persons

Foreign born persons Vs Republicans	-0.395191096738
Foreign born persons Vs Democrats	0.391563485748

% NonEnglish

Higher the percent of the people who speak a language other than English at home in the population in the county, higher was the vote share of the Democratic party, in this case the votes for Clinton. This shows the preference of the Non English speaking persons for the Democrats.

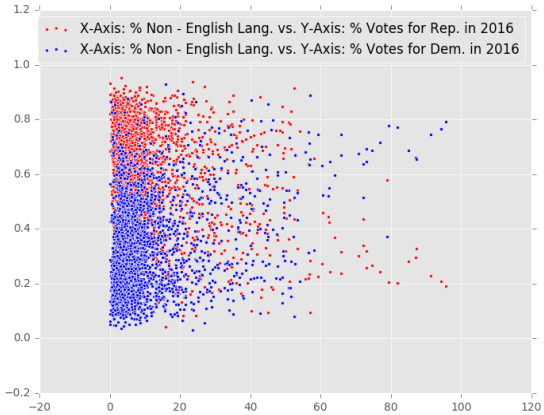

Figure 3f: Graph depicting effect of percent of non English speaking persons on each party votes

Table 3f: Correlation values for non English speaking persons

Non - English Lang. Vs Republicans	-0.326459278674
Non - English Lang. Vs Democrats	0.320319960268

% High School Graduate

Higher the percent of the persons that are high school graduates in the population in the county, higher was the vote share of the Republican party, in this case the votes for Trump. This shows the preference of the high school graduates for the Republicans.

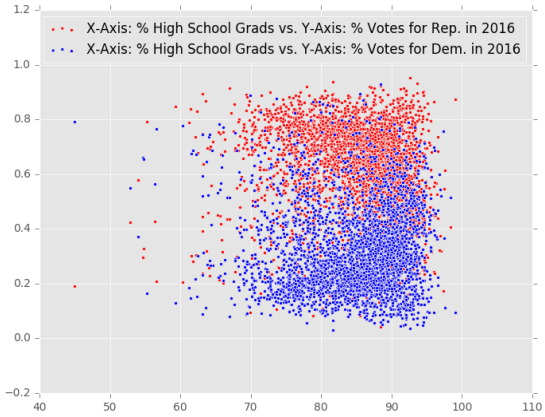

Figure 3g: Graph depicting effect of percent of high school graduates on each party votes

Table 3g: Correlation values for high school graduates

Edu_highschool Vs Republicans	-0.0901192732634
Edu_highschool Vs Democrats	0.00704367550301

% Bachelor's Degree Graduate

Higher the percent of the persons that are bachelor degree graduates in the population of the county, higher was the vote share of the Democratic party, in this case the votes for Clinton. This shows the preference of the bachelor's degree graduates for the Republicans.

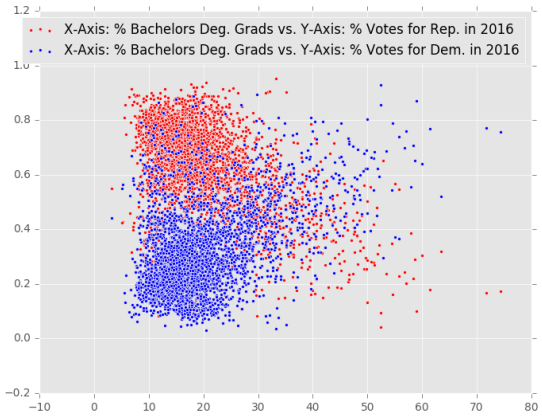

Figure 3h: Graph depicting effect of percent of bachelor's degree graduates on each party votes

Table 3h: Correlation values for Bachelor's degree graduates

Edu_batchelors Vs Republicans	-0.487248749918
Edu_batchelors Vs Democrats	0.43422423644

% Female Persons

Higher the percent of the females in the population of the county, the vote share is tilted towards that of the Republican party, in this case the votes for Trump. As a trend, this graph shows that the female population was supportive of the Republican party.

