PSTAT 131 Final Project

Lauren Wong (9641309) and Alison Do (9482126)

December 12, 2018

Background

1. What makes voter behavior prediction (and thus election forecasting) a hard problem?

Voter behavior prediction (and thus election forecasting) can be a hard problem because there are numerous random variables that need to be accounted for in terms of what a voter will decide/say. A lot of the times, these variables are immeasurable since they rely on the voter's own background and mindset, which is not always easy to predict considering the constant change in state and national economy, or changes in societal conditions, etc. Some of those voter behavior variables can include the demographic or gender of a voter, or the circumstances of the voter's state of living such as income, family, etc. which all may lead to change in intents over time. Ultimately, there are many factors that need to be considered, but not all of them are easily predicted, leading to errors that can have drastic changes in the overall election forecasting.

2. What was unique to Nate Silver's approach in 2012 that allowed him to achieve good predictions?

Nate Silver's methodology in 2012 was unique because his forecast focused on what other people didn't look at: decisiveness in the public. Silver's method consisted of using Bayes' Theorem and graph theory in order to calculate the probability of the support percentage being over 50% for each individual state on each day. There was also an assumption that polling errors are correlated, and polls in other states can miss in the same direction. He would then use his model on the following day's results to calculate the probability of the support percentages shifting by a certain amount. This approach allowed him to gather thorough data daily to increase the accuracy of his predictions.

3. What went wrong in 2016? What do you think should be done to make future predictions better?

There could be a variety of reasons as to why the predictions were not accurate in 2016, the biggest being that voter behavior is incredibly diverse. For example, in this specific election, many may have been unwilling to voice who they were planning to vote for. As a result, the polls based on voting behavior may have predicted the final outcome incorrectly. In addition, it is difficult for journalists and the media to keep updated with the latest prediction models, considering how much the voting data must have changed on the daily. In the future, predictions can be made better by considering more voter behavior features such as voting late, or being undecided. Moreover, systematic polling errors can be evaluated more deeply and fixed, incorrectly collected/inaccurate reports on voter behaviors can be updated, and so forth.

Data

Election data

4. Report the dimension of `election.raw` after removing rows with `fips=2000`. Provide a reason for excluding them.

The reason why we removed the rows with fips = 2000, is because the counties had NA values, which have the potential to hinder future calculations in our project. The dimension of election.raw after removing rows with fips = 2000 is 18345 x 5.

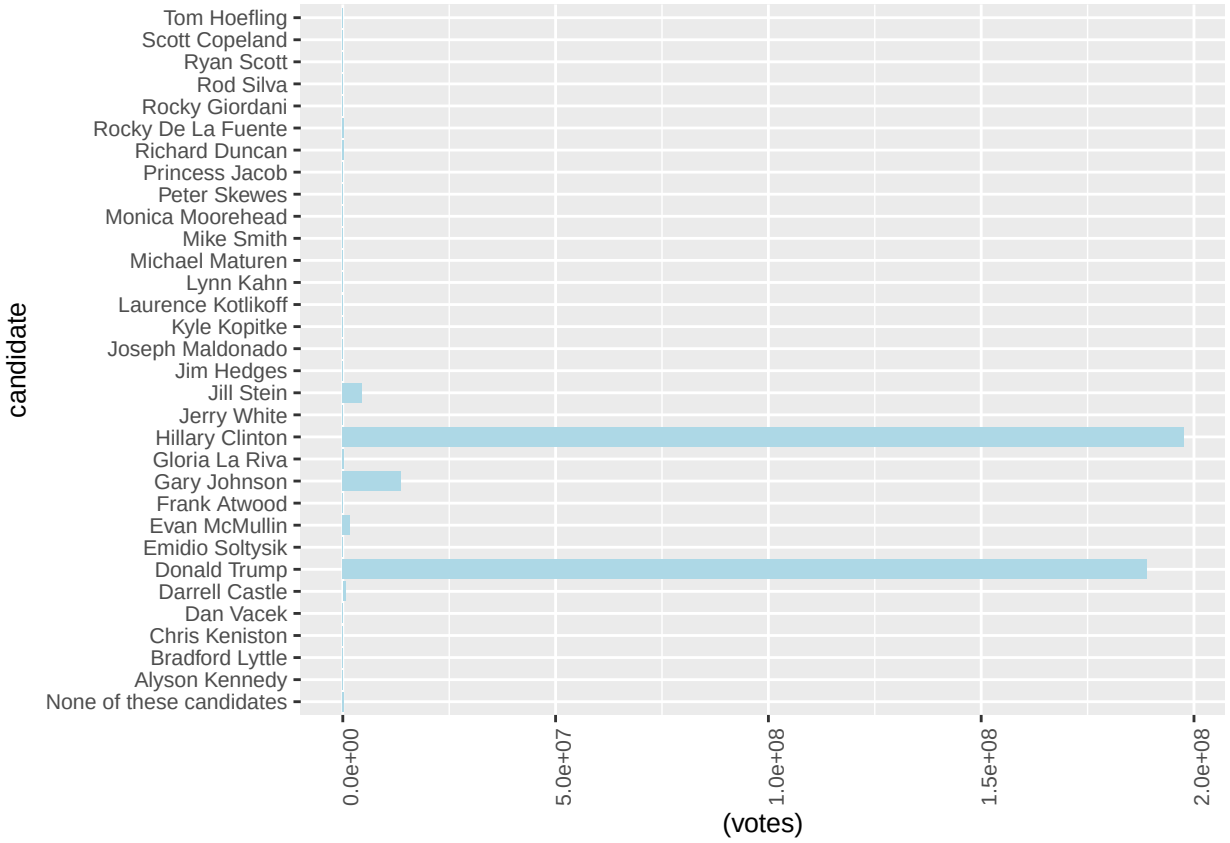
Census data

Data wrangling

5. Remove summary rows from `election.raw` data: i.e.,

Election Federal data has 32 observations of 5 variables. Election State data has 298 observations of 5 variables. Election Data has 18,007 observations of 5 variables.

6. How many named presidential candidates were there in the 2016 election? Draw a bar chart of all votes received by each candidate.

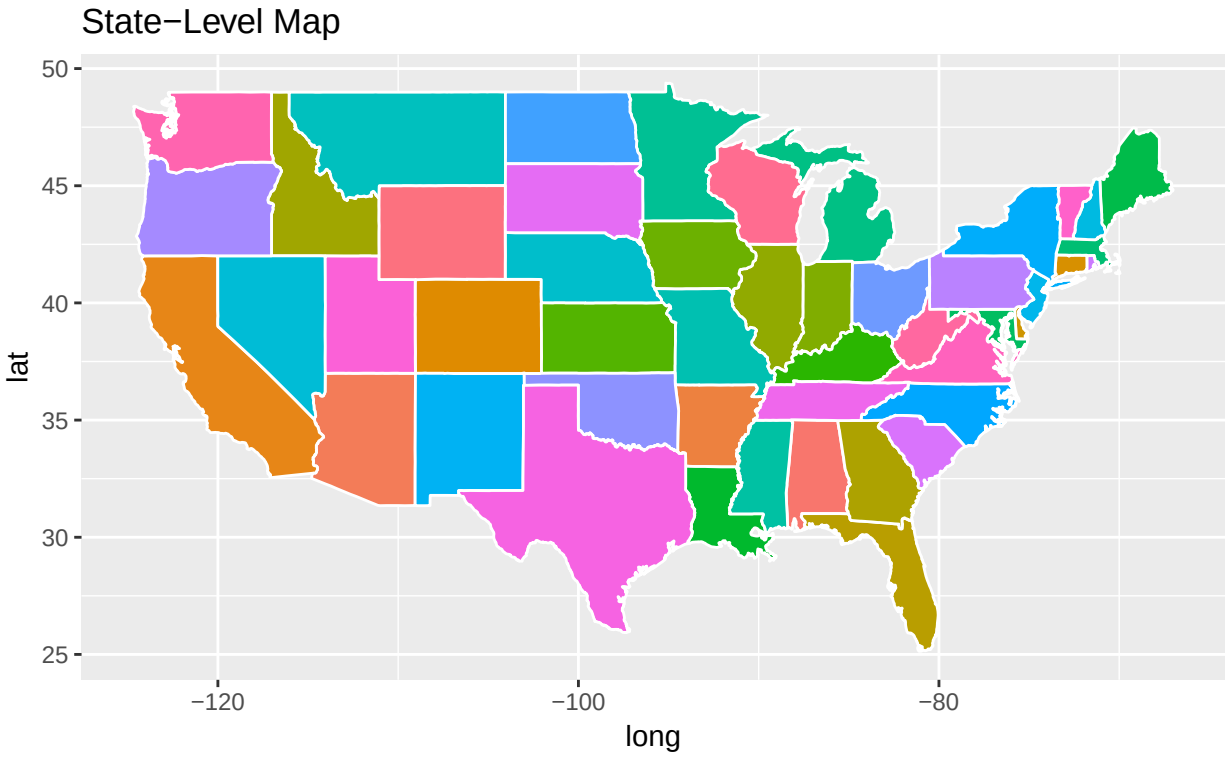


There are 32 named presidential candidates in the 2016 election.

