Problem set 8

Jingyuan Zhou

3/4/2017

Part 1: Sexy Joe Biden (redux times two)

1. Let's first split the data into a training set and a testing set

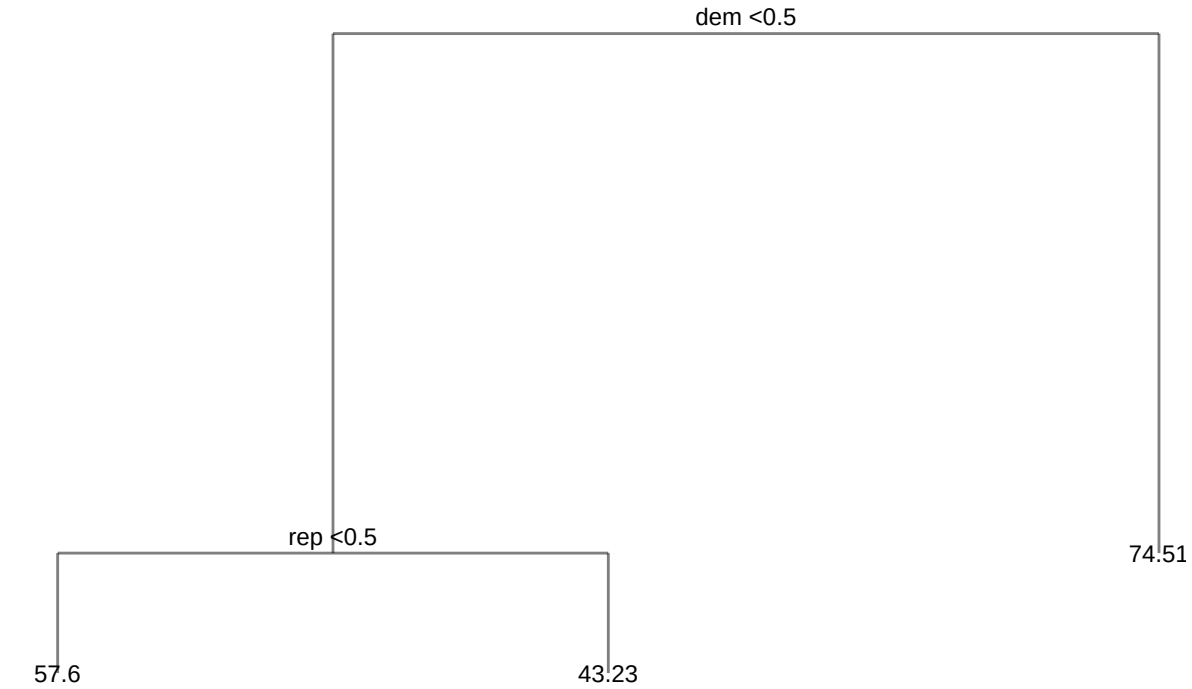
```
biden_split <- resample_partition(biden_data, c(test = 0.3, train = 0.7))
```

2. Fit a decision tree to the training data, with biden as the response variable and the other variables as predictors. Plot the tree and interpret the results. What is the test MSE?

```
# estimate model
biden_tree1 <- tree(biden ~ ., data = biden_split$train)

#mod <- prune.tree(auto_tree, best = 2)

# plot unpruned tree
mod <- biden_tree1

tree_data <- dendro_data(mod)
ggplot(segment(tree_data)) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend),
               alpha = 0.5) +
  geom_text(data = label(tree_data),
            aes(x = x, y = y, label = label_full), vjust = -0.5, size = 3) +
  geom_text(data = leaf_label(tree_data),
            aes(x = x, y = y, label = label), vjust = 0.5, size = 3) +
  theme_dendro() +
  labs(title = "Biden feeling thermometer tree",
       subtitle = "all predictors")
```

Biden feeling thermometer tree

all predictors




```
biden_tree1_mse <- mse(biden_tree1, biden_split$test)
biden_tree1_mse
```

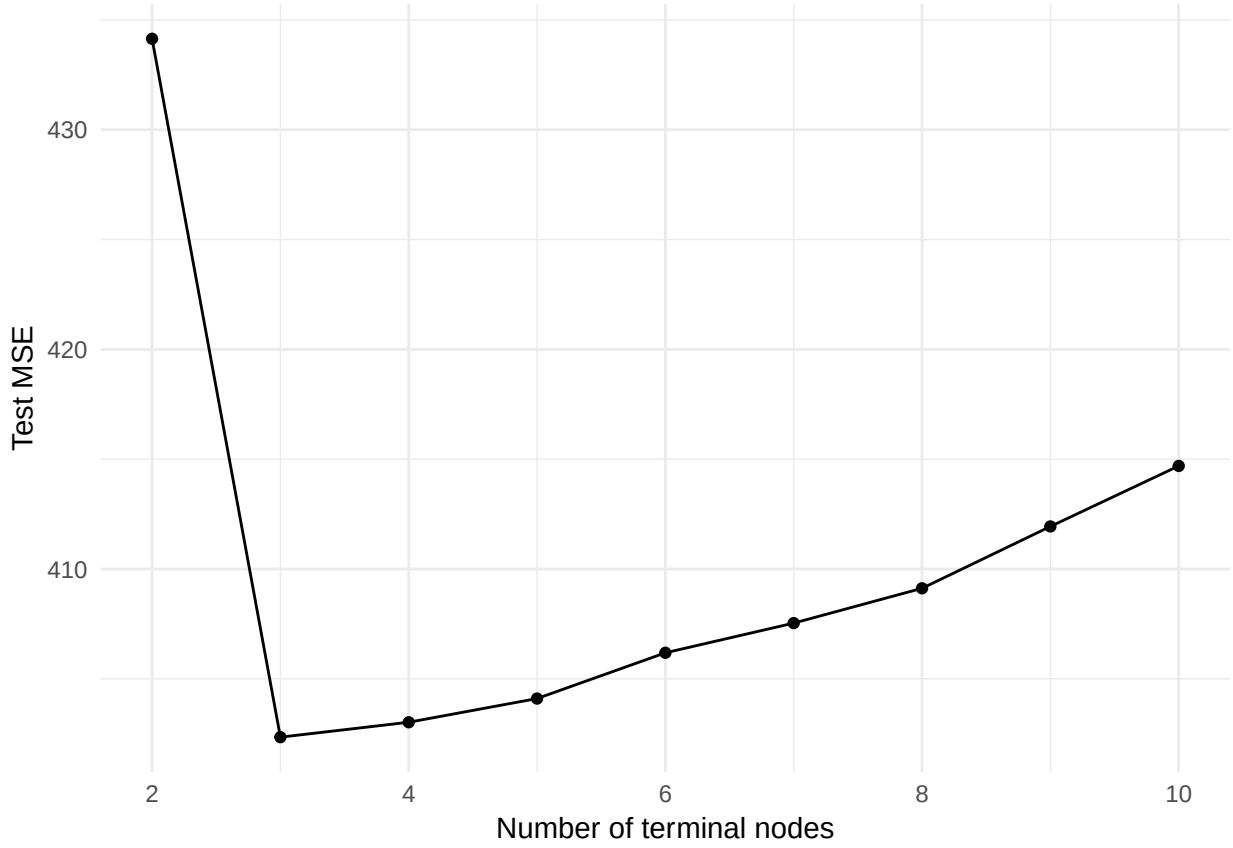
```
## [1] 406
```

Using all other variables as predictors and the default setting for tree(), the decision tree that we've obtained has a mse value of 406 on the test data. This model has three terminal nodes (57.6, 43.23 and 74.51), one internal node (rep<0.5), and three branches. For observations with dem<0.5 and rep<0.5, the person being independent, the model estimates the biden score to be 57.6. For observations with dem<0.5 and rep>=0.5, the person being republican, the model estimates the biden score to be 43.23. For observations with dem>=0.5, the person being democract, the model estimates the biden score to be 74.51.

3. Now fit another tree to the training data with the determined list options. Use cross-validation to determine the optimal level of tree complexity, plot the optimal tree, and interpret the results. Does pruning the tree improve the test MSE?