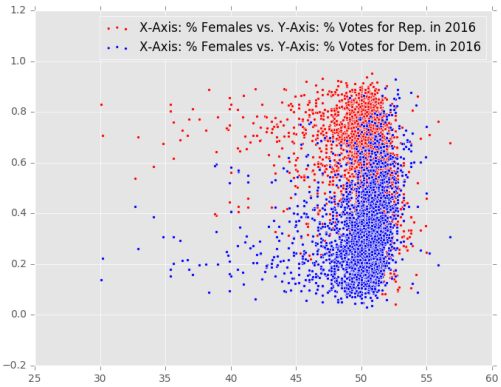

Figure 3i: Graph depicting effect of percent of female persons on each party votes

Table 3i: Correlation values for female persons

Females Vs Republicans	-0.169973518408
Females Vs Democrats	0.188291194133

Number Of Veterans

A Veteran is a person who has served in the military. Higher the number of the veterans of the population in the county, higher was the vote share of the Democratic party, in this case the votes for Clinton. This shows the preference of the veterans for the Democrats.

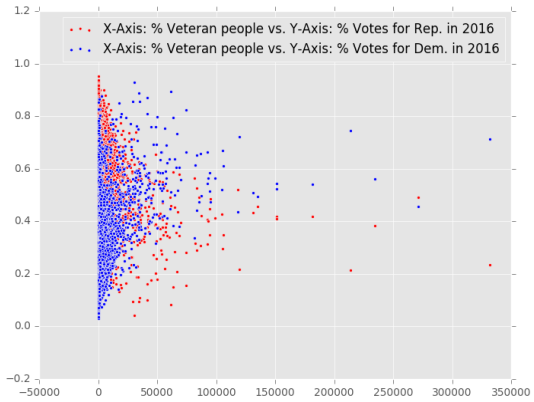

Figure 3j: Graph depicting effect of number of veterans on each party votes

Table 3j: Correlation values for the Veterans

Veteran people Vs Republicans	-0.371746352113
Veteran people Vs Democrats	0.370795706716

Median Household Income

This feature involved is the median household income calculated over the last four years. We can deduce from the graph, that as this income slowly increases, the vote share of the Democratic party increases, in this case the votes for Clinton. This shows the preference of the rich for the Democrats.

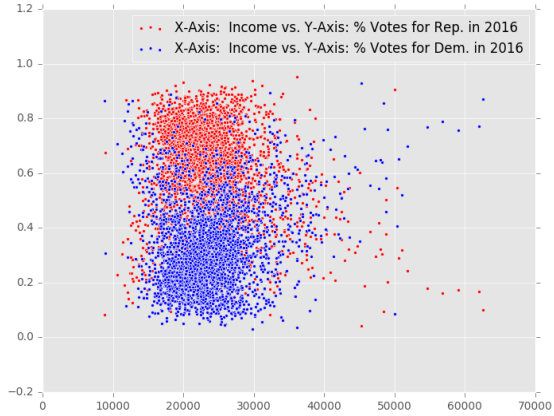

Figure 3k: Graph depicting effect of median household income on each party votes

Table 3k: Correlation values for median household income

Income Vs Republicans	-0.236665738023
Income Vs Democrats	0.197489494911

% Persons Below Poverty Level

This feature puts the statistics of the people below the poverty level in a county, i.e. the poor and unprivileged persons. The graph here seems to indicate that as this percent has increased, the vote share for the Democratic party has increased too. In this case, the counties where the percent of persons below the poverty level was really high, the number of votes Clinton garnered was higher.

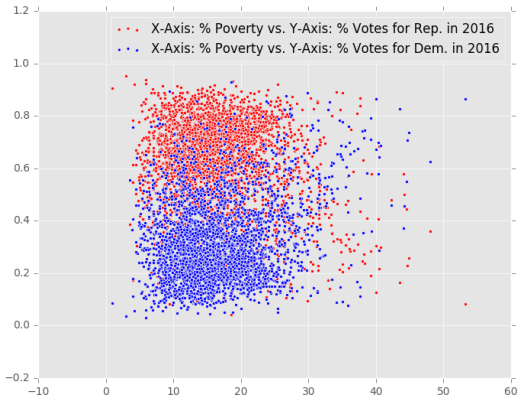

Figure 3l: Graph depicting effect of percent of persons below poverty level on each party votes

Table 3l: Correlation values for persons below poverty level

Poverty Vs Republicans	-0.142598076832
Poverty Vs Democrats	0.200742382063

Votes for the Republicans in the 2012 election

We have also considered the past, i.e. the number of votes for the Republicans in the last election to understand correlation. In the last election, Mitt Romney was the presidential candidate for the Republican party.

Votes for Democrats in the 2012 election

In the last election, Barack Obama was the presidential candidate for the Democratic party. He went on the win the election and was elected the President of the United States.