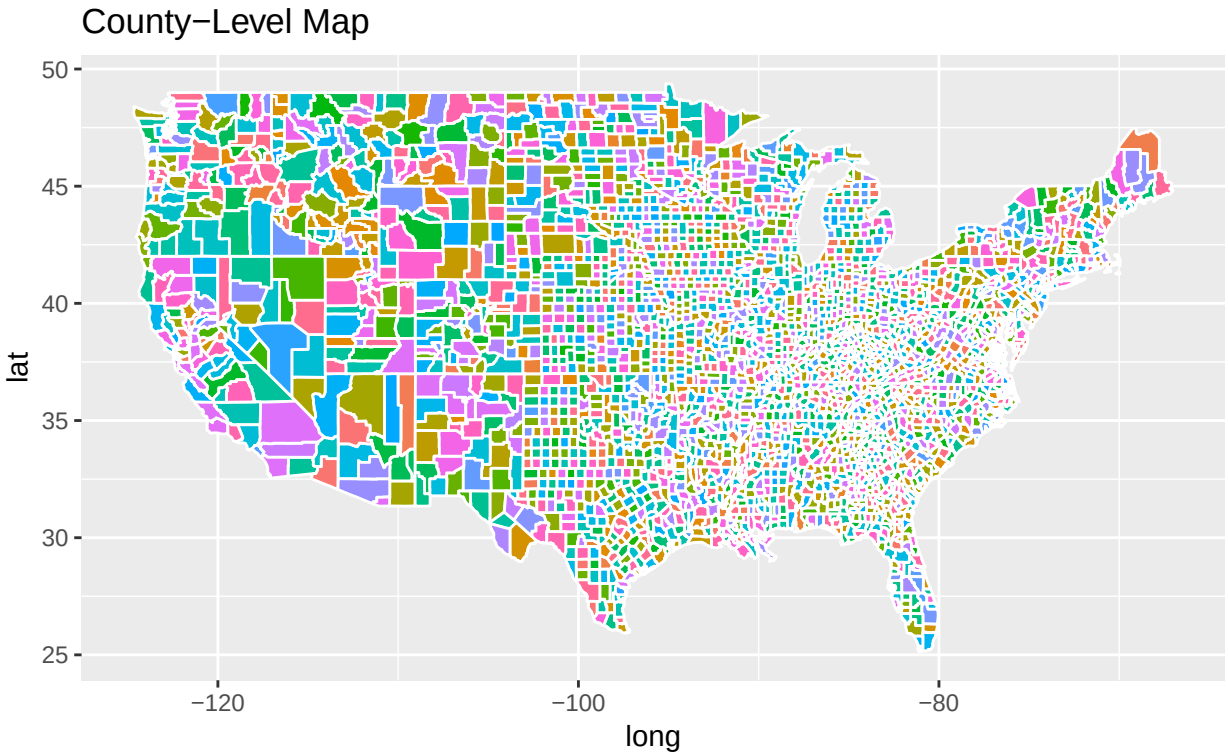
Visualization

7. Create variables `county_winner` and `state_winner` by taking the candidate with the highest proportion of votes.

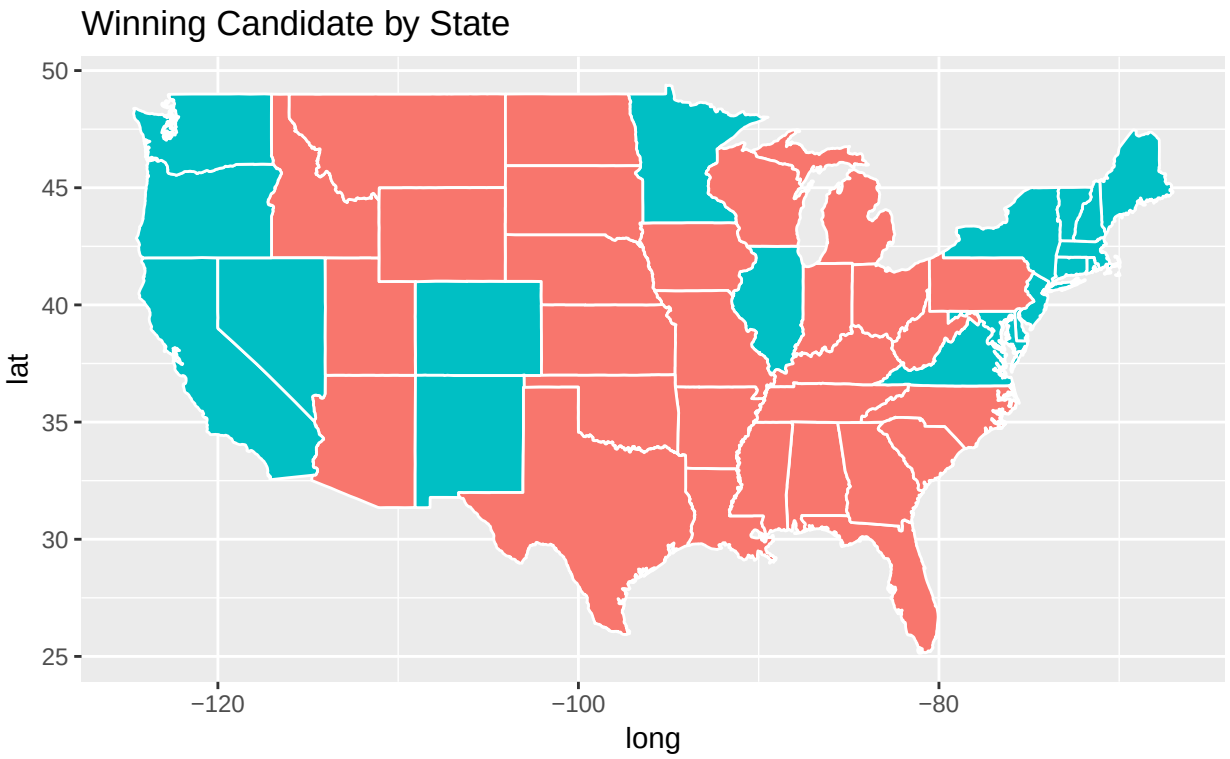
County winner data has 3,111 observations of 7 variables and State Winner has 50 observations of 7 variables.