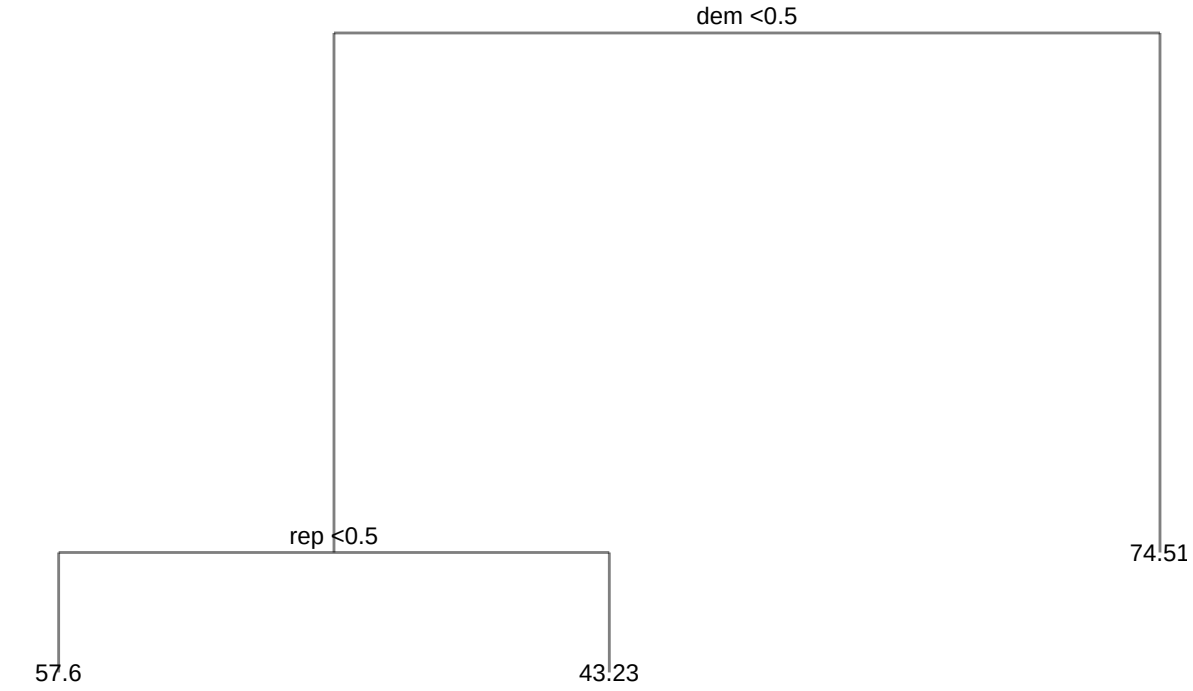
```
# generate 10-fold CV trees
biden_cv <- crossv_kfold(biden_data, k = 10) %>%
  mutate(tree = map(train, ~ tree(biden ~ ., data = ., control = tree.control(nobs = nrow(biden_data) ,

# calculate each possible prune result for each fold
biden_cv <- expand.grid(biden_cv$.id, 2:10) %>%
  as_tibble() %>%
  mutate(Var2 = as.numeric(Var2)) %>%
  rename(.id = Var1,
         k = Var2) %>%
  left_join(biden_cv) %>%
```

```
  mutate(prune = map2(tree, k, ~ prune.tree(.x, best = .y)),
         mse = map2_dbl(prune, test, mse))

biden_cv %>%
  select(k, mse) %>%
  group_by(k) %>%
  summarize(test_mse = mean(mse),
            sd = sd(mse, na.rm = TRUE)) %>%
  ggplot(aes(k, test_mse)) +
  geom_point() +
  geom_line() +
  labs(x = "Number of terminal nodes",
       y = "Test MSE")
```




```
#from the plot we see that the optimal number of terminal nodes is 3
# estimate model
biden_tree2 <- tree(biden ~ ., data = biden_split$train, control = tree.control(nobs = nrow(biden_split$

# plot unpruned tree
mod <- prune.tree(biden_tree2, best = 3)

tree_data <- dendro_data(mod)
ggplot(segment(tree_data)) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend),
               alpha = 0.5) +
  geom_text(data = label(tree_data),
            aes(x = x, y = y, label = label_full), vjust = -0.5, size = 3) +
```

```
  geom_text(data = leaf_label(tree_data),
            aes(x = x, y = y, label = label), vjust = 0.5, size = 3) +
  theme_dendro() +
  labs(title = "Biden feeling thermometer tree",
       subtitle = "all predictors")
```




```
biden_tree2_mse <- mse(mod, biden_split$test)
biden_tree2_mse
```

```
## [1] 406
```

After using ten fold cross validation, we observe that the number of terminal nodes with lowest test MSE is 3. Thus, we prune the tree to make it only have three terminal nodes. The resulting tree is exactly the same as the previous one, so the mse of this tree is also the same, 404. We may guess that cross validation is implemented for the default setting for the `tree()` function.

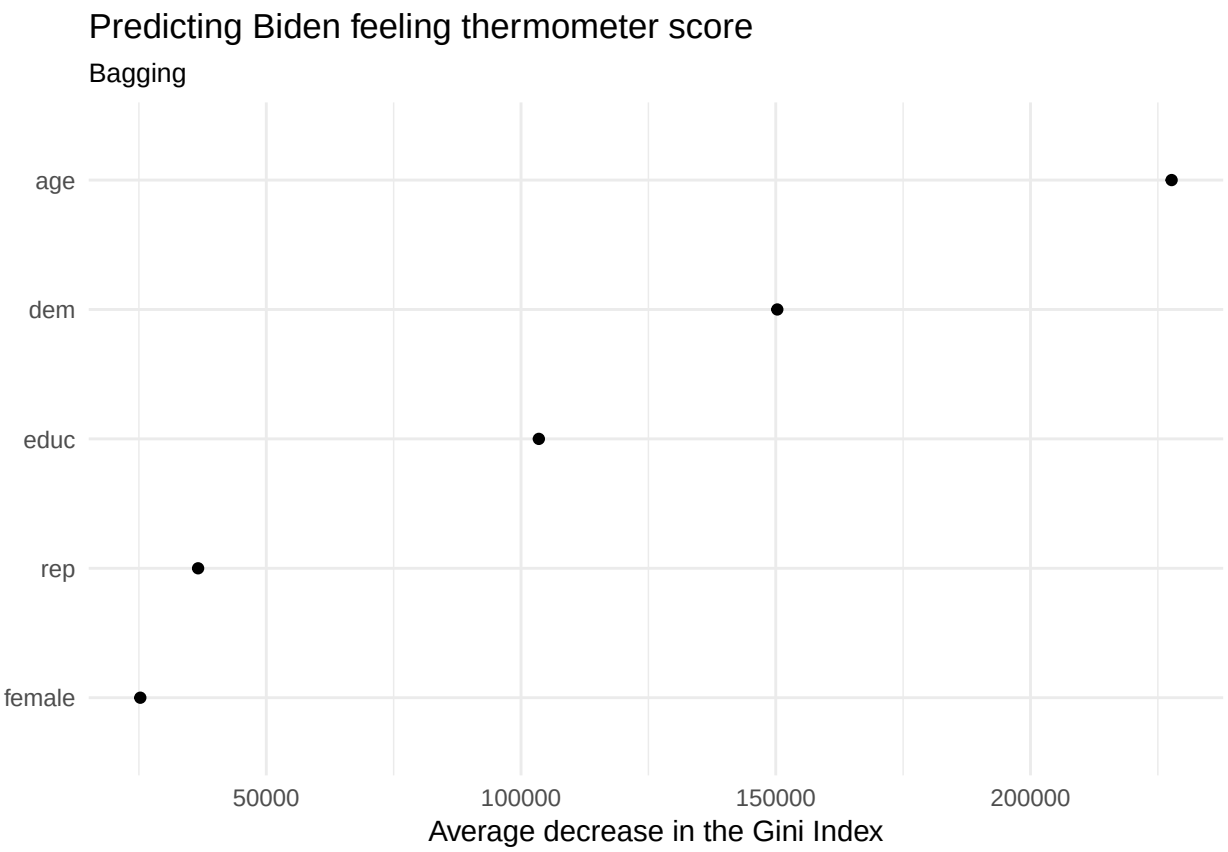
4. Use the bagging approach to analyze this data. What test MSE do you obtain? Obtain variable importance measures and interpret the results.

```
#select(-Name, -Ticket, -Cabin, -Sex, -PassengerId) %>%
# biden_rf_data <- biden_data %>%
#     mutate_each(funs(as.factor(female, dem, rep)), age, educ) %>%
#     na.omit

biden_bag <- randomForest(biden ~ ., data = biden_split$train,
                          mtry = 5, ntree = 500)

biden_bag_mse <- mse(biden_bag, biden_split$test)
```

```
biden_bag_mse
```

```
## [1] 485
```

```
data_frame(var = rownames(importance(biden_bag)),
           MeanDecreaseGini = importance(biden_bag)[,1]) %>%
  mutate(var = fct_reorder(var, MeanDecreaseGini, fun = median)) %>%
  ggplot(aes(var, MeanDecreaseGini)) +
  geom_point() +
  coord_flip() +
  labs(title = "Predicting Biden feeling thermometer score",
       subtitle = "Bagging",
       x = NULL,
       y = "Average decrease in the Gini Index")
```