Classification Algorithms

K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm is a non parametric lazy learning algorithm. A non parametric learning algorithm is one that does not make assumptions on the underlying data distribution. A lazy algorithm in the learning sense means it does not use the training data points to perform any generalization. There is no explicit training phase or it is very minimal.[6]

In KNN classification, the output is a class membership. A test entry is classified by a majority vote of its neighbors, with it being assigned to the most common class among its k nearest neighbors. If for instance, k=1, then the entry to be labeled is simply assigned to the class of its nearest neighbour. The training examples are vectors in a multidimensional feature space, each with a class label. The two phases of the algorithm are:

Training Phase
This consists only of storing the features as vectors and the class labels of each of the training samples.

Classification/ Testing Phase
In this phase, we use a value k, as a user defined constant as well as a test point that must be labelled by assigning the class that is most frequent among the k training samples nearest to the test point.

In our implementation of this algorithm, we have used the Euclidean distance between points as the distance metric. Euclidean distance is the straight line distance between two points. In an n dimensional space, the general formula for Euclidean distance is

$$d(x,y) = \sqrt{(y_1-x_1)^2 + (y_2-x_2)^2 + \ldots + (y_n-x_n)^2}$$

General formula for Euclidean Distance

Choosing K
The value of K is non-parametric and the general rule of thumb for choosing K is K = sqrt(N)/2, where N stands for the number of training samples in the dataset.

Since the number of training samples in our dataset is 2490, we chose a value of 25 for K. This was helpful in increasing the accuracy, since the value is odd it ensures there is no tie between choosing a class while majority voting.

Classify (X,Y,x) // X: training data, Y: class labels
X: unknown sample
for i = 1 to m **do**
  Compute Euclidean distance $d(X_i,x)$
end for
Compute set I containing indices for the k smallest distances $d(X_i,x)$
return majority label for $\{Y_i$ where $i \in I\}$

Pseudo code for K-Nearest Neighbors [7]

Decision Trees

A decision tree represents a function that takes as input a vector of attribute values and returns a decision – a single

output value. A decision tree reaches its decision by performing a sequence of tests. Each internal node in the tree corresponds to a test of the value of one of the input attributes, and the branches from the node are labelled with the possible values of the attribute. Each leaf node in the tree represents a value to be returned by the function.

function *DecisionTreeLearning*(*examples, attributes, parentexamples*)
if *examples* is empty **then return** *Plurality − Value(parent_examples)*
else if all *examples* have same classification *c* then
return c
else if attributes is empty then **return** *Plurality − Value(examples)*
else
A ← $argmax_{a} \in attributesImportance(a, examples)$
tree ← a new decision tree with root test A
for *vk* ∈ *A* **do**
exs ← *{e : e* ∈ *examples* ∧ *e.A = vk}*
subtree ← *DecisionTreeLearning(exs, attributes − A, examples)*
add a branch to *tree* with label (A = *vk*) and subtree *subtree*
end for
end if
end function

Pseudo code for Decision Tree Learning algorithm [1]

Decision Trees are very straightforward to read, even by someone without a machine learning background. The algorithm constructs the tree using a greedy approach, we split on the feature that currently best separates the data. The best feature to be split on is determined using different metrics such as:

Information Gain
The measure of purity is called information. It represents the expected amount of information that would be needed to specify whether a new instance should be classified as either of the categorical class labels. Entropy on the other hand is the measure of impurity. For a binary class where the class labels are a and b, the formula for entropy is

$$Entropy = - P(a) * \log(P(a)) – P(b)*\log(P(b))$$

The information gain is the difference between the entropy values before and after the split at the node i.e. it is the amount of information we gained by doing the split using the feature.

Information gain = Entropy before split- Entropy after split

Gini Impurity
The Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labelled if it was randomly labeled according to the distribution of labels in the subset. It reaches its minimum when all cases at the node are classified as a single class or target category.

Some specific decision tree algorithms are:

- ID3 (Iterative Dichotomiser 3)
- C 4.5
- CART (Classification and Regression Tree)

In our implementation of decision trees, we have used the CART algorithm. Since the classification task at hand is a binary classification, we use the Gini coefficient or Gini index.