8. **Draw county-level map and color by county**

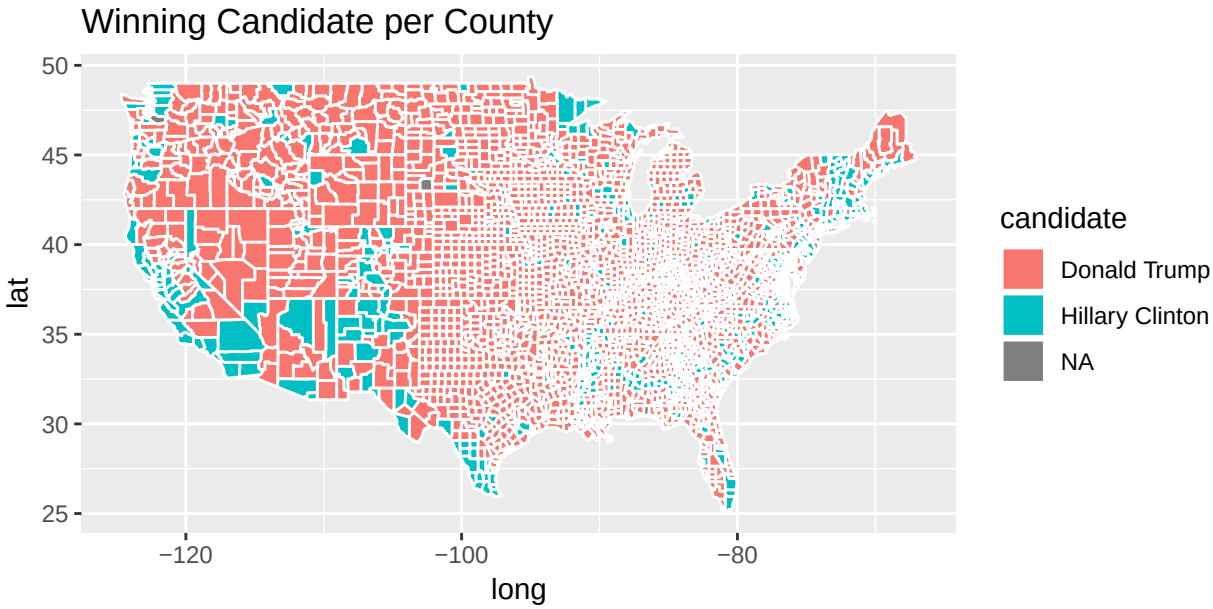


9. **Now color the map by the winning candidate for each state.**

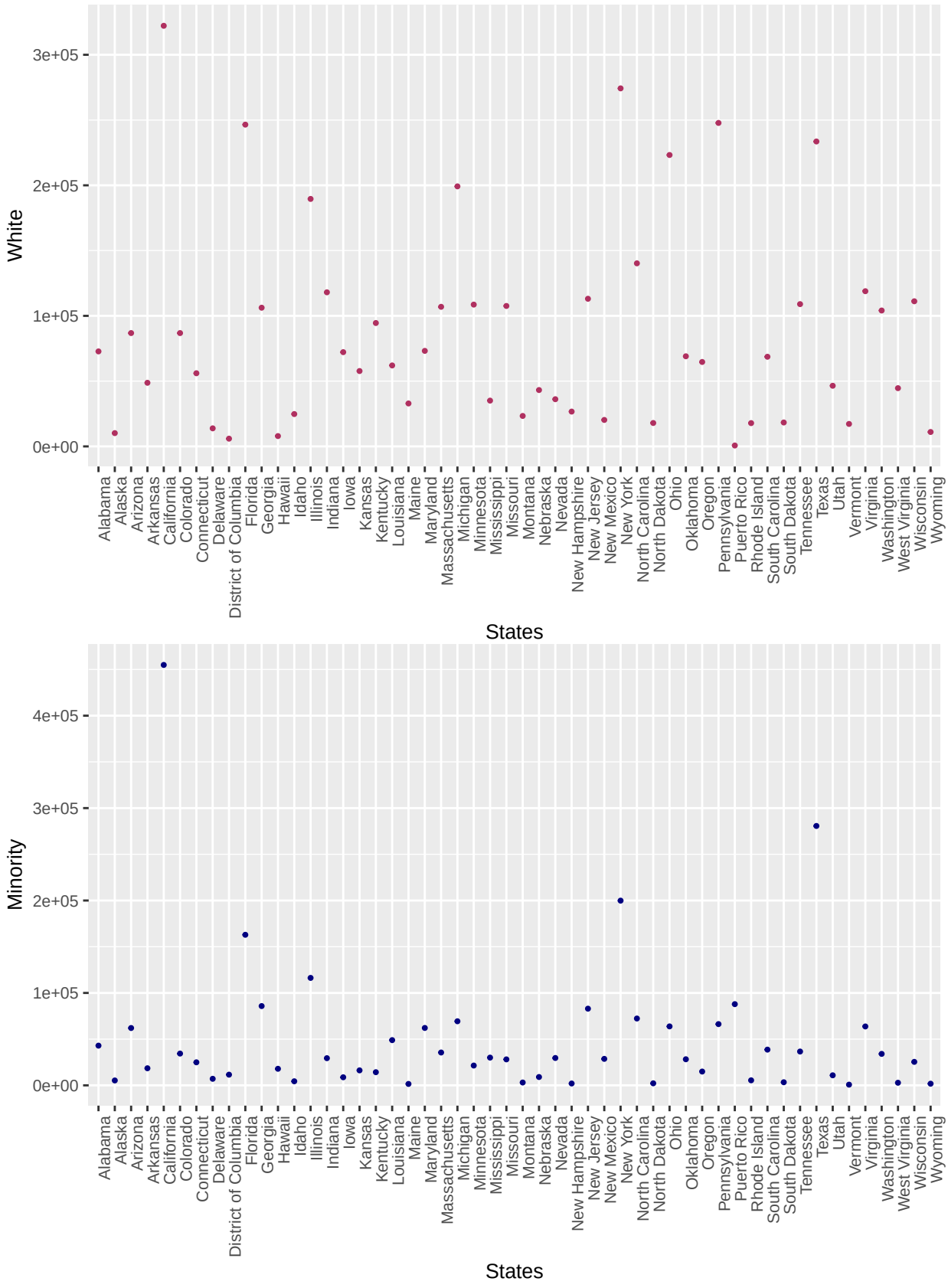


The red in the map represents Donald Trump's wins per state and the blue in the map represents Hillary Clinton's wins per state.

10. The variable `county` does not have `fips` column. So we will create one by pooling information from `maps::county.fips`.



11. Create a visualization of your choice using census data.



For our visualization we decided to plot population of Minorities versus Whites in each State who voted in

the election. This shows that the demographics of a voter can affect the amount of votes candidates receive. From this, we can see that minorities are less likely to vote, possibly due to less opportunities, or from discouragement of being involved in the white-dominant political environment.

12. In this problem, we aggregate the information into county-level data by computing `TotalPop`-weighted average of each attributes for each county. Create the following variables:

Census.del data has 72,720 observations of 28 variables. Census.subct data has 72,720 observations of 30 variables. Census.ct data has 3,218 observations of 28 variables.

Dimensionality reduction

13. Run PCA for both county & sub-county level data.

It is important to standardize the variables to have mean zero and standard deviation one before performing PCA. If we failed to scale the variables before performing PCA, then most of the principal components that we observed would be driven by a weighted variable that has the largest mean and variance. Thus, rendering it impossible to scale the other variables evenly.

What are the three features with the largest absolute values of the first principal component? Which features have opposite signs and what does that mean about the correaltion between these features?

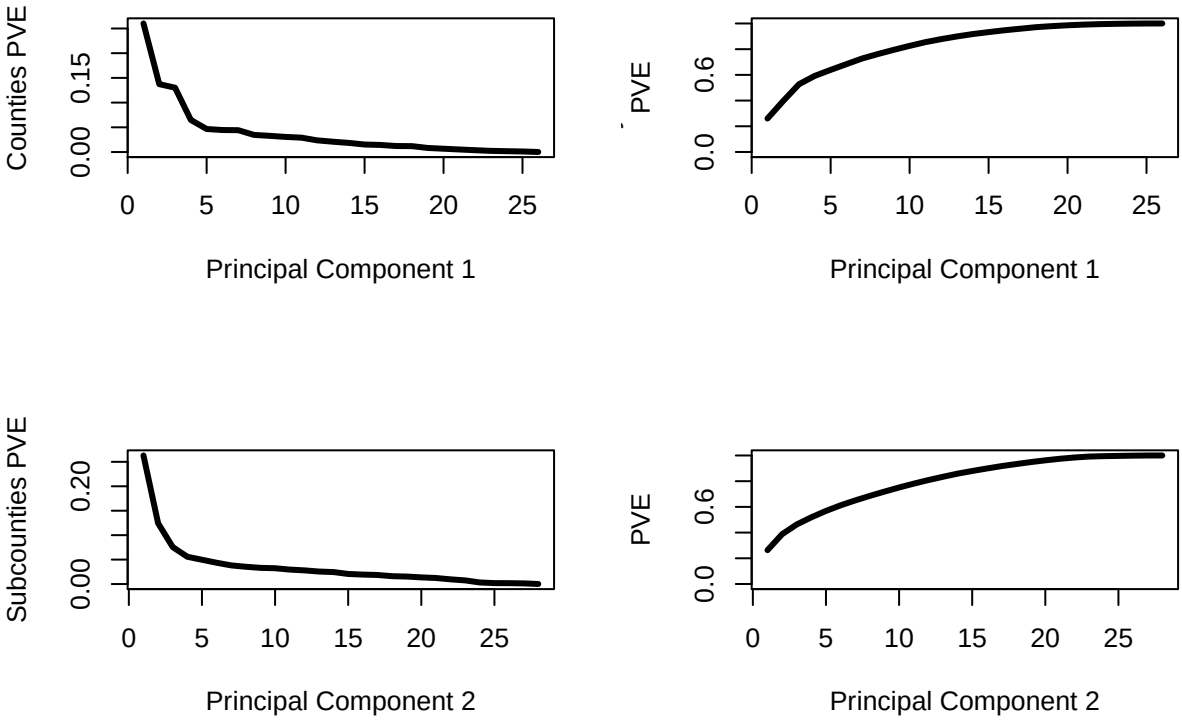
For county-level PCA data, the three largest absolute values of PC1 are IncomePerCap, ChildPoverty, and Poverty. For sub-county level PCA data, the three largest absolute values of PC1 are IncomePerCap, Professional, and Poverty. Overall for both county- and sub-county levels, we can observe that a person/family's employment status and amount of wealth weigh more heavily on their decision to vote for a specific presidential candidate, most likely someone who would help them benefit in future job/wealth opportunities.

Looking at the signs of the features, the positive and negative signs refer to whether or not the features have a positive or negative correlation with one another within the Principal Component. For example, for county-level PC1 data, Income has a value of 0.318634063 whereas Poverty has a value of -0.338292096. The opposite signs imply that Income and Poverty have a negative correlation; if Income increases, Poverty decreases and vice versa. On the other hand, for sub-county level PC1 data, we can observe that Carpool and Transit are both positive values compared to Drive, which has a negative value; this implies that if the amount of people who drive decreases, then they have most likely began to use transit more or carpool with one another more. Transit and Carpool both increase together (positive correlation) since they have the same sign.

14. Determine the number of minimum number of PCs needed to capture 90% of the variance for both the county and sub-county analyses.

14 is the minimum number of PCs needed to capture 90% of the variance for the county.

17 is the minimum number of PCs needed to capture 90% of the variance for the subcounty.



Clustering

15. With `census.ct`, perform hierarchical clustering with complete linkage.

Compare and contrast the results. For both approaches investigate the cluster that contains San Mateo County. Which approach seemed to put San Mateo County in a more appropriate clusters? Comment on what you observe and discuss possible explanations for these observations.