Test MSE of this bagged model is 484, which is larger than that of our precious models, 406..... To interpret the importance of parameters for this model, the larger the average decrease in the Gini Index is, the more important the parameter is. From the plot, we could see that for this bagged biden feeling thermometer model, age, democract or not and education are the three most important parameters, while republican or not and gender are less important.

5. Use the random forest approach to analyze this data. What test MSE do you obtain? Obtain variable importance measures and interpret the results. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

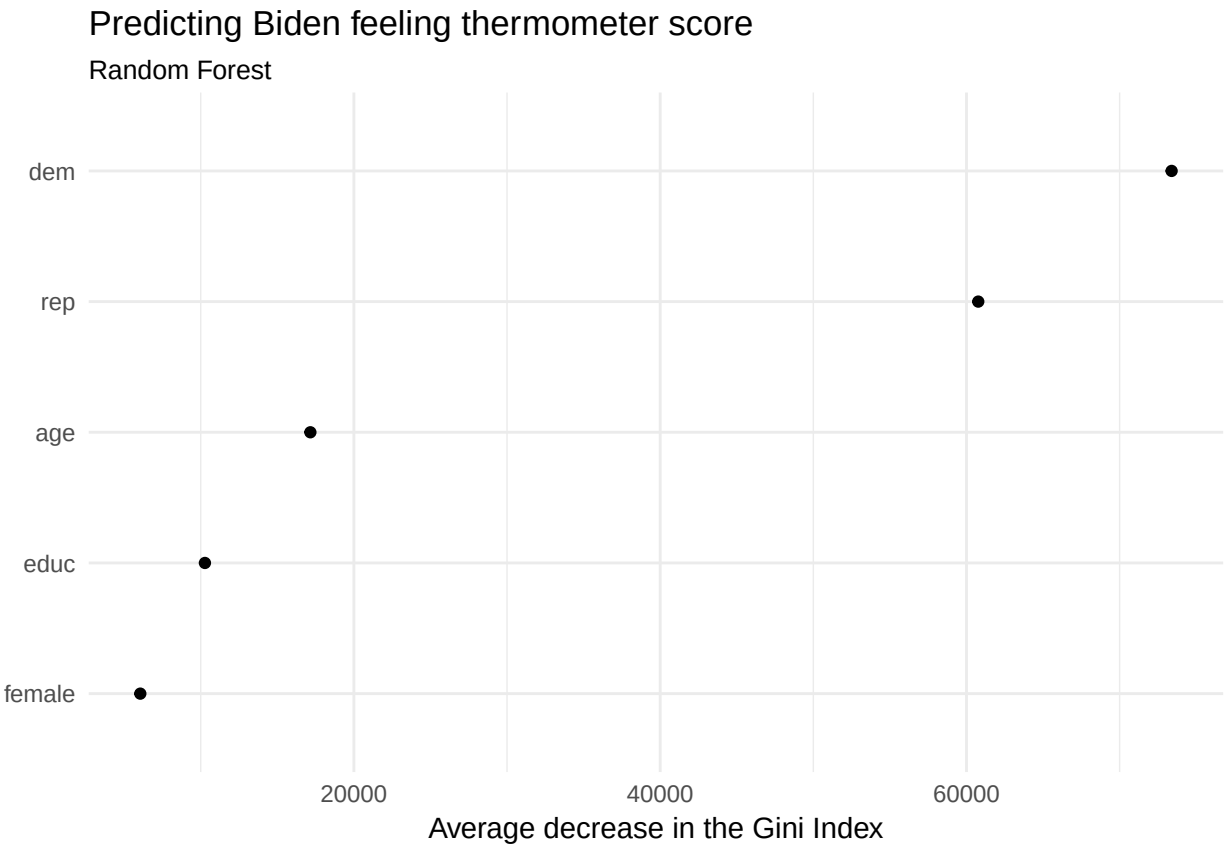
```
biden_rf <- randomForest(biden ~ ., data = biden_split$train, ntree = 500)
biden_rf
```

```
##
## Call:
##  randomForest(formula = biden ~ ., data = biden_split$train, ntree = 500)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 406
##                     % Var explained: 25.7
```

```
biden_rf_mse <- mse(biden_rf, biden_split$test)
biden_rf_mse
```

```
## [1] 409
```

```
data_frame(var = rownames(importance(biden_rf)),
           MeanDecreaseGini = importance(biden_rf)[,1]) %>%
  mutate(var = fct_reorder(var, MeanDecreaseGini, fun = median)) %>%
  ggplot(aes(var, MeanDecreaseGini)) +
  geom_point() +
  coord_flip() +
  labs(title = "Predicting Biden feeling thermometer score",
       subtitle = "Random Forest",
       x = NULL,
       y = "Average decrease in the Gini Index")
```



MSE of the random forest model is 410, 20% smaller to that of the bagged model. Importances of parameters are different from bagging. Looking at the average decrease in the Gini Index of each variable, we can tell that `dem` and `rep` are the two most important parameters, which correspond more to the decision tree models that we've obtained previously. Compared to the importance of each parameters of bagging model, the values of this random forest model are much lower. While `age`, the most important variable of bagging model has an average decrease in the Gini Index as 300000, `dem` and `rep`, the two most important variables of random forest model have average decrease in the Gini Index as about 105000 and 80000 respectively. This might be because of the variable restriction imposed when considering splits.

6. Use the boosting approach to analyze the data. What test MSE do you obtain? How does the value of the shrinkage parameter influence the test MSE?

```
biden_models <- list("boosting" = gbm(biden ~ ., data = biden_split$train,
                                      n.trees = 10000),
                     "boosting_s1" = gbm(biden ~ ., data = biden_split$train,
                                      n.trees = 10000, shrinkage = 0.1),
                     "boosting_s2" = gbm(biden ~ ., data = biden_split$train,
                                      n.trees = 10000, shrinkage = .01),
                     "boosting_s3" = gbm(biden ~ ., data = biden_split$train,
                                      n.trees = 10000, shrinkage = .0001))
```

```
## Distribution not specified, assuming gaussian ...
## Distribution not specified, assuming gaussian ...
## Distribution not specified, assuming gaussian ...
## Distribution not specified, assuming gaussian ...
```

```
mse_boosting <- function(model, data) {
  pred <- predict(model,newdata = as_tibble(data),
                                          n.trees = 10000)
  mean((pred - as_tibble(data)$biden)**2)
}

boosting_mse <- mse_boosting(biden_models$boosting, biden_split$test)
boosting_s1_mse <- mse_boosting(biden_models$boosting_s1, biden_split$test)
boosting_s2_mse <- mse_boosting(biden_models$boosting_s2, biden_split$test)
boosting_s3_mse <- mse_boosting(biden_models$boosting_s3, biden_split$test)

data_frame(shrinkage = c(0.1, 0.01, 0.001, 0.0001),
           model = biden_models[c("boosting_s1", "boosting_s2","boosting", "boosting_s3")],
           mse = c(boosting_s1_mse, boosting_s2_mse, boosting_mse, boosting_s3_mse )) %>%
  select(-model) %>%
  knitr::kable(caption = "Influence of shrinkage parameter on the test MSE",
               col.names = c("shrinkage", "MSEs"))
```

Table 1: Influence of shrinkage parameter on the test MSE

shrinkage	MSEs
0.100	420
0.010	407
0.001	400
0.000	444

Test MSE of the boosting model with default setting is 399.716, which is only slightly smaller than those of previous models. In order to test the influence of shrinkage parameter on the performance of the model, we fitted three separate boosting models with shrinkage parameters 0.1, 0.01, 0.0001 respectively. As the table indicates, as the shrinkae values decreases, the MSE also decreases, meaning that the models have better performance. Since shrinkage parameter controls the rate that boosting learns, this corresponds to the fact that generally statistical learning approaches that learn slowlier tend to perform better.