The problem with building decision trees is overfitting. One way that this problem can be solved is by pruning the fully grown tree according to some measure of how good our tree perform on training data and how complex the nodes of the tree are. Minimal Cost-Complexity Pruning uses the measure of Cost- Complexity. [8]

$$R\alpha(T) = R(T) + \alpha|T|$$

Where R(T) is the Resubstitution Error and accounts for our performance in the training data, |T| is how much nodes our tree has and α is the complexity parameter. [8]

In our implementation, the best feature to be split, that separated the data in best way was the number of votes for Democrats in 2012. It had a Gini index of 0.2644.

Random Forests

The Random Forests algorithm is an ensemble learning algorithm. Ensemble learning methods are methods that generate many classifiers and aggregate the results obtained from them The two well known methods are bagging and boosting.

Bagging (Bootstrap Aggregating) creates bootstrap samples of a training set using sampling with replacement. Each sample is then used to train a different component of a base classifier. Finally, the classification is completed by plurality voting. Not only does this approach help to overcome overfitting in decision trees, it also works for neural networks.

In the Random Forest algorithm, we create various trees using random samples of the original training set. Next, instead of choosing the feature that gives the maximum

information gain, we choose a random subset of features and move forward.

In our implementation of Random Forests, we ran the algorithm with 10 trees and achieved a higher accuracy than the Decision Tree algorithm.

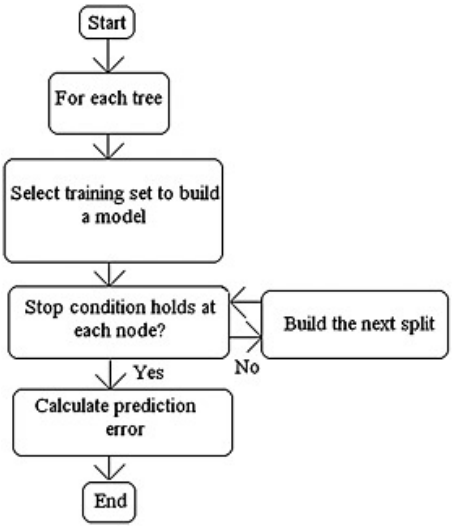

Figure 4: Flow chart for Random Forests algorithm

Naïve Bayes

This type of classifier works by applying the Bayes' theorem with strong independence assumptions between the features.

Bayes' Theorem
The Bayes' theorem can be stated as follows:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Where A and B are events and P(B) is not 0

- P(A) and P(B) are probabilities that A and B occur independently
- P(A|B) and P(B|A) are conditional probabilities

The Naïve Bayes Classifier works on the concept of combining prior probabilities (based on prior experience) as well as the likelihood (conditional probability) to form a posterior proabability. To estimate these parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set.

$$P(\mathsf{Y}, \mathsf{F}_1 \ldots \mathsf{F}_n) = P(\mathsf{Y}) \prod_i P(\mathsf{F}_i|\mathsf{Y})$$

Gaussian Naïve Bayes
While dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Multinomial Naïve Bayes
With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial ($p_1,\ldots,p_n$) where p_i is the probability that event i occurs. It is used for discrete counts.

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Bernoulli Naïve Bayes
In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. It is useful if the features involved are binary (zeros and ones). [12]

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma 2}}$$

Out-of-core Naïve Bayes model fitting
Naive Bayes models can be used to tackle large scale classification problems for which the full training set might not fit in memory. To handle this MultinomialNB, BernoulliNB, and GaussianNB expose a partial_fit method that can be used incrementally as done with other classifiers as demonstrated in Out-of-core classification of text documents. All naive Bayes classifiers support sample weighting. Contrary to the fit method, the first call to partial_fit needs to be passed the list of all the expected class labels. [15]

Applications

- Text classification
- Sentiment Analysis
- Recommendation System

In our implementation of the Naïve Bayes, we implemented all the three types and achieved the highest accuracy with the Gaussian Naïve Bayes.

Neural Networks : Multilayer Perceptron

The word network in the term 'artificial neural network' refers to the interconnections between the neurons in the different layers of each system. An artificial neural network is defined by three types of parameters:

- The interconnection pattern between different layers of neurons
- The learning process for updating weights of the interconnections
- The activation function that converts a neuron's weighted input to its output activation

A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. A multilayer perceptron consists of multiple layers of nodes in a graph, with each layer fully connected to the next one. There are three distinct layers namely the input, hidden and output layers. Between the input and output layers, there can one or more non-linear layers called hidden layers.

Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation followed by a non-linear activation function. The multiplayer perceptron classifier trains its model using Backpropagation.