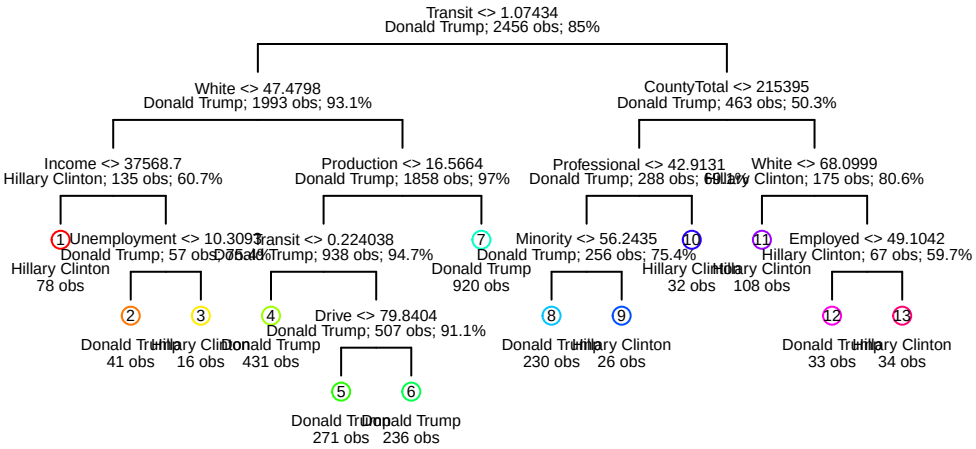
We noticed that before using PCA, clustering decreases form 2584 to 13 in the first 5 clusters then proceeds to decrease to 1, increase to 14, and decrease to 11. Reclustering with PCA, however, shows a trend of decreasing in the first 3 clusters to increasing from cluster 4 to 5, and then decreasing from cluster 6 to 8, and increasing until the 10th.

Analyzing these trends leads us to believe that San Mateo is placed into Clust 2, group 9. It seems that the complete linkage cluster, Clust 2, is more appropriate because a complete link is less susceptible to noise and outliers while the other method is sensitive to them. San Mateo belongs more in Group 9 than in Group 8. Furthermore, a smaller distance from the mean resembles a more appropriate cluster and contain less variance than variables further away from each other.

Classification

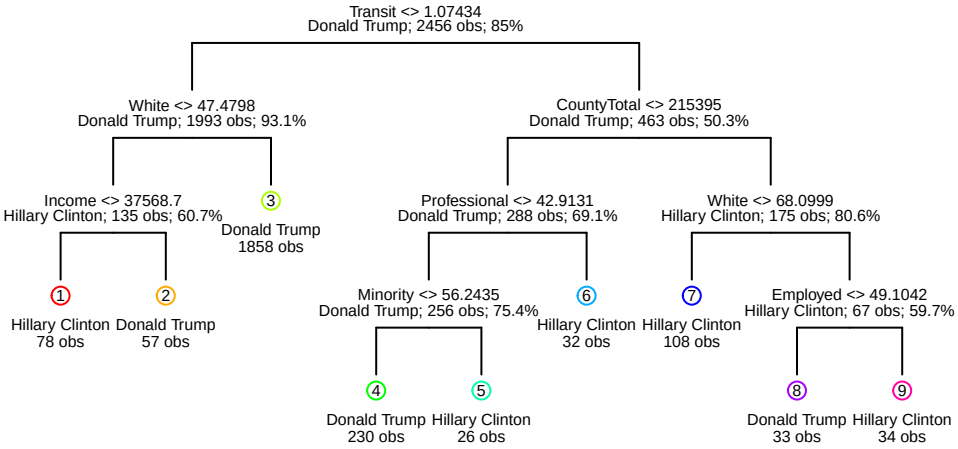
16. Decision tree: train a decision tree by `cv.tree()`.

Unpruned Tree



Total classified correct = 94.1 %

Pruned Tree



Total classified correct = 93.9 %

	train.error	test.error
tree	0.0610749	0.0846906
logistic	NA	NA
lasso	NA	NA

Intepret and discuss the results of the decision tree analysis. Use this plot to tell a story about voting behavior in the US.

As can be seen, our unpruned decision tree received a total of 94.1% classification success, which is great for a data set that is as large as ours. Looking in depth, we can see that for people who tend to use transit less, they are more likely to vote for Donald Trump if they're white and have a medium to high income. On the contrary, Hillary Clinton was popular amongst counties, minorities, and people who had a low to medium income or are unemployed.

Now looking at our pruned decision tree, we can see that it received a total of 93.9% classification success, which is 0.2% less than our unpruned tree. A reason for this may be because pruning tends to decrease the

number of variables the decision tree contains, which may hinder the accuracy for bigger data sets such as ours. However, this 0.2% difference is still only a small difference, and therefore may be negligible. The pruned decision tree overall provides a more clean visualization (almost all of our previous observations from the unpruned tree still applies here), and helps us more easily observe the different factors that may affect a voter's decision.

17. Run a logistic regression to predict the winning candidate in each county.

	train.error	test.error
tree	0.0610749	0.0846906
logistic	0.0692182	0.0830619
lasso	NA	NA

What are the significant variables? Are the consistent with what you saw in decision tree analysis? Interpret the meaning of a couple of the significant coefficients in terms of a unit change in the variables.

The significant variables are: Citizen, IncomePerCap, Professional, Service, Production, Drive, Carpool, Employed, PrivateWork, Unemployment. Yes, for the most part, they are consistent with what we saw in the decision tree. For example, Employment and Unemployment are significant because they determine what social class the person belongs to, which made a difference in which candidate the voters chose. Voters who Carpooled, similar to people who utilized Transit, indicated a more liberal and possibly lower income status due to their awareness of protecting the environment and saving money. Meanwhile, the Drive category indicated a more financially stable group of people who were able to drive themselves and not worry about saving money.

18.Use the cv.glmnet function from the glmnet library to run K-fold cross validation and select the best regularization parameter for the logistic regression with LASSO penalty.

	train.error	test.error
tree	0.0610749	0.0846906
logistic	0.0692182	0.0830619
lasso	0.0732899	0.0830619

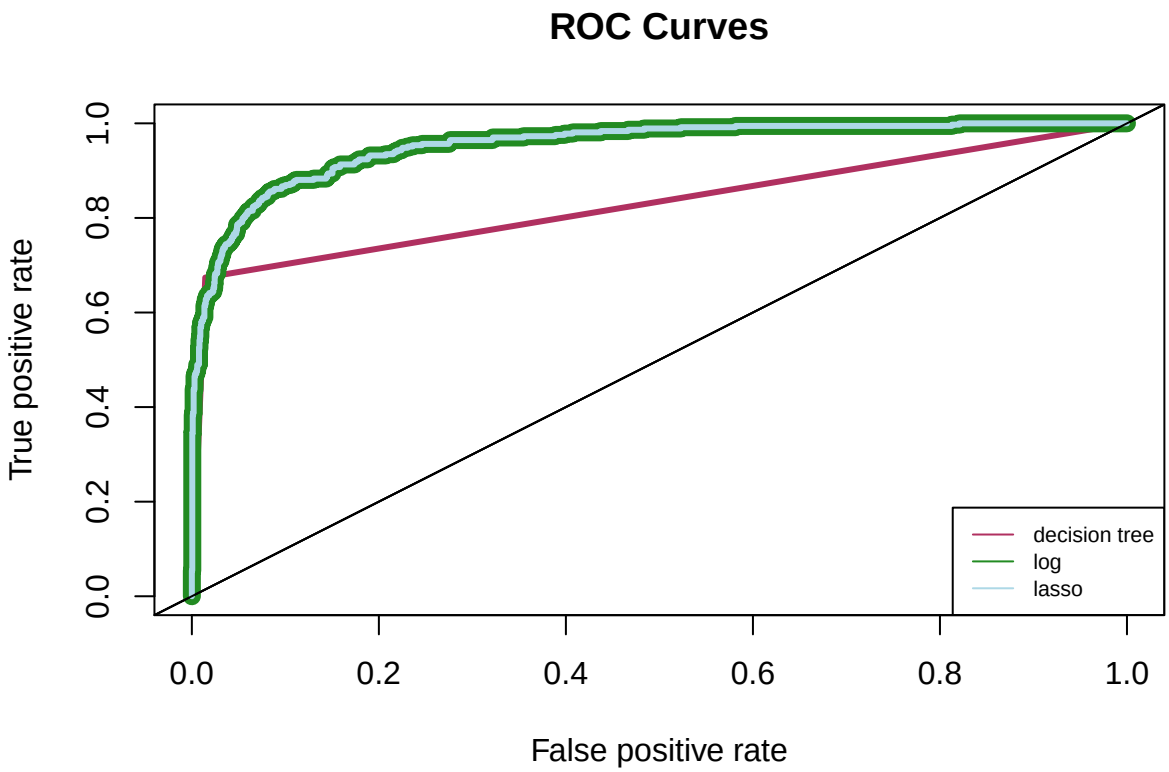
What is the optimal value of in cross validation? What are the non-zero coefficients in the LASSO regression for the optimal value of ? How do they compare to the unpenalized logistic regression? Save training and test errors to the records variable.