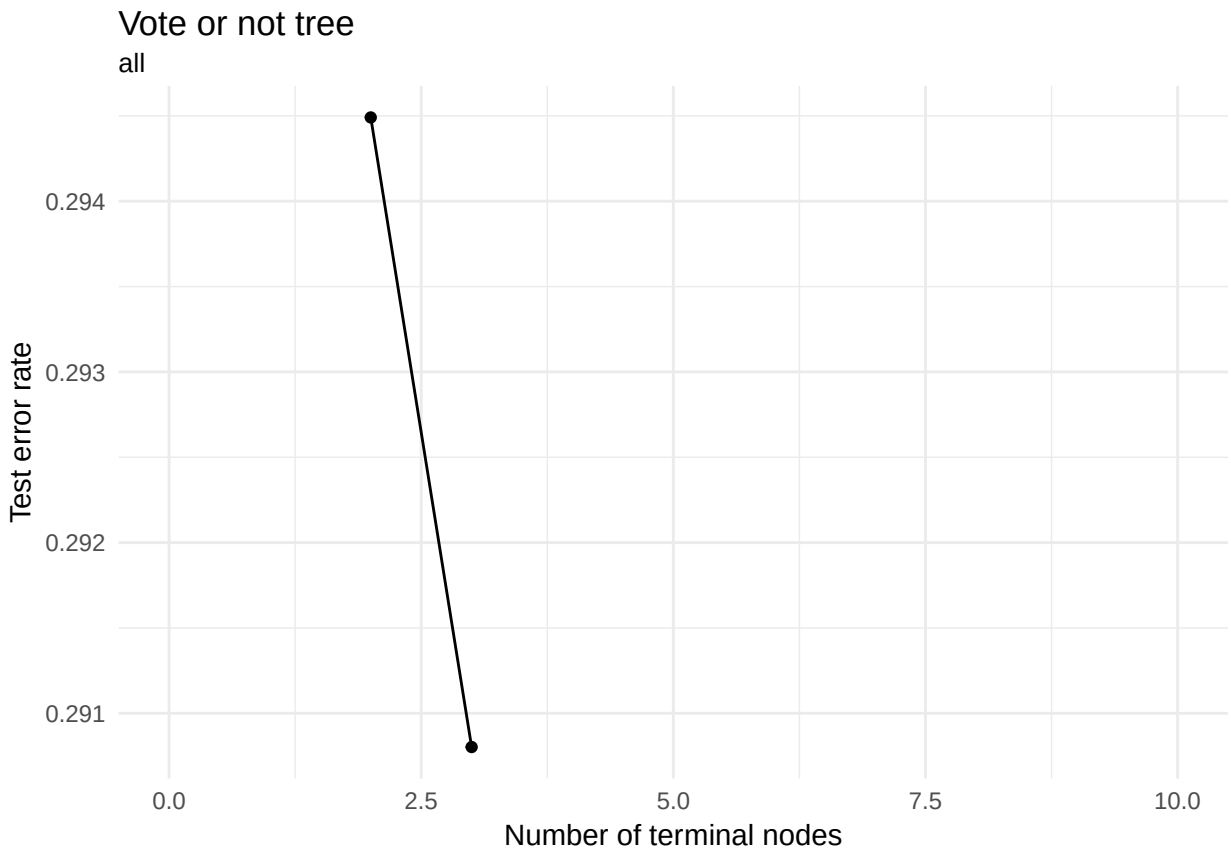
Part 2: Modeling voter turnout

1. Use cross-validation techniques and standard measures of model fit (e.g. test error rate, PRE, ROC curves/AUC) to compare and evaluate at least five tree-based models of voter turnout. Select the best model and interpret the results using whatever methods you see fit (graphs, tables, model fit statistics, predictions for hypothetical observations, etc.)

Decision tree

```
gss <- gss_data %>%
  as_tibble() %>%
  mutate(Voted = factor(vote96, levels = 0:1, labels = c("Not voted", "Voted")))

gss_split <- resample_partition(gss, c(test = 0.3, train = 0.7))
```

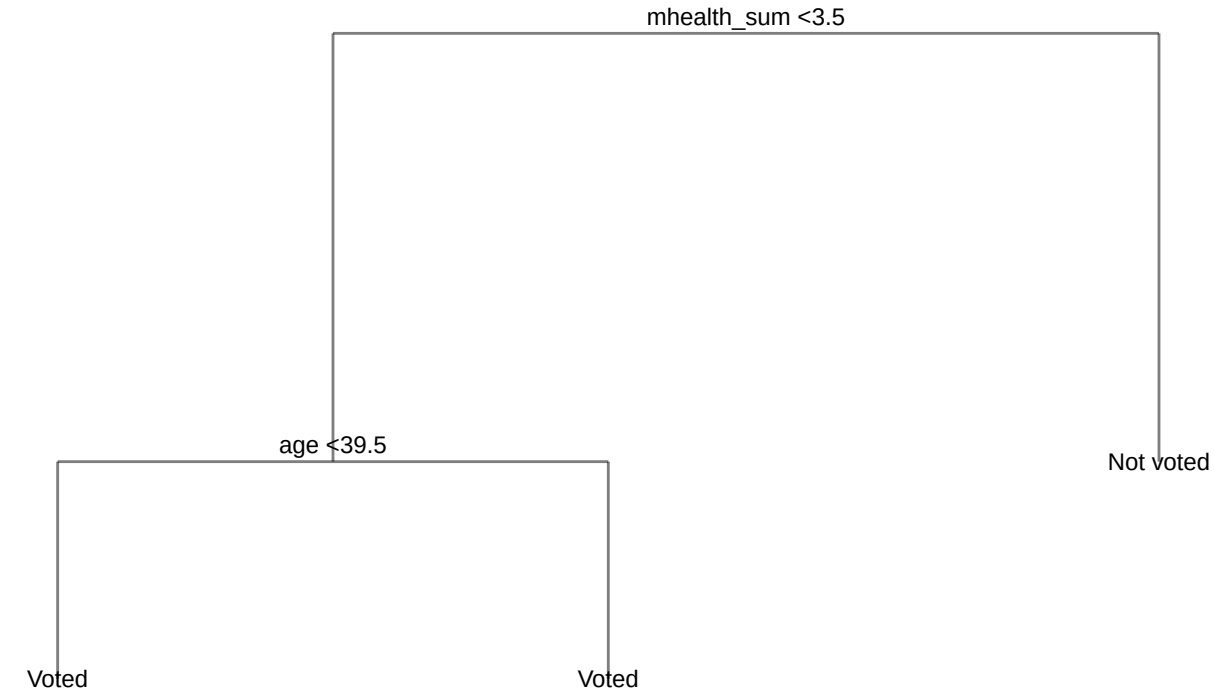
```r
# estimate model
gss_dt <- tree(Voted~ . - vote96, data = gss_split$train , control = tree.control(nobs = nrow(gss_split$

# generate 10-fold CV trees
gss_cv <- as.data.frame(gss_split$train)  %>%
  na.omit() %>%
  crossv_kfold(k = 10) %>%
  mutate(tree = map(train, ~ tree(Voted~ . - vote96, data = . ,
     control = tree.control(nobs = nrow(gss_split$train),
                            mindev = .001))))

# calculate each possible prune result for each fold
gss_cv <- expand.grid(gss_cv$.id,
                      seq(from = 2, to = ceiling(length(mod$frame$yval) / 2))) %>%
  as_tibble() %>%
  mutate(Var2 = as.numeric(Var2)) %>%
  rename(.id = Var1,
         k = Var2) %>%
  left_join(gss_cv) %>%
  mutate(prune = map2(tree, k, ~ prune.misclass(.x, best = .y)),
         mse = map2_dbl(prune, test, err.rate.tree))

gss_cv %>%
  group_by(k) %>%
  summarize(test_mse = mean(mse),
            sd = sd(mse, na.rm = TRUE)) %>%
  ggplot(aes(k, test_mse)) +
  geom_point() +
  geom_line() +
  xlim(0, 10)+
  labs(title = "Vote or not tree",
       subtitle = "all",
       x = "Number of terminal nodes",
       y = "Test error rate")
```




```
#optimal number of terminal nodes is 3

# plot pruned tree
mod <- prune.tree(gss_dt, best = 3)

tree_data <- dendro_data(mod)
ggplot(segment(tree_data)) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend),
               alpha = 0.5) +
  geom_text(data = label(tree_data),
            aes(x = x, y = y, label = label_full), vjust = -0.5, size = 3) +
  geom_text(data = leaf_label(tree_data),
            aes(x = x, y = y, label = label), vjust = 0.5, size = 3) +
  theme_dendro() +
  labs(title = "Voter turnout",
       subtitle = "all predictors")
```

