Assignment 4

Yihan Feng

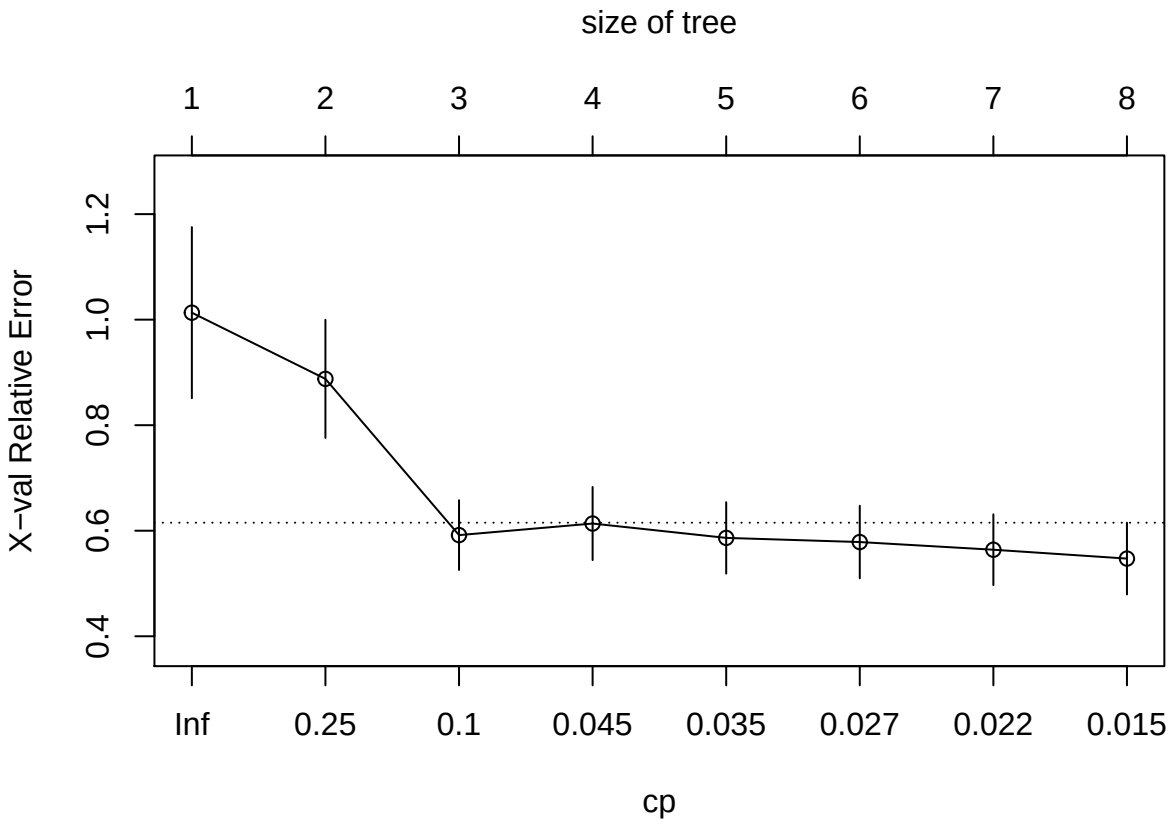
4/6/2021

Problem 1

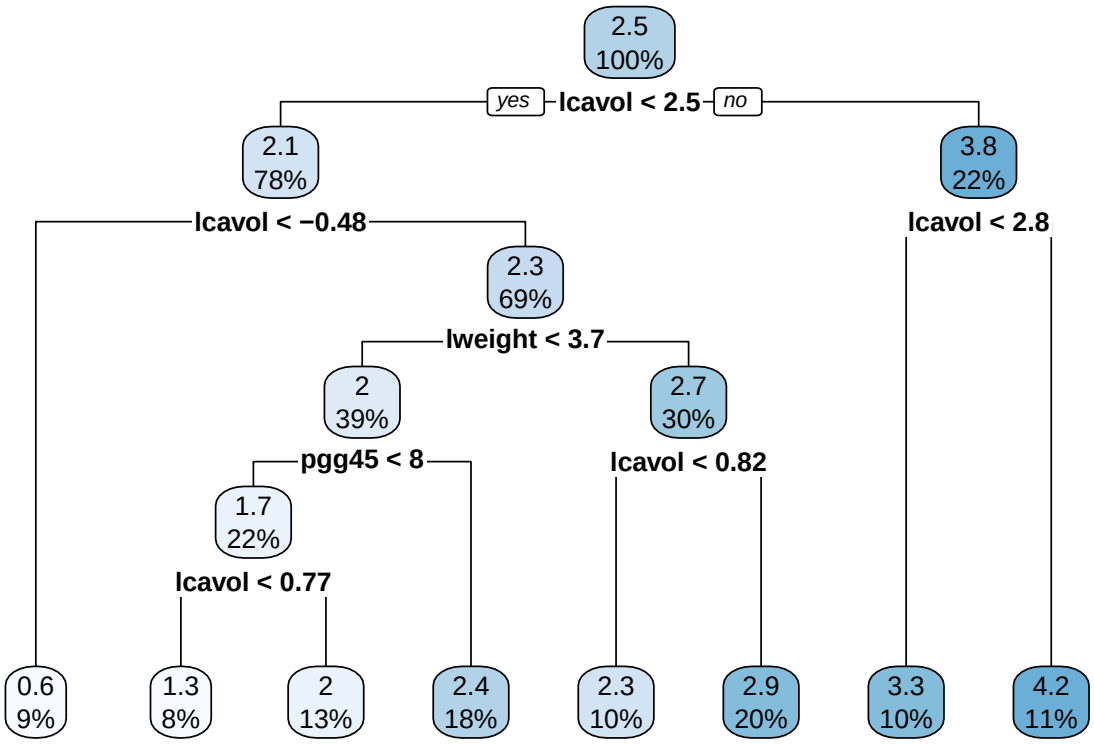
```
data("Prostate")
prostate.df = Prostate

trRows = createDataPartition(prostate.df$lpsa,
                             p = 0.75,
                             list = FALSE)
control1 = trainControl(method = "cv")
```

```
set.seed(1)
tree = rpart(lpsa ~ . ,
             prostate.df)

plotcp(tree)
```