The best value in cross validation is 0.0001. The non-zero coefficients in the LASSO regression for the optimal value of were all of the variables excluding Transit, SelfEmployed, and Minority. There were mostly non-zero coefficients because many of the variables in our dataset affect the outcome. While logistic regression is great for big data sets, the LASSO regression is meant for data sets with not enough data, which have estimates with high variance. The purpose of the LASSO regression is to use the shrinkage method to reduce the variance. However, since our data set is large and many of our variables actually affect our outcome, they didn't have a coefficient of zero. In fact, the largest non-zero coefficients that we had were Men, Office, MeanCommute, and PrivateWork, which were the same variables with the highest estimates from the logistic regression.

Compared to the unpenalized logistic regression, the lasso regression has less variables needed to work with due to some variables being equal to zero.

Overall, the lasso and logistic regression fits look very similar. Errors are close to each other because there is already enough data to estimate the coefficients to high accuracy. Therefore, the LASSO regression does not provide any more insightful in this scenario, unlike a different scenario with a smaller data set.

19. Compute ROC curves for the decision tree, logistic regression and LASSO logistic regression using predictions on the test data.



Based on your classification results, discuss the pros and cons of the various methods. Are the different classifiers more appropriate for answering different kinds of questions about the election?

According to the ROC curves, the logistic regression and lasso methods return the highest true positive rate, as both curves hug the upper left corner the most. We also decided to calculate the AUC (Area Under Curve) for each method to better determine which method provides the best predictions. The AUC for the decision tree method is the lowest at 0.8297726 while the AUC values for logistic regression and the lasso method are the highest and the exact same value at 0.9528419, confirming that these two methods are the best predictive models.

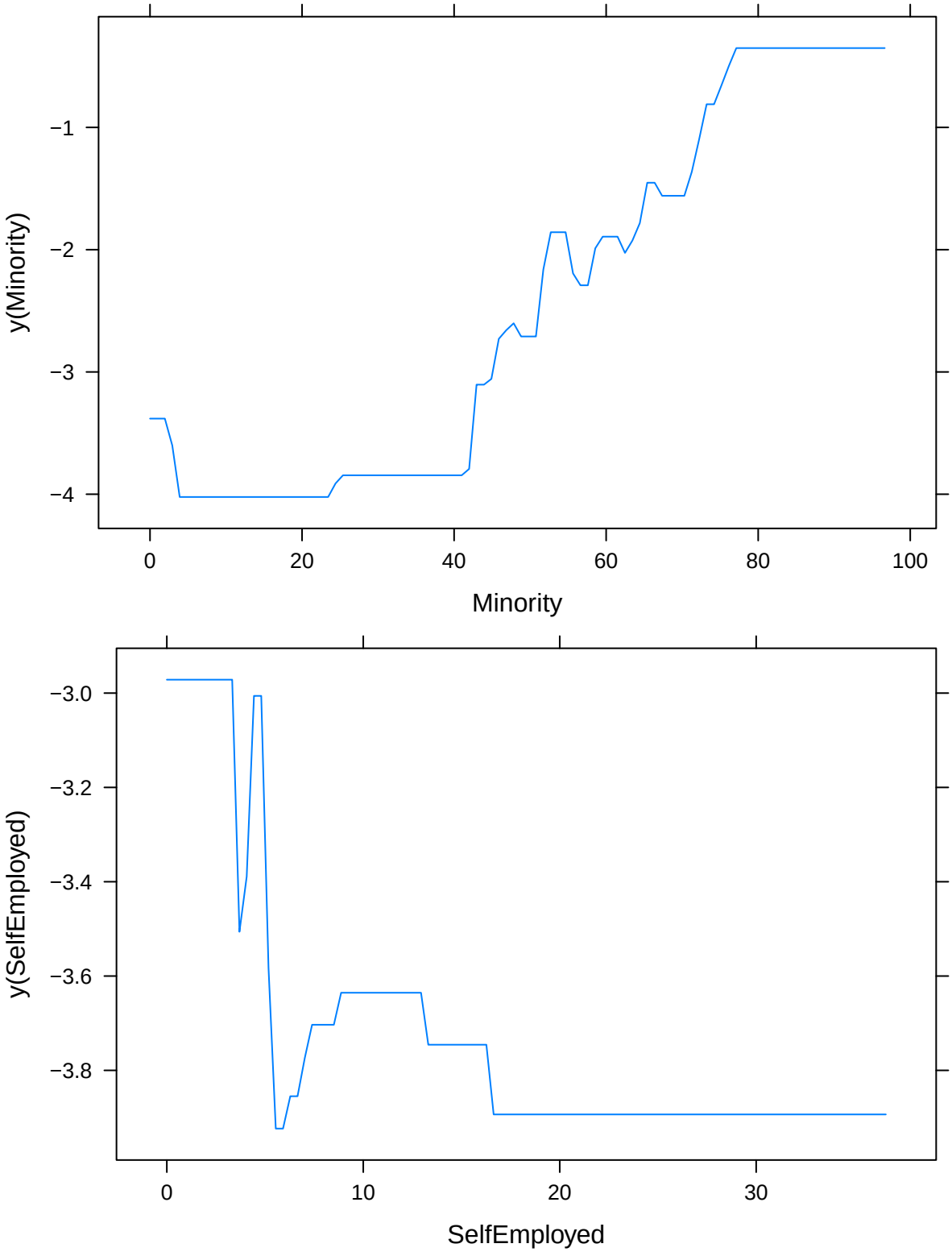
In regards to which classifier is more appropriate, we believe that in the context of understanding voter behavior which entails narrowing down the most important variables, the decision tree fails to answer this election question better or equally as well as the lasso and logistic regression models. The lasso is able to do so with our large data set, and the logistic regression model helps us better identify, through perfect separation, the best candidate in the election for each variable group, thus giving us more insight on voter behavior.

Taking it further

20. Interpret and discuss any overall insights gained in this analysis and possible explanations. Use any tools at your disposal to make your case: visualize errors on the map, discuss what does/doesn't seems reasonable based on your understanding of these methods, propose possible directions (collecting additional data, domain knowledge, etc).

For our final open question, we decided to explore the classification methods: boosting, bagging, and random forests. Our goal is to fit these methods to our data and compare their respective final errors, deeming the method with the smallest error as the best.

Boosting



	test.error
boosting	0.9983713

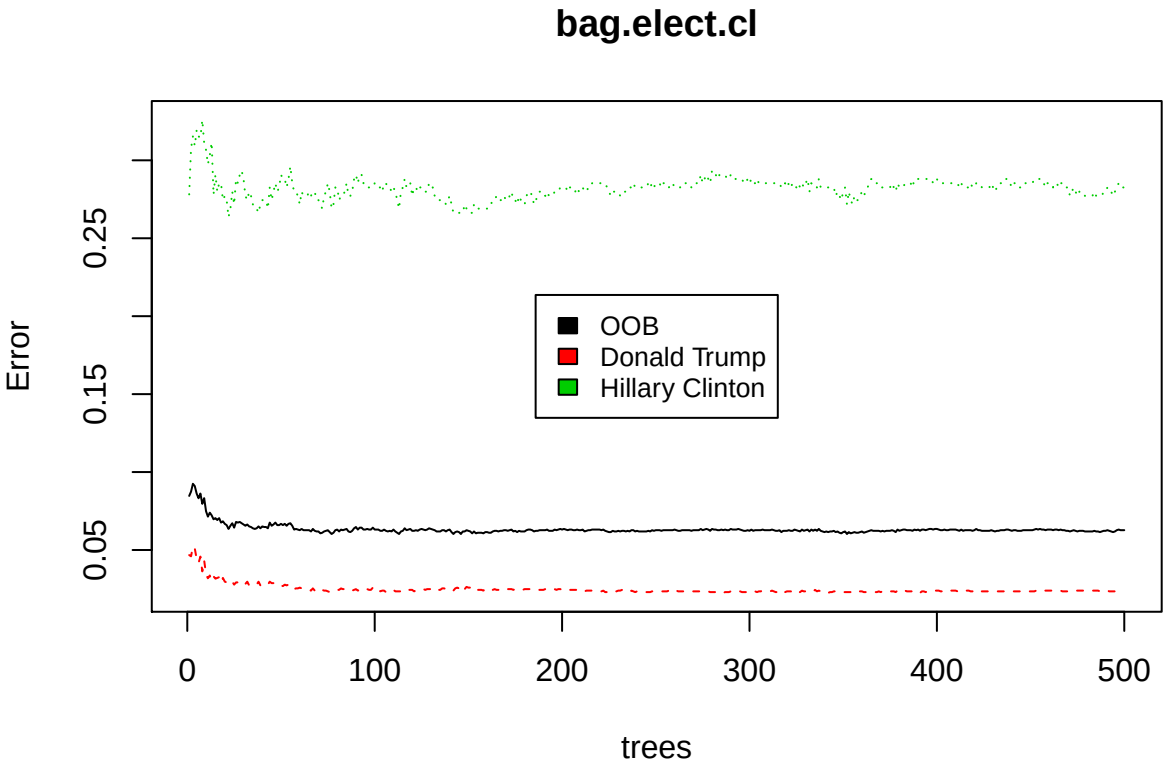
First, we used the Boosting method. In doing so, we received an error of 0.9983713, which is a significantly high error. This may be due to the fact that boosting is more fit for smaller data sets and decision trees.