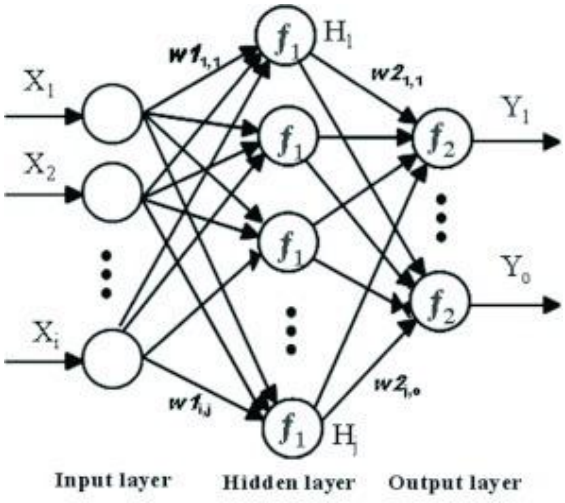

Figure 5: A perceptron with one hidden layer

Backpropagation

The backpropagation algorithm repeats a two phase cycle, propagation and weight update.

When an input vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the desired output, using a loss function, and an error value is calculated for each of the neurons in the output layer. The error values are then propagated backwards, starting from the output, until each neuron has an associated error value which roughly represents its contribution to the original output. [11]

Backpropagation uses these error values to calculate the gradient of the loss function with respect to the weights in the network. In the second phase, this gradient is fed to the optimization method, which in turn uses it to update the weights, in an attempt to minimize the loss function.

In our implementation of this neural network, the number of hidden layers are 100 and α is 0.0001.

Support Vector Machine

The Support Vector Machine (SVM) supervised learning algorithm creates a model that is a representation of the training examples as points in space, mapped in such a way that the examples classified into different categorized are divided by a clear gap that is as wide as possible.

SVMs find the separator with the maximum margin between the two classes. To understand SVMs let us understand the Margin Infused Relaxed Algorithm (MIRA) first.

MIRA

There are some problems with the perceptron learning algorithm:

- Noise: if the data is not separable, the weights might thrash
- It finds a "barely" separating solution
- Overtraining

One way to overcome these effects is to adjust the weight update. MIRA chooses an update size that fixes the current mistake but minimizes the change to the weights. [10]

SVMs are like the MIRA algorithm, in this case you optimize over all examples at once. Basically, a support vector machine constructs a hyperplane or a set of hyperplanes in a high dimensional space, which can be used for tasks like classification, regression. A good classification is achieved by the hyperplane that maximizes the margin, i.e. the hyperplane that has the largest distance to the nearest training data point of any class. This obviously means that larger the margin, lower the classification error of the model classifier.

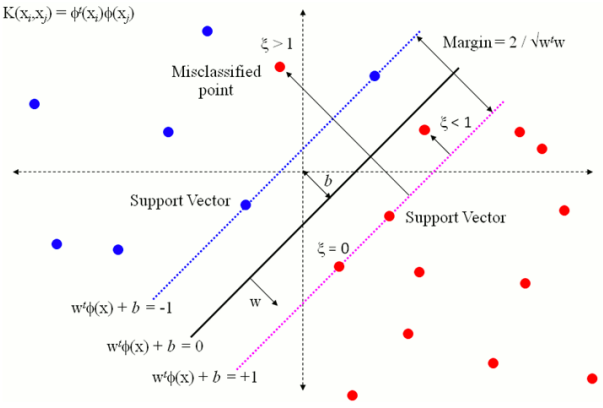

Figure 6a: Support Vectors and maximizing the margin

Kernel Trick

SVMs can also efficiently perform non linear classification using the kernel trick, which maps the inputs into high dimensional feature spaces.

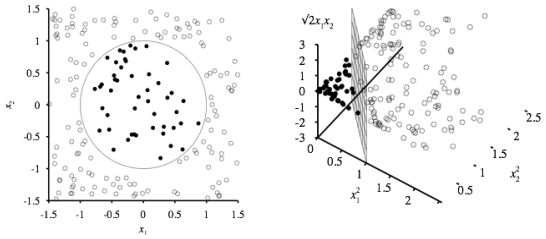

Figure 6b: Mapping into higher dimensional feature spaces

In our implementation, we have used the Support vector clustering (SVC) extension of the SVM. It is a method that builds on kernel function and is also appropriate for unsupervised learning and data mining.