Voter turnout

all predictors




```
gss_dt_err <- err.rate.tree (mod, gss_split$test)

pred <- predict(mod, newdata = as_tibble(gss_split$test), type = "class")
roc_dt <- roc(response = as.numeric(as_tibble(gss_split$test)$Voted), predictor =as.numeric(pred))
auc_dt <- auc(roc_dt)
```

Performance of this decision tree model Test error rate: `R gss_dt_err` AUC: `R auc_dt`

Bagging

```
gss_train <- na.omit(as_tibble(gss_split$train))
gss_test <- na.omit(as_tibble(gss_split$test))

gss_bag <- randomForest(Voted ~ .-vote96, data = gss_train,
                            mtry = 7, ntree = 500)

gss_bag_err <- err.rate.tree (gss_bag, gss_test)
pred <- predict(gss_bag, newdata = gss_test, type = "class")
roc_bag <- roc(response = as.numeric(gss_test$Voted), predictor =as.numeric(pred))
auc_bag <- auc(roc_bag)

# data_frame(var = rownames(importance(biden_bag)),
#            MeanDecreaseGini = importance(biden_bag)[,1]) %>%
#   mutate(var = fct_reorder(var, MeanDecreaseGini, fun = median)) %>%
#   ggplot(aes(var, MeanDecreaseGini)) +
#   geom_point() +
#   coord_flip() +
#   labs(title = "Predicting Biden feeling thermometer score",
```

```
#         subtitle = "Bagging",
#         x = NULL,
#         y = "Average decrease in the Gini Index")
```

Performance of this bagging model Test error rate: `gss_bag_err` AUC: `R auc_bag`

Random forest

```
gss_rf <- randomForest(Voted ~ .-vote96, data = gss_train, ntree = 500)

gss_rf_err <- err.rate.tree (gss_rf, gss_test)
pred <- predict(gss_rf, newdata = gss_test, type = "class")
roc_rf <- roc(response = as.numeric(gss_test$Voted), predictor =as.numeric(pred))
auc_rf <- auc(roc_rf)
```

Performance of this random forest model Test error rate: `gss_rf_err` AUC: `R auc_rf`

Boosting

```
gss_boosting1 <- gbm(vote96 ~ .-Voted, data = gss_train, n.trees = 10000, interaction.depth = 1, shrinka
```

```
## Distribution not specified, assuming bernoulli ...
```

```
pred <- predict(gss_boosting1, newdata = gss_test, n.trees = 10000)

gss_b1_err <- mean((pred - gss_test$vote96)**2)
#pred <- predict(gss_boosting1, newdata = gss_test, type = "class")
roc_b1 <- roc(response = gss_test$vote96, predictor =pred)
auc_b1 <- auc(roc_b1)
```

Boosting

```
gss_boosting2 <-  gbm(vote96 ~ .-Voted, data = gss_train, n.trees = 10000, interaction.depth = 4, shrink
```

```
## Distribution not specified, assuming bernoulli ...
```

```
pred <- predict(gss_boosting2, newdata = gss_test, n.trees = 10000)

gss_b2_err <- mean((pred - gss_test$vote96)**2)
roc_b2 <- roc(response = gss_test$vote96, predictor =pred)
auc_b2 <- auc(roc_b2)
```

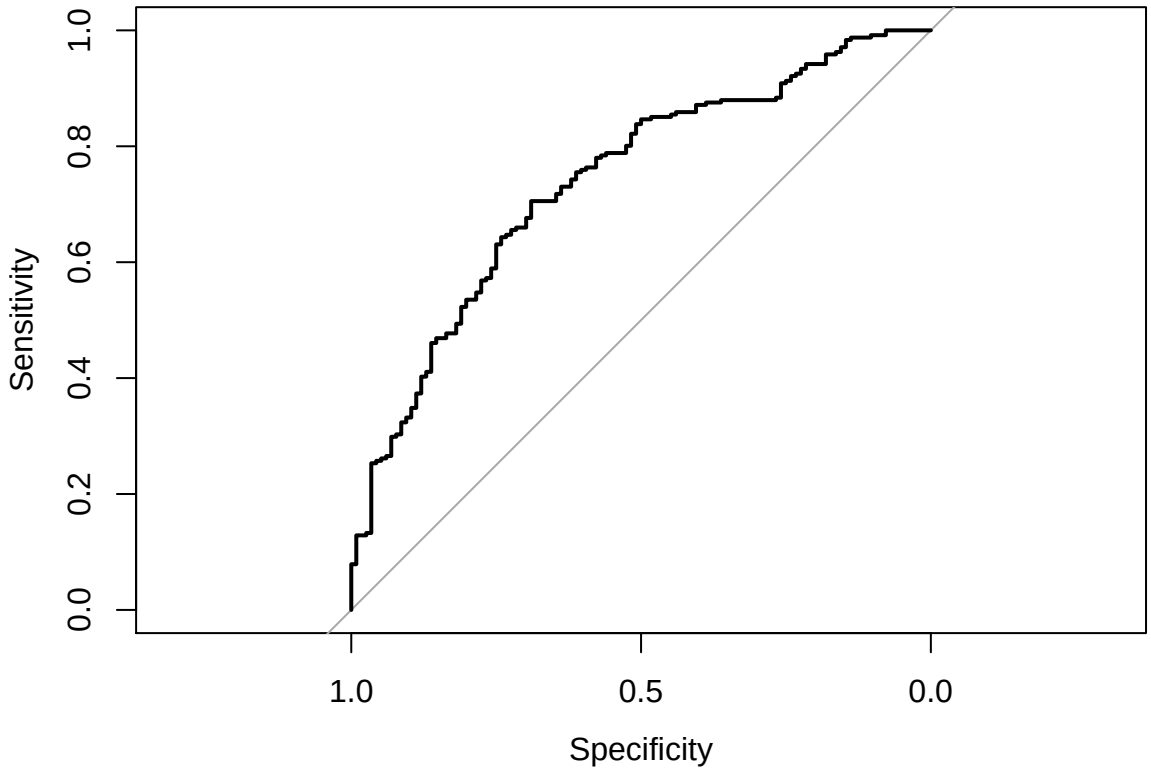
```
data_frame(model = c("Decision tree", "Bagging", "Random forest", "Boosting1", "Boosting2"),
           error_rate = c(gss_dt_err, gss_bag_err, gss_rf_err, gss_b1_err,gss_b2_err),
           auc = c(auc_dt, auc_bag, auc_rf, auc_b1, auc_b2) )%>%
  knitr::kable(caption = "Comparison between five tree-based models",
               col.names = c("Model", "test error rate", "AUC"))
```

Table 2: Comparison between five tree-based models

Model	test error rate	AUC
Decision tree	0.348	0.537

Model	test error rate	AUC
Bagging	0.294	0.637
Random forest	0.297	0.615
Boosting1	1.020	0.739
Boosting2	0.549	0.744

```
plot(roc_b1)
```