From our relative influence graph, we are told that the variables Minority, Self-Employed, and Child Poverty are the more influential variables. However, this does not display all of our variables and thus, does not provide enough information. It is also inconsistent with our Random Forest and the Self-Employment graph that we plotted below, which shows that Self-Employed actually decreases.

Overall, the Boosting method is not as great at fitting our large data set as logistic regression and other previously used methods. Its plots and graphs do not provide us with enough information as to how to interpret the variables' importance to our data and voter behavior.

Bagging

Next, we tested the bagging method. In doing so, we observed an error of 0.04885. which was relatively lower than the Boosting method and therefore the better method so far. We believe that it has a small error due to the fact that bagging involves using large unpruned decision trees, which our data set can provide. Although this method only uses 2/3 of the total data and may not provide as much insight on variable importance, it nonetheless reduces variance and is great for larger data sets. Overall, it is not as great of a classification method as logistic regression and the decision tree, it is still better than the boosting method in this case.



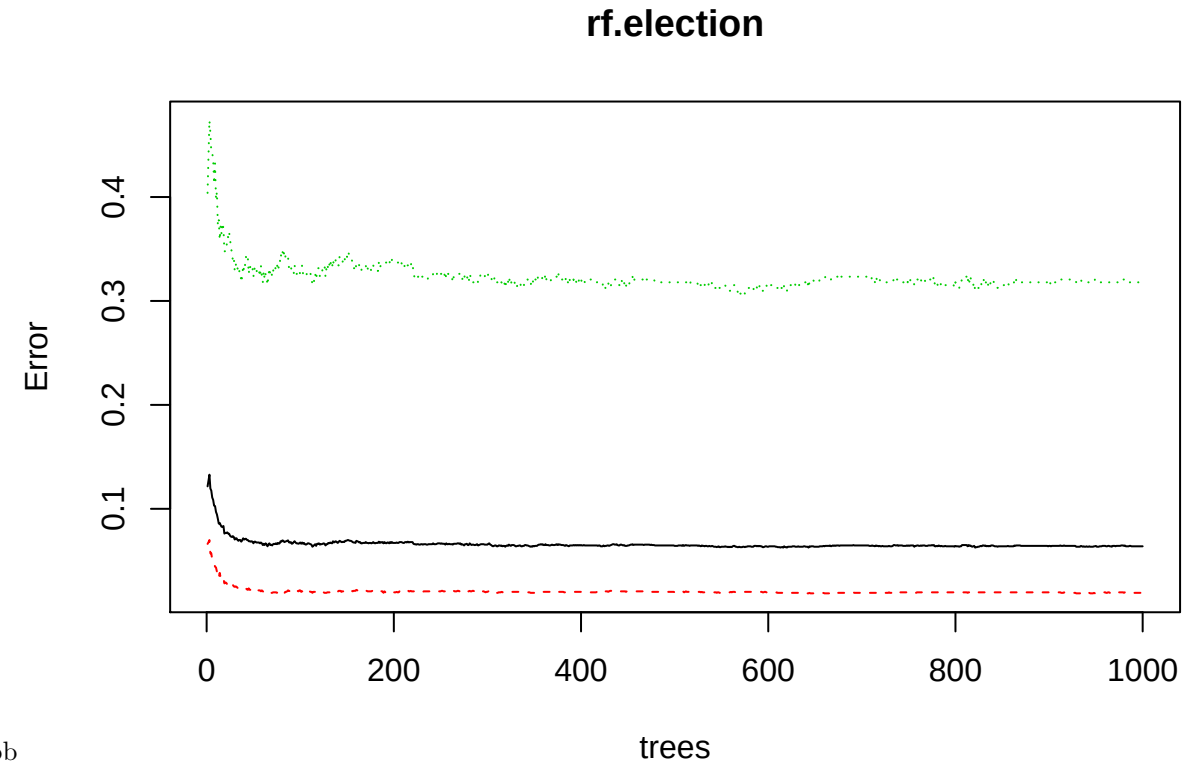
	test.error
boost error	0.9983713
bag error	0.0488599

Random Forest

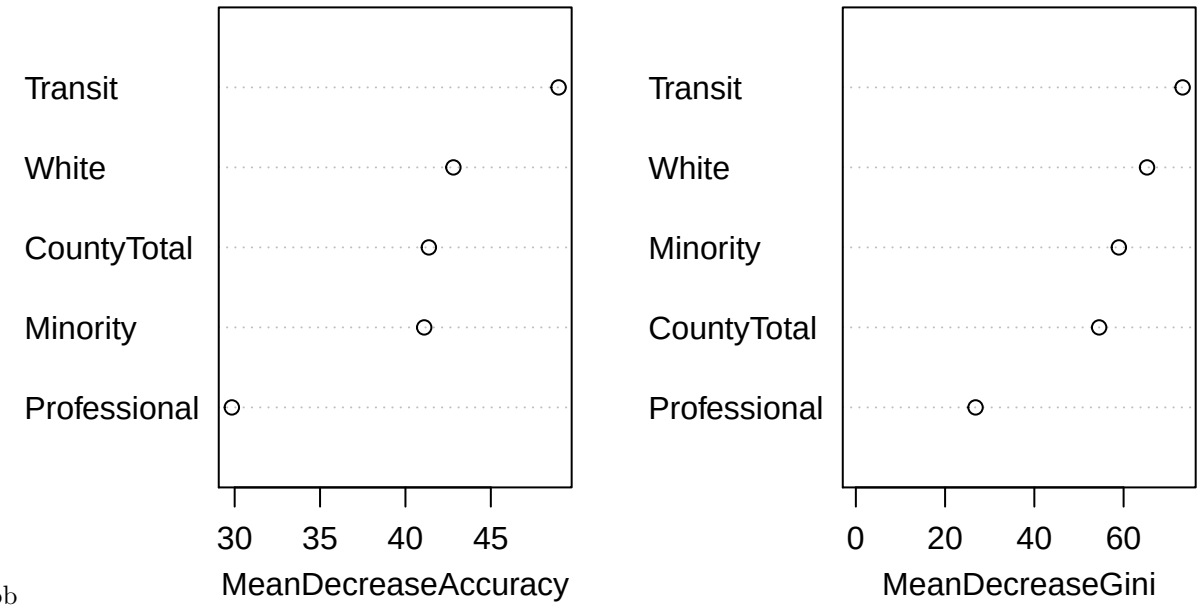
Lastly, we computed random forest by creating more trees. In doing so, we get an error of 0.04885, which was the same as our Bagging method. This is small and a good result for a large data set. Looking into it further, we can see from the Variance Importance charts that the variables Transit, White and Minority play the biggest roles in decreasing the Gini impurity, which is one of the main goals of this method. Following closely are the variables County Total and Professional. This result is also supported by previous methods used in this project, such as our decision trees, which show the same variables with the most important towards the top of the trees, and the rest branching downwards.

In context of the election, this makes sense due to the fact that a voter's demographic as well as social-economic status played a role in their choice of presidential candidate (for example, most white people tend to vote for Donald Trump while Minority voted for Hillary Clinton).

Overall, although it is prone to over-fitting like the logistic regression model, the random forest tree is an informative method that helps identify strong predictors in the data set.



forest-1.bb



forest-2.bb

	train.error	test.error
tree	0.0610749	0.0846906
logistic	0.0692182	0.0830619
lasso	0.0732899	0.0830619

	test.error
boosting	0.9983713
bagging	0.0488599
random forest	0.0488599

In conclusion, from this open question, we found that of the three additional methods that we tested, Boosting is the worst with an error close to 1. On the other hand, Bagging and Random Forest both have errors that were similar to the test errors of our previously used methods: Decision Tree, Logistic Regression, and Lasso.