With the SVC classifier, we have used the polynomial kernel function that represents the similarity of the training samples in a feature space over polynomials of the original variables, this allowing learning of non-linear models too.

The training data was scaled, and this method of preprocessing the data improved the accuracy of the SVM algorithm from 93% to 95%.

Interpretation And Impact Of Feature Selection

To test the accuracy of prediction using each classifier as well as understand the importance of selecting the right features, we used three feature sets and passed each to all the classification algorithms.

F1: Persons 65 years and over (percent)
Persons below 5 years (percent)
Persons below 18 years (percent)
Number of veterans in the last four years

F2: Persons 65 years and over (percent)
Female persons (percent)
White persons (percent)
Black persons (percent)
Hispanic/Latino persons (percent)
High school degree or higher (percent)
Bachelor's degree or higher (percent)

F3: Selected feature set

We chose the above feature sets in the increasing order of relevance and possible correlation to the eventual winning party in the county.

The first set consists of features like persons below 5 years and persons below 18 years of age, which is not closely related or important to the label to be predicted.

Moving to the second set, we considered the important factor of race that has definitely played a significant role in the elections this year. The preference of the factions from various races, ethnicities, color and religion was helpful in improving the accuracy of prediction of the eventual winner.

The last set is the selected feature set. The factors like economic standing as well as the winning party in the previous election in the county, added on in this set improved the accuracy of classification even further. This feature set covers all the relevant features that have been influential in determining the election results:

- Race
- Education
- Economic standing
- Previous election results (stronghold in state)

Evidently, feature set 3 shows the highest accuracy in predicting the winning political party in all of the classification algorithms. This highlights the importance of choosing the right features for a machine learning classification task.

Results

The accuracy achieved for each of the feature sets on the classification algorithms is an average accuracy over a number of implementations of the algorithm. The results are in the form and percentages i.e. the ratio of the number of tuples (counties) correctly classified to the total number of tuples.

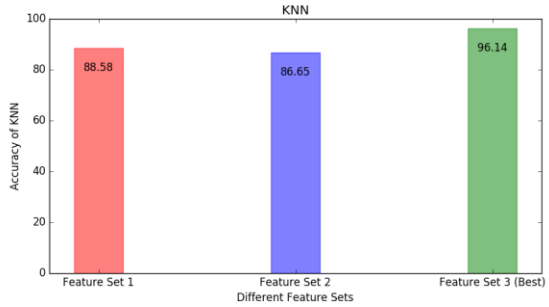

Figure 7a: Classification accuracy for K Nearest Neighbor

Feature Set	Max Accuracy
F1	88.58%
F2	86.65%
F3	96.14%

Figure 7b: Classification accuracy for Decision Tree

Feature Set	Max Accuracy
F1	85.55%
F2	90.69%
F3	94.38%

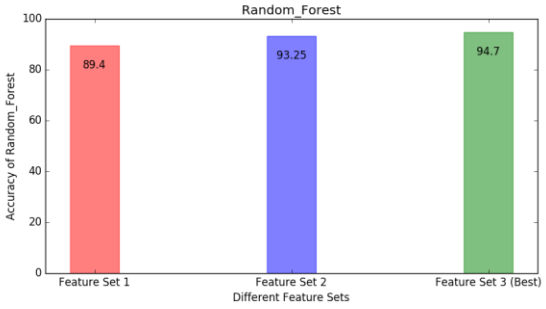

Figure 7c: Classification accuracy for Random Forest

Feature Set	Max Accuracy
F1	89.40%
F2	93.25%
F3	94.70%

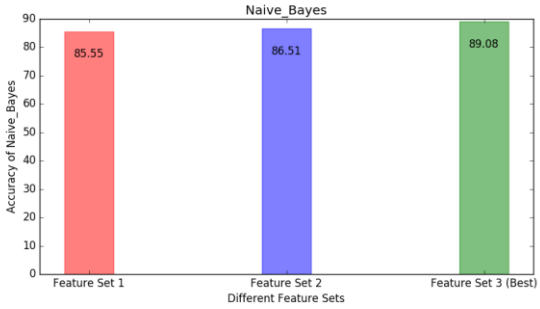

Figure 7d: Classification accuracy for Naïve Bayes

Feature Set	Max Accuracy
F1	80.41%
F2	86.51%
F3	95.34%

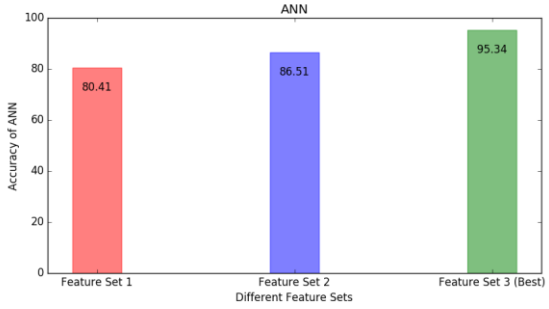

Figure 7e: Classification accuracy for the Multi Layer Perceptron Artificial Neural Network