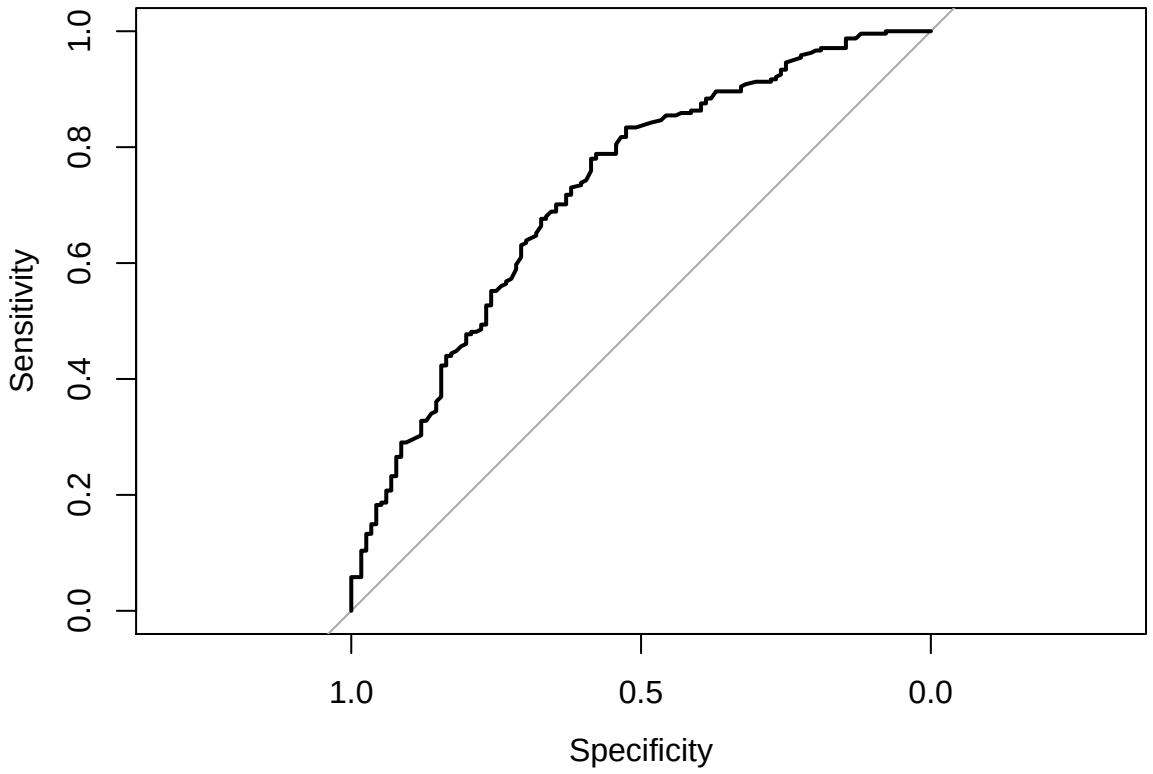
2. Use cross-validation techniques and standard measures of model fit (e.g. test error rate, PRE, ROC curves/AUC) to compare and evaluate at least five SVM models of voter turnout. Select the best model and interpret the results using whatever methods you see fit (graphs, tables, model fit statistics, predictions for hypothetical observations, etc.)

```
set.seed(1234)
#linear kernel with age + educ
gss_tune <- tune(svm, Voted ~ age + educ, data = gss_train,
                     kernel = "linear",
                     range = list(cost = c(.001, .01, .1, 1, 5, 10, 100)))
#summary(gss_tune)
gss_best <- gss_tune$best.model
summary(gss_best)
```

```
##
## Call:
## best.tune(method = svm, train.x = Voted ~ age + educ, data = gss_train,
```

```
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##       gamma:  0.5
##
## Number of Support Vectors:  523
##
##  ( 262 261 )
##
##
## Number of Classes:  2
##
## Levels:
##  Not voted Voted
```

```
# get predictions for test set
fitted <- predict(gss_best, gss_test, decision.values = TRUE) %>%
  attributes

roc_1 <- roc(gss_test$Voted, fitted$decision.values)
plot(roc_1)
```



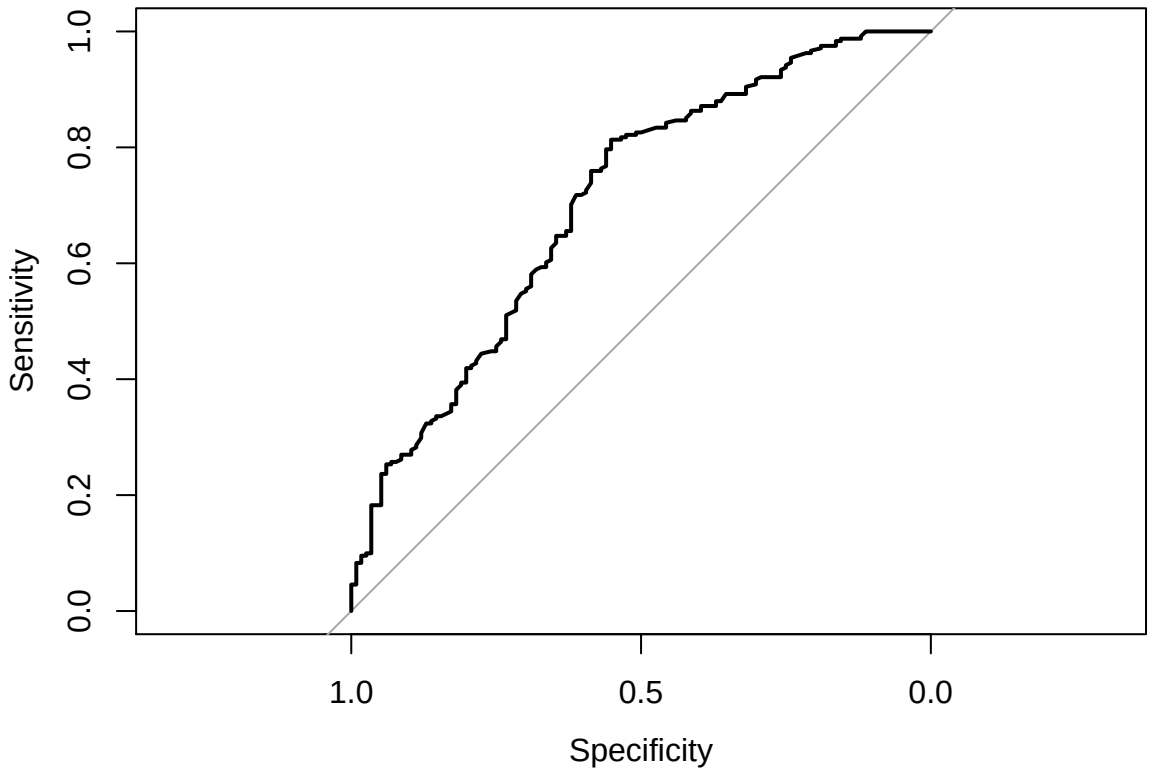

```
auc_1 <- auc(roc_1)

# sum_err_1 <- mean(fitted != actual, na.rm = TRUE))
```

```r
# round(fitted$decision.values)+1
```

```r
set.seed(1234)
#polynomial kernel with age+ educ
gss_poly_tune <- tune(svm, Voted ~ age + educ, data = gss_train,
                      kernel = "polynomial",
                      range = list(cost = c(.001, .01, .1, 1, 5, 10, 100)))
#summary(gss_poly_tune)
gss_poly_best <- gss_poly_tune$best.model
summary(gss_poly_best)
```

```
##
## Call:
## best.tune(method = svm, train.x = Voted ~ age + educ, data = gss_train,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
##     kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  5
##      degree:  3
##       gamma:  0.5
##      coef.0:  0
##
## Number of Support Vectors:  508
##
##  ( 255 253 )
##
##
## Number of Classes:  2
##
## Levels:
##  Not voted Voted
```

```r
# get predictions for test set
fitted <- predict(gss_poly_best, gss_test, decision.values = TRUE) %>%
  attributes

roc_2 <- roc(gss_test$Voted, fitted$decision.values)
plot(roc_2)
```