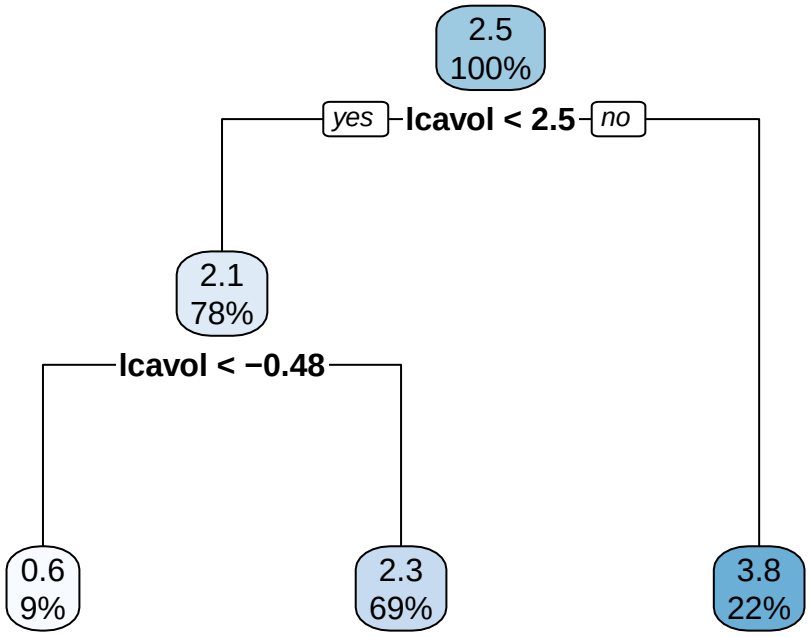
(a) Fit a regression tree with lpsa as the response and the other variables as predictors. Use cross-validation to determine the optimal tree size. Which tree size corresponds to the lowest cross-validation error? Is this the same as the tree size obtained using the 1 SE rule?




```
rpart.plot(tree)
```




```
cpTable = tree$cptable
minErr = which.min(cpTable[,4])

tree.1se = prune(tree, cp = cpTable[cpTable[,4]<cpTable[minErr,4]+cpTable[minErr,5],1][1])
rpart.plot(tree.1se)
```

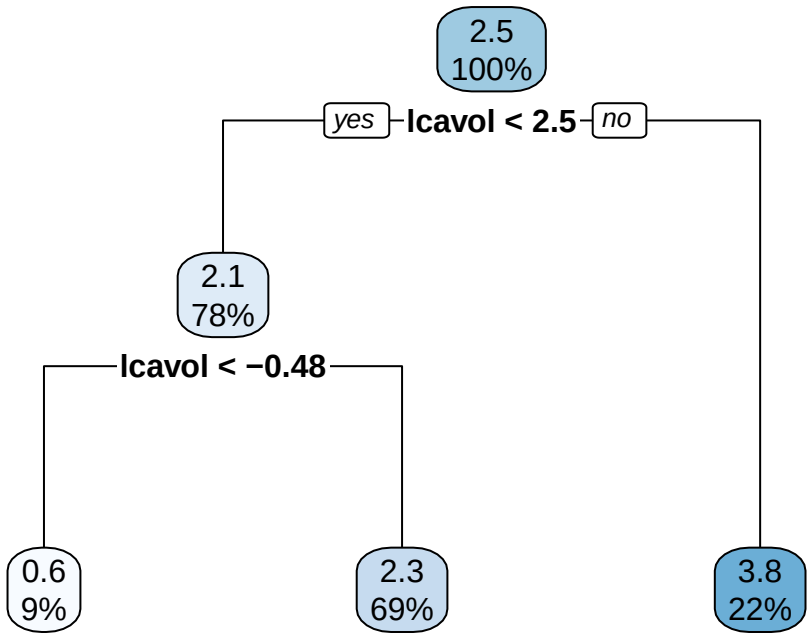



```
tree.min = prune(tree, cp = cpTable[minErr,1][1])
```

The tree size corresponds to the lowest cross-validation error is 8. The tree size corresponds to 1 SE rules is 3.

```
rpart.plot(tree.1se)
```

(b) Create a plot of the final tree you choose. Pick one of the terminal nodes, and interpret the information displayed.