Appendix

```
knitr::opts_chunk$set(echo = FALSE, cache = TRUE, eval = FALSE)
indent1 = '    '
indent2 = paste(rep(indent1, 2), collapse='')
indent3 = paste(rep(indent1, 3), collapse='')

library(knitr)
library(tidyverse)
library(ggmap)
library(maps)
library(Rtsne)
library(NbClust)
library(tree)
library(randomForest)
library(maptree)
library(class)
library(reshape2)
library(ggplot2)
library(kableExtra)
library(dplyr)
library(glmnet)
library(ROCR)
library(cluster)
library(plotmo)
library(gbm)
```

Data

```
## read data and convert candidate from string to factor
election.raw <- read_delim("/Users/laurenwong/Downloads/data/election/election.csv",
  delim = ",") %>%
  mutate(candidate=as.factor(candidate))

census_meta <- read_delim("/Users/laurenwong/Downloads/data/census/metadata.csv",
  delim = ";", col_names = FALSE)
census <- read_delim("/Users/laurenwong/Downloads/data/census/census.csv",
  delim = ",")
```

Election Data

```
kable(election.raw %>%
  filter(county == "Los Angeles County")) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
    full_width=FALSE)
```

```
##Subsetting Election (county-level)
election.raw <- subset(election.raw, !election.raw$fips==2000)
dim(election.raw)
```

Census Data

```
kable(census %>% head, "html")  %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
  full_width=FALSE) %>%
  scroll_box(width = "100%")
```

```
kable(census_meta)
```

Data wrangling

```
#federal level
election_federal <- filter(election.raw, is.na(county) & fips =="US")
election_federal
```

```
#state level State-level summary rows have names of each states as fips value.
election_state <- filter(election.raw, fips != "US" & is.na(county) & fips != "DC" &
  as.character(election.raw$fips) == as.character(election.raw$state))
election_state
```

```
#county level
election <- filter(election.raw, !is.na(county))
election
```

6

```
num.candidate<- unique(election.raw$candidate)
length(num.candidate)
num.candidate
```

```
#bar plot
ggplot(election.raw, aes(x=candidate, y=(votes))) +
  geom_bar(stat="identity",fill="lightblue") +
  theme(text = element_text(size=10),
  axis.text.x = element_text(angle=90, hjust=1)) +
  coord_flip()
```

Visualization

```
# create county_winner variable
county_winner <- election %>%
group_by(fips) %>%
mutate(total=sum(votes), pct = votes/total) %>%
top_n(1)
# create state_winner variable
state_winner <- election_state %>%
group_by(fips) %>%
mutate(total=sum(votes), pct = votes/total) %>%
top_n(1)
```

```
states = map_data("state")
ggplot(data = states) +
geom_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white") +
coord_fixed(1.3)+guides(fill=FALSE)+ggtitle("State-Level Map")
```

8

```
counties <- map_data("county")
ggplot(data = counties) +
  geom_polygon(aes(x = long, y = lat, fill = subregion, group = group), color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE) +
  ggtitle("County-Level Map")
```

9

```
states = map_data("state")
states = mutate(states, fips = state.abb[match(states[,5],tolower(state.name))])
states <- left_join(state_winner,states,by=c("state"="fips"))
```

```
ggplot(data=states) +
  geom_polygon(aes(x = long, y = lat, fill = candidate, group = group),
               color = "white") +
  coord_fixed(1.3) +
  guides(fill=F) +
  ggtitle("Winning Candidate by State")
```

10

```
county.fips <- maps::county.fips %>%
  separate(polyname, c("region", "subregion"), sep=",") %>%
  mutate(fips=as.factor(fips))
join_first=left_join(counties, county.fips, by=c("region","subregion"))
join_counties=left_join(join_first, county_winner, by=c("fips"))
```

```
ggplot(data=join_counties) +
  geom_polygon(aes(x = long, y = lat, fill = candidate,
  group = group), color = "white") +
  coord_fixed(1.3) +
  ggtitle("Winning Candidate per County")
```

11

```
#Mutate data to make the Minority column
census_new <- census %>%
  dplyr::mutate (Minority = Hispanic + Black + Native + Asian + Pacific)
#will mutate this data more for number 11
```

```r
#group by white people
census_white <- aggregate(census_new$White, by=list(census_new$State),
    FUN = sum, na.rm = TRUE)
#group by minority
census_min <- aggregate(census_new$Minority, by=list(census_new$State),
    FUN = sum, na.rm = TRUE)

#plot for white
plot_white <- ggplot(census_white, aes(x=census_white$Group.1,y=census_white$x)) +
  geom_point(color= "maroon", size=0.7) +
  theme(text = element_text(size=10),
  axis.text.x = element_text(angle=90, hjust=1)) +
  labs(x="States", y="White")

#plot for minority
plot_min <- ggplot(census_min, aes(x=census_min$Group.1,y=census_min$x)) +
  geom_point(color= "navy", size=0.7) +
  theme(text = element_text(size=10),
  axis.text.x = element_text(angle=90, hjust=1)) +
  labs(x="States", y="Minority")

plot_white
plot_min
```

12

```r
#clean census data
census.del <- census[2:37] %>%
  na.exclude(census) %>%
  mutate(Men = Men/TotalPop*100,
         Employed = Employed/TotalPop*100,
         Citizen = Citizen/TotalPop*100,
         Minority = Hispanic + Black + Native + Asian + Pacific) %>%
  dplyr::select(-Walk, -PublicWork, -Construction, -Hispanic,
  -Black, -Native, -Asian, -Pacific, -Women)
```

```r
#sub-county
census.subct <- census.del %>%
  group_by(State,County) %>%
  add_tally(TotalPop) %>%
  mutate(CountyTotal = n, Weight = TotalPop/CountyTotal) %>%
  select(-n) %>%
  ungroup
```

```r
#county
census.ct <- census.subct %>%
  group_by(State,County) %>%
  summarise_at(vars(Men:CountyTotal), funs(weighted.mean))
head(census.ct)
```

Dimensionality reduction

13

```
#county-level
pc1 <- prcomp(census.ct[3:28], scale=TRUE, center=TRUE)
#county has only 26 PCs
dim(pc1$x)

#sub-county level
pc2 <- prcomp(census.subct[3:30], scale = TRUE, center = TRUE)
#sub-county has only 28 PCs
dim(pc2$x)

#convert to dataframe
ct.pc <- data.frame(pc1$x[,1:2])
subct.pc <- data.frame(pc2$x[,1:2])

#rotation matrix
ct_rotation <- data.frame(pc1$rotation[,1])
abs(ct_rotation)
ct_rotation

subct_rotation <- data.frame(pc2$rotation[,1])
abs(subct_rotation)
subct_rotation
```

14

```
#PC1 uses census.ct for counties
pr.var1 <- pc1$sd^2 #variance explained by each principal component
pve1 <- pr.var1 / sum(pr.var1) #proportion of variance explained by each principal component
cumulative_pve1 <- cumsum(pve1)

#PC2 uses census.subct for subcounties
pr.var2 <- pc2$sd^2 #variance explained by each principal component
pve2 <- pr.var2 / sum(pr.var2) #proportion of variance explained by each principal component
cumulative_pve2 <- cumsum(pve2)

par(mfrow=c(2, 2)) #graphs side by side

#plot of county-level
plot(pve1, type="l", lwd=3, xlab="Principal Component 1", ylab="Counties PVE")
plot(cumulative_pve1, type="l", xlab="Principal Component 1", ylab="County Cumulative
PVE", ylim=c(0, 1), lwd=3)

#plot of sub-county level
plot(pve2, type="l", lwd=3, xlab="Principal Component 2", ylab="Subcounties PVE")
plot(cumulative_pve2, type="l", xlab="Principal Component 2", ylab="Subcounties Cumulative
PVE", ylim=c(0, 1), lwd=3)
```

```
#minimum number of PCs needed for 90% of variance
PC1.Num <- which(cumulative_pve1>=0.9)[1]
#PC1.Num #county
PC2.Num <- which(cumulative_pve2>=0.9)[1]
#PC2.Num #sub-county
```

Clustering

15

```
census.ct_clust <- scale(census.ct[3:28])
census.ct_dist <- dist(census.ct_clust, method = "euclidean")
census.ct_hclust <- hclust(census.ct_dist, method = "complete")
clust1 <- cutree(census.ct_hclust, k = 10)
table(clust1)

#reclust with 2 principal components
ct.pc <- data.frame(pc1$x[,1:2])
ct.pc_clust <- scale(ct.pc)
ct.pc_dist <- dist(ct.pc_clust, method = "euclidean")
ct.pc_hclust <- hclust(ct.pc_dist, method = "complete")
clust2 <- cutree(ct.pc_hclust, k = 10)
table(clust2)

#San Mateo
clust1[which(census.ct$County == "San Mateo")]
clust2[which(census.ct$County == "San Mateo")]
```

Classification

```
tmpwinner <- county_winner %>%
  ungroup %>%
  mutate(state = state.name[match(state, state.abb)]) %>%               ## state abbreviations
  mutate_at(vars(state, county), tolower) %>%                           ## to all lowercase
  mutate(county = gsub(" county| columbia| city| parish", "", county))  ## remove suffixes
tmpcensus <- census.ct %>%
  dplyr::ungroup() %>%
  mutate_at(vars(State, County), tolower)

election.cl <- tmpwinner %>%
  left_join(tmpcensus, by = c("state"="State", "county"="County")) %>%
  na.omit

## save meta information
election.meta <- election.cl %>%
  select(c(county, fips, state, votes, pct, total))

## save predictors and class labels
```

```
election.cl = election.cl %>%
  select(-c(county, fips, state, votes, pct, total))
```

```
set.seed(10)
n <- nrow(election.cl)
in.trn <- sample.int(n, 0.8*n)
trn.cl <- election.cl[ in.trn,]
tst.cl <- election.cl[-in.trn,]
```

```
set.seed(20)
nfold <- 10
folds <- sample(cut(1:nrow(trn.cl), breaks = nfold, labels = FALSE))
```

```
calc_error_rate = function(predicted.value, true.value){
  return(mean(true.value!=predicted.value))
}
records = matrix(NA, nrow = 3, ncol = 2)
colnames(records) = c("train.error","test.error")
rownames(records) = c("tree","logistic","lasso")
```