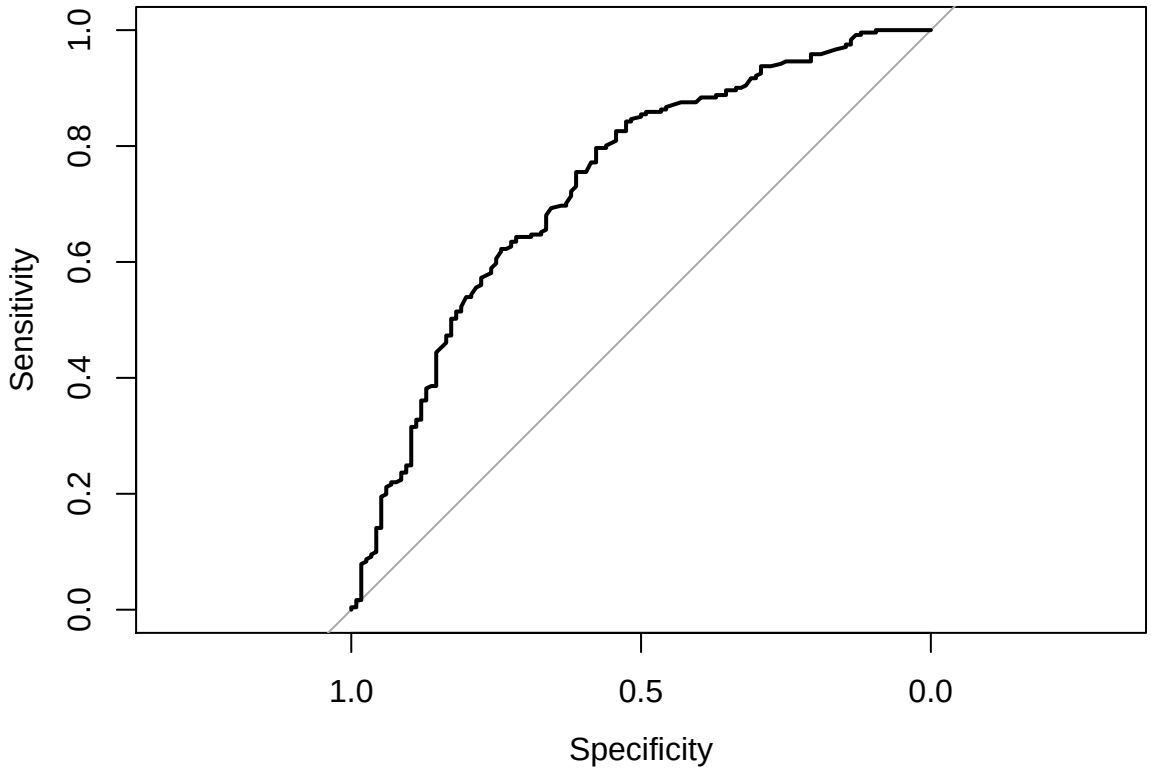

```
auc_2 <- auc(roc_2)
```

```
set.seed(1234)
#radical kernel with age+ educ
gss_rad_tune <- tune(svm, Voted ~ age + educ, data = gss_train,
                     kernel = "radial",
                     range = list(cost = c(.001, .01, .1, 1, 5, 10, 100)))

gss_rad_best <- gss_rad_tune$best.model
summary(gss_rad_best)
```

```
##
## Call:
## best.tune(method = svm, train.x = Voted ~ age + educ, data = gss_train,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  0.5
##
## Number of Support Vectors:  496
##
##  ( 260 236 )
##
##
## Number of Classes:  2
```

```
##
## Levels:
##  Not voted Voted
```

```
# get predictions for test set
fitted <- predict(gss_rad_best, gss_test, decision.values = TRUE) %>%
  attributes

roc_3 <- roc(gss_test$Voted, fitted$decision.values)
plot(roc_3)
```



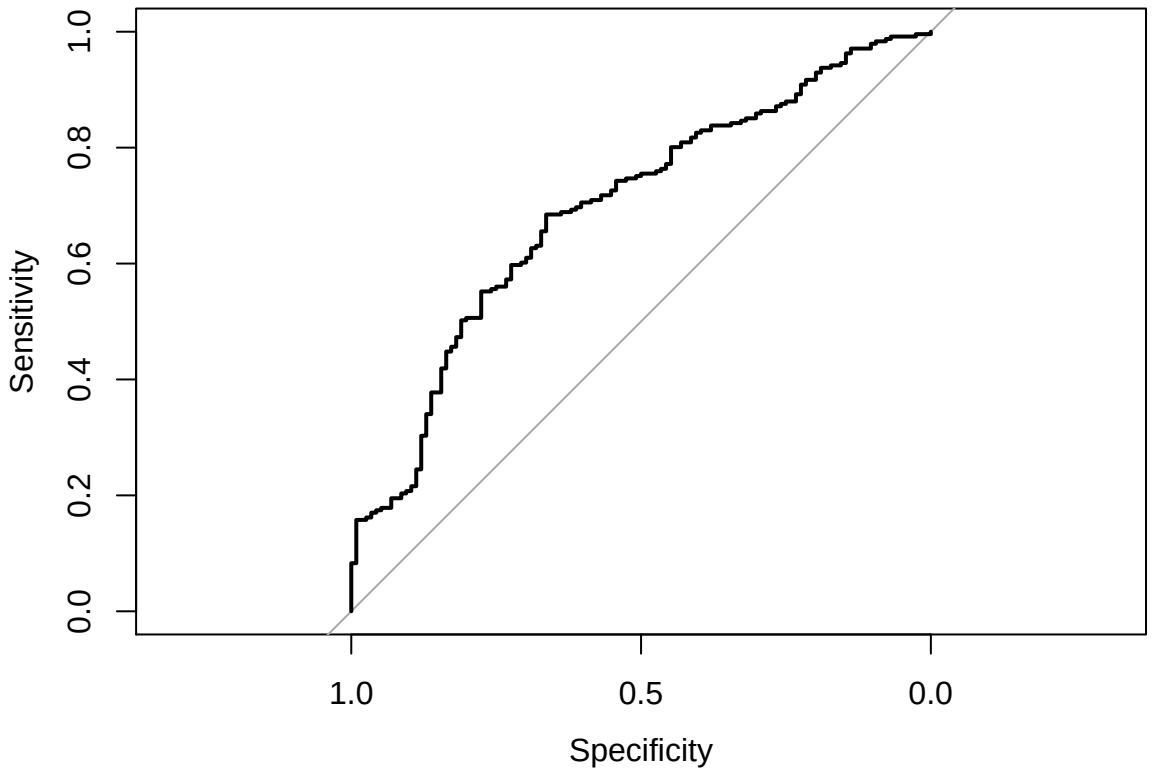

```
auc_3 <- auc(roc_3)
```

```
set.seed(1234)
#polynomial kernel with age+ educ
gss_poly_tune2 <- tune(svm, Voted ~ .-vote96, data = gss_train,
                       kernel = "polynomial",
                       range = list(cost = c(.001, .01, .1, 1, 5, 10, 100)))
#summary(gss_poly_tune)
gss_poly_best2 <- gss_poly_tune2$best.model
summary(gss_poly_best2)
```

```
##
## Call:
## best.tune(method = svm, train.x = Voted ~ . - vote96, data = gss_train,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
##     kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
```

```
##  SVM-Kernel:  polynomial
##        cost:  5
##      degree:  3
##       gamma:  0.143
##      coef.0:  0
##
## Number of Support Vectors:  482
##
##  ( 251 231 )
##
##
## Number of Classes:  2
##
## Levels:
##  Not voted Voted
```

```
# get predictions for test set
fitted <- predict(gss_poly_best2, gss_test, decision.values = TRUE) %>%
  attributes

roc_4 <- roc(gss_test$Voted, fitted$decision.values)
plot(roc_4)
```