Feature Set	Max Accuracy
F1	80.41%
F2	86.51%
F3	95.34%

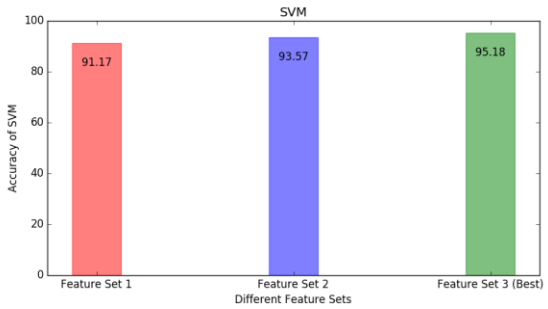

Figure 7f: Classification accuracy for Support Vector Machine

Feature Set	Max Accuracy
F1	91.17%
F2	93.57%
F3	95.18%

Conclusion

There has been a lot of debate and discussion about the eventual results of the US Presidential Elections in 2016. This paper gives us an idea of the perspective of the people, the voters themselves. Through analysis of the features, we have observed the factors that weighed heavily in determining the winning political party. Issues such as racism, discrimination, abortion and their effects can be seen through the relation between each of the related features and the votes won by the winning candidate.

There were some interesting trends noticed through the voting and population statistics. It seemed though the higher educated persons in the county preferred that the candidate from the Democratic party becomes the President. Another interesting take away was the preference of females towards the Republicans.

The highest accuracy of prediction amongst all the classification algorithms is shown by the K Nearest Neighbor algorithm followed by the Artificial Neural Network – Multilayer Perceptron and the renowned Support Vector Machine algorithm.

Future Work

The results of the classification can be improved by understanding and including more factors that clearly discriminate the lines of thoughts of the two parties such as issues like LGBTQ rights, gun control. It could be helpful to study the history of the candidate involved, which would improve the prediction. For instance, a Presidential candidate or a party that has been accused of corruption in the past, would be expected to win few votes in the elections. This would help constructing a stronger and more relevant feature set for prediction.

This paper covers some of the supervised learning techniques for classification. These results can be compared to the accuracies of implementation with unsupervised and reinforcement learning techniques as well.

Acknowledgements

We would like to thank professor Christopher Amato and the teaching assistant Matt Corsaro for guiding us with this project. We would also like to thank Kaggle for making this dataset available and for encouraging students to take up Machine Learning.

References

[1] Russell, S., Norvig, P., & Intelligence, A. (2010). A modern approach. Artificial Intelligence. Prentice-Hall, Englewood Cliffs, 25, 27.

[2] Presidential Election Process https://www.usa.gov/election

[3] Kaggle 2012 and 2016 Presidential Elections https://www.kaggle.com/joelwilson/2012-2016-presidential-elections

[4] Brownlee, J. How To Prepare Your Data For Machine Learning in Python http://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/

[5] Brownlee, J. Feature Selection for Machine Learning in Pythonhttp://machinelearningmastery.com/feature-selection-machine-learning-python/

[6] Thiruuruganathan, S. https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/

[7]https://www.researchgate.net/figure/260397165_fig7_Pseudocode-for-KNN-classification

[8] Ota Jun, O. Diagnosing Breast Cancer using Random Forests

[9] Liaw, A and Weiner, M. Classification and Regression by randomForest Vol.2/3, December 2002

[10] Amato, C Northeastern University CS5100 Images and Slides

[11] Backpropagation https://www.wikipedia.org/

Backpropagation

[12] Lydakis, A. Impact Of Feature Selection in a Classification Task

[13] Mazariegos Carpio, E. Predicting Soccer Matches Using Machine Learning

[14] Sampath,R. Teng, Y. Classification and Regression Approaches to Predicting United States Senate Elections

[15] Naïve Bayes http://scikit-learn.org/stable/modules /naive_bayes.html