16

```
#training error
trn.clX = trn.cl %>% select(-candidate)
trn.clY = trn.cl$candidate
#test error
tst.clX = tst.cl %>% select(-candidate)
tst.clY = tst.cl$candidate

tree_params <- tree.control(nrow(trn.cl))
#using a 10 fold CV to select tree which minimizes cv misclassification error.
model_tree <- tree(as.factor(candidate) ~ ., data = trn.cl, control = tree_params)
set.seed(1)
cv <- cv.tree(model_tree, folds, FUN = prune.misclass, K = 10)
tree_select <- which(cv$dev == min(cv$dev))

#select the smallest tree size w/ that minimum rate
best_size <- min(cv$size[tree_select]) #best size is 9

#visualize the tree before pruning
draw.tree(model_tree, nodeinfo=TRUE, cex=0.50)
title("Unpruned Tree")
```

```
#visualize the tree after pruning
#prune the tree to the size found and plot with draw.tree
pruned_tree <- prune.tree(model_tree, best = best_size,method="misclass")
#summary(pruned_tree)
draw.tree(pruned_tree, nodeinfo=TRUE, cex=0.50)
title("Pruned Tree")

# traning error
pred_pruned <- predict(pruned_tree, trn.clX, type = "class")
pruned_error <- calc_error_rate(pred_pruned, trn.clY)
```

```r
# test error
pred_tr <- predict(pruned_tree, tst.clX, type = "class")
error_tr <- calc_error_rate(pred_tr, tst.clY)
```

```r
# putting errors into records
records[1,1] <- pruned_error
records[1,2] <- error_tr
kable(records)
```

17

```r
glm_fit <- glm(factor(candidate)~., data = trn.cl, family = "binomial")
summary(glm_fit)
```

```r
set.seed(1)
#glm train
glm_train <- predict(glm_fit, newdata = trn.cl, type = "response")
winner_train <- factor(ifelse(glm_train < 0.5, "Donald Trump", "Hillary Clinton"),
    levels=c("Donald Trump", "Hillary Clinton"))
result <- factor(ifelse(trn.cl$candidate == "Donald Trump","Donald Trump", "Hillary Clinton"))
table(predicted = winner_train, true = result)

#glm test
glm_test <- predict(glm_fit, newdata = tst.cl, type = "response")
winner_test <- factor(ifelse(glm_test < 0.5, "Donald Trump", "Hillary Clinton"),
    levels=c("Donald Trump", "Hillary Clinton"))
result2 <- factor(ifelse(tst.cl$candidate == "Donald Trump","Donald Trump", "Hillary Clinton"))
table(predicted = winner_test, true = result2)
```

```r
#train and test errors
glm_train_error <- calc_error_rate(winner_train, result)
glm_test_error <- calc_error_rate(winner_test, result2)
#print matrix
records[2,1] <- glm_train_error
records[2,2] <- glm_test_error
kable(records)
```

18

```r
y.trn <- ifelse(trn.cl[,1] == "Donald Trump", 0, 1)
x.trn <- model.matrix(candidate~. , trn.cl)[,-1]

set.seed(1)
cv_lasso <- cv.glmnet(lambda = c(1, 5, 10, 50) * 1e-4, x.trn, y.trn, foldid = folds,
    alpha =1,family = "binomial")
plot(cv_lasso)
bestlambda <- cv_lasso$lambda.min #1e-04
abline(v = log(bestlambda), col="navy", lwd = 3, lty = 2)

lasso_mod <- glmnet(x.trn, y.trn, alpha = 1, family = "binomial")
coeff <- predict(lasso_mod, type = "coefficients", s = bestlambda)
```

```
plot.glmnet(lasso_mod, xvar="lambda", label = TRUE)

x.tst = model.matrix(candidate~., tst.cl)[,-1]

set.seed(1)
lasso_train_pred = predict(lasso_mod, newx = x.trn, s = bestlambda)
lasso_train = ifelse(lasso_train_pred < 0.5, "Donald Trump","Hillary Clinton")
las_train_err = calc_error_rate(as.tibble(lasso_train), trn.cl[,1])

lasso_test_pred = predict(lasso_mod, newx = x.tst, s = bestlambda)
lasso_test = ifelse(lasso_test_pred < 0.5, "Donald Trump","Hillary Clinton")
las_test_err = calc_error_rate(as.tibble(lasso_test), tst.cl[,1])

#non-zero coeff
#as.matrix(coeff)
```

```
#matrix
records[3,1] <- las_train_err
records[3,2] <- las_test_err
kable(records)
```

19

```
#mutating candidates 0 = Hillary, 1 = Trump
trn.m = trn.cl %>%
  mutate(candidate = as.factor(ifelse(candidate == 0, "Hillary Clinton", "Donald Trump")))
```

```
#decision tree
pred_tree <- prediction(as.numeric(pred_pruned), as.numeric(trn.cl$candidate))
perf_tree <- performance(pred_tree, measure = 'tpr', x.measure = 'fpr')
plot(perf_tree, col = "maroon", lwd = 3, main = "ROC Curves")
abline(0,1)

#logistic
pred_log <- prediction(as.numeric(glm_train), as.numeric(trn.cl$candidate))
perf_log <- performance(pred_log, measure = 'tpr', x.measure = 'fpr')
plot(perf_log, add = TRUE, col = "forestgreen", lwd = 9)
abline(0,1)

#lasso
pred_lasso <- prediction(as.numeric(glm_train), as.numeric(trn.cl$candidate))
perf_lasso <- performance(pred_log, measure = 'tpr', x.measure = 'fpr')
plot(perf_lasso, add = TRUE, col = "lightblue", lwd = 3)
abline(0,1)
legend("bottomright", legend = c("decision tree", "log", "lasso"),
    col = c("maroon","forestgreen", "lightblue"), lty = 1, cex = 0.7)
```

```
auc_tree <- performance(pred_tree, "auc")@y.values #0.8297726
auc_tree
auc_log <- performance(pred_log, "auc")@y.values  #0.9528419
auc_log
auc_lasso <- performance(pred_lasso, "auc")@y.values  #0.9528419
```

```
auc_lasso
```

Taking it further

20

```
set.seed(1)
#Trump = 0, Clinton = 1
true_test <- as.numeric(ifelse(tst.cl$candidate == "Donald Trump", 0,1))
boost.elect.cl <- gbm(ifelse(candidate == "Donald Trump", 0,1)~., data = trn.cl,
    distribution = "bernoulli", n.trees = 800)
#summary(boost.elect.cl, main = "Boosting Election.cl")

par(mfrow = c(1,2))
plot(boost.elect.cl, i = "Minority", ylab= "y(Minority)")
```

```
plot(boost.elect.cl, i = "SelfEmployed", ylab= "y(SelfEmployed)")

yhat.boost <- predict(boost.elect.cl, newdata = tst.cl, n.trees = 800, type = "response")
```

```
#confusion matrix
boost.error <- table(pred = yhat.boost, truth = true_test)
test.boost.error <- 1 - sum(diag(boost.error))/sum(boost.error) #0.9983713
record1 <- matrix(c(test.boost.error, test.boost.error), nrow = 1, ncol = 1)
colnames(record1) = c("test.error")
rownames(record1) = c("boosting")
kable(record1)
```

```
set.seed(1)
trn.cl$candidate <- factor(trn.cl$candidate)
bag.elect.cl <- randomForest(candidate~., data = trn.cl, mtry = 10, importance = TRUE)

plot(bag.elect.cl)
legend("center", colnames(bag.elect.cl$err.rate), col = 1:4, cex = 0.8, fill = 1:4)

bag.elect.cl <- randomForest(candidate~., data = trn.cl, mtry = 10, ntree = 700,
    importance = TRUE)
yhat.bag <- predict(bag.elect.cl, newdata = tst.cl)

#confusion matrix
bag.error <- table(pred = yhat.bag, truth = true_test)
test.bag.error <- 1 - sum(diag(bag.error))/sum(bag.error)   #0.04885993
record1 <- matrix(c(test.boost.error, test.bag.error), nrow = 2, ncol = 1)
colnames(record1) = c("test.error")
rownames(record1) = c("boost error","bag error")
kable(record1)
```

```
set.seed(1)
options(stringsAsFactors = FALSE)
true_test <- as.numeric(ifelse(tst.cl$candidate == "Donald Trump", 0,1))

#glimpse(election) #18,007 observations, 5 variables
```

```r
#change candidate to factor
trn.cl$candidate <- factor(trn.cl$candidate)
rf.election <- randomForest(candidate~., data = trn.cl, mtry = 3, ntree = 1000,
    importance = TRUE)
plot(rf.election)

yhat.rf <- predict(rf.election, newdata = tst.cl)

#importance(rf.election)
varImpPlot(rf.election, sort = TRUE,
    main = "Variable Importance for Random Forest Election", n.var = 5)
```

```r
#tree, log reg, and lasso records
kable(records)

#create matrix
rf.error <- table(pred = yhat.rf, truth = true_test)
test.rf.error <- 1 - sum(diag(rf.error))/sum(rf.error)  #0.04885993

record1 <- matrix(c(test.boost.error, test.bag.error, test.rf.error), nrow = 3, ncol = 1)
colnames(record1) = c("test.error")
rownames(record1) = c("boosting", "bagging","random forest")
kable(record1)
```