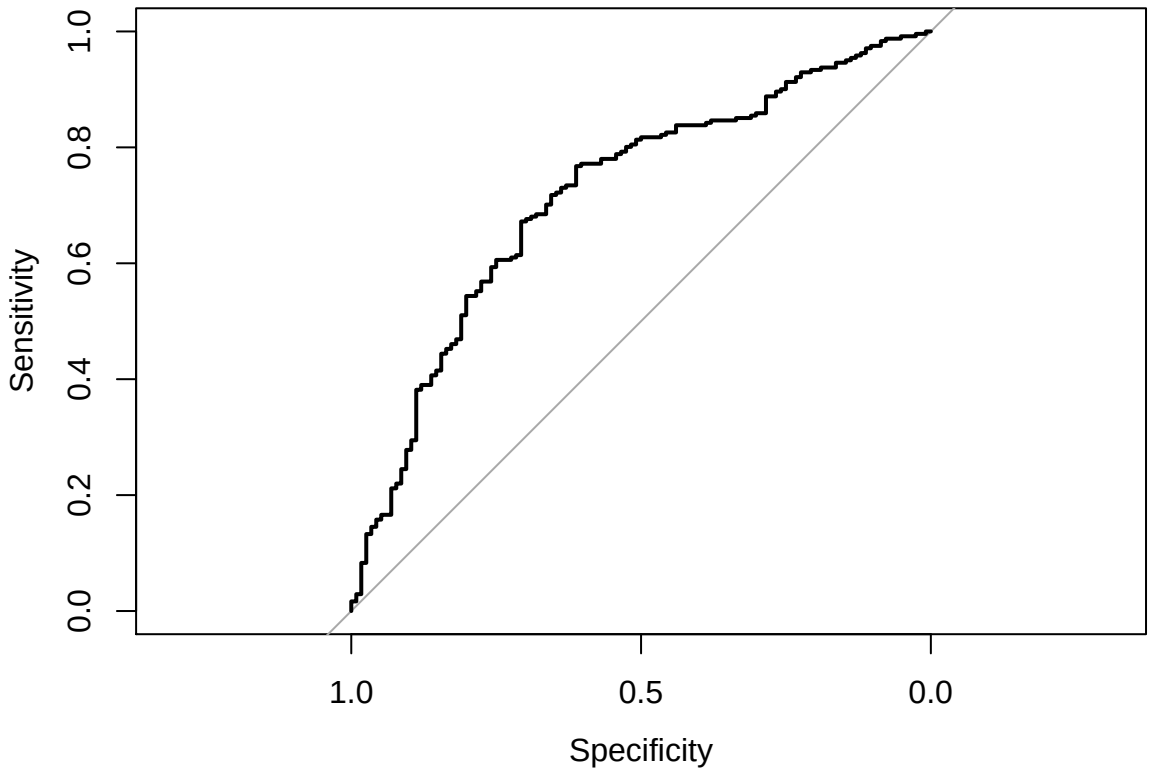

```
auc_4 <- auc(roc_4)
```

```
set.seed(1234)
#radical kernel with age+ educ
gss_rad_tune2 <- tune(svm, Voted ~ .-vote96, data = gss_train,
                      kernel = "radial",
                      range = list(cost = c(.001, .01, .1, 1, 5, 10, 100)))
```

```r
gss_rad_best2 <- gss_rad_tune2$best.model
summary(gss_rad_best2)
```

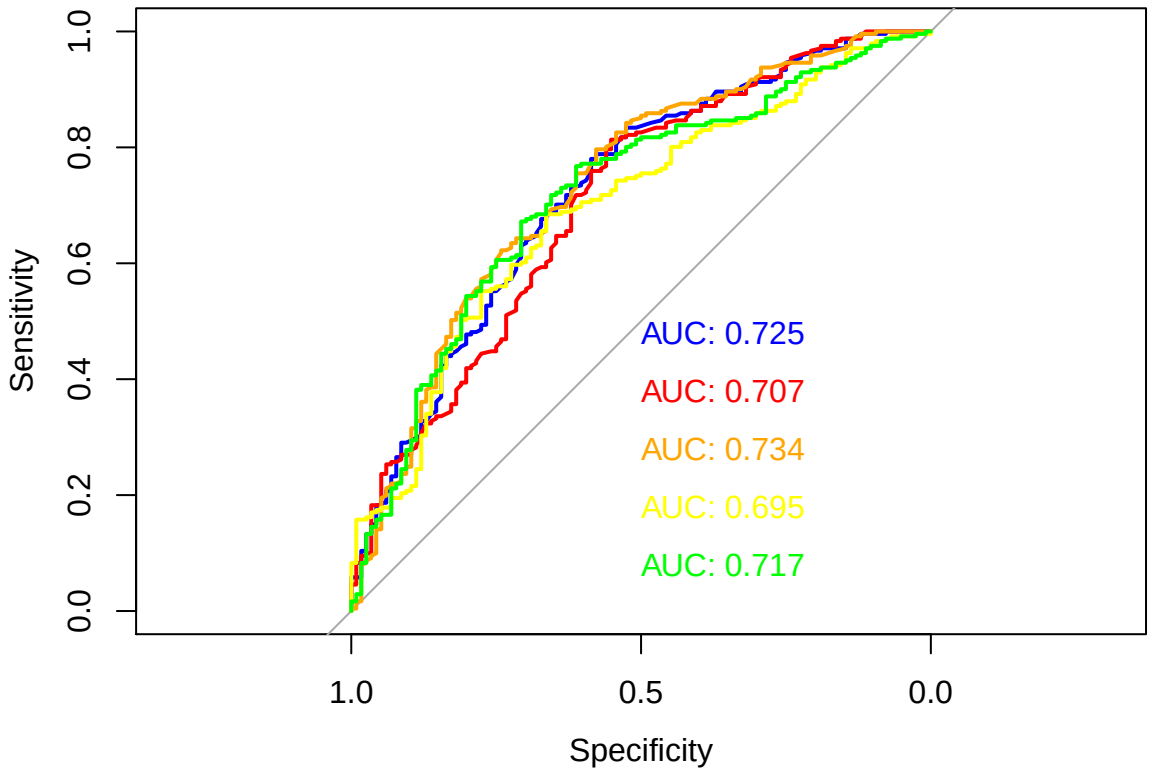
```
##
## Call:
## best.tune(method = svm, train.x = Voted ~ . - vote96, data = gss_train,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.143
##
## Number of Support Vectors:  504
##
##  ( 268 236 )
##
##
## Number of Classes:  2
##
## Levels:
##  Not voted Voted
```

```r
# get predictions for test set
fitted <- predict(gss_rad_best2, gss_test, decision.values = TRUE) %>%
  attributes

roc_5 <- roc(gss_test$Voted, fitted$decision.values)
plot(roc_5)
```




```
auc_5 <- auc(roc_5)
```

```
# data_frame(model = c("Linear with two variables", "Polynomial with two variables", "Radial with two v
#             error_rate = c(gss_dt_err, gss_bag_err, gss_rf_err, gss_b1_err,gss_b2_err),
#             auc = c(auc_1, auc_2, auc_3, auc_4, auc_5) )%>%
#   knitr::kable(caption = "Comparison between five svm models",
#                col.names = c("Model", "test error rate", "AUC"))

plot(roc_1, print.auc = TRUE, col = "blue")
plot(roc_2, print.auc = TRUE, col = "red", print.auc.y = .4, add = TRUE)
plot(roc_3, print.auc = TRUE, col = "orange", print.auc.y = .3, add = TRUE)
plot(roc_4, print.auc = TRUE, col = "yellow", print.auc.y = .2, add = TRUE)
plot(roc_5, print.auc = TRUE, col = "green", print.auc.y = .1, add = TRUE)
```



From question 1, we've realized that both age and education are very important parameters to predict voter turnout. Thus, we started training svm models using these two variables. After comparing AUCs of all five models, we can see that the one with radial kernel using all parameters is the best model.

Part 3: OJ Simpson

1. What is the relationship between race and belief of OJ Simpson's guilt? Develop a robust statistical learning model and use this model to explain the impact of an individual's race on their beliefs about OJ Simpson's guilt.

```
s_split <- resample_partition(s_data, c(test = 0.3, train = 0.7))
s_train <- as_tibble(s_split$train)
s_test <- as_tibble(s_split$test)
```

2. How can you predict whether individuals believe OJ Simpson to be guilty of these murders? Develop a robust statistical learning model to predict whether individuals believe OJ Simpson to be either probably guilty or probably not guilty and demonstrate the effectiveness of this model using methods we have discussed in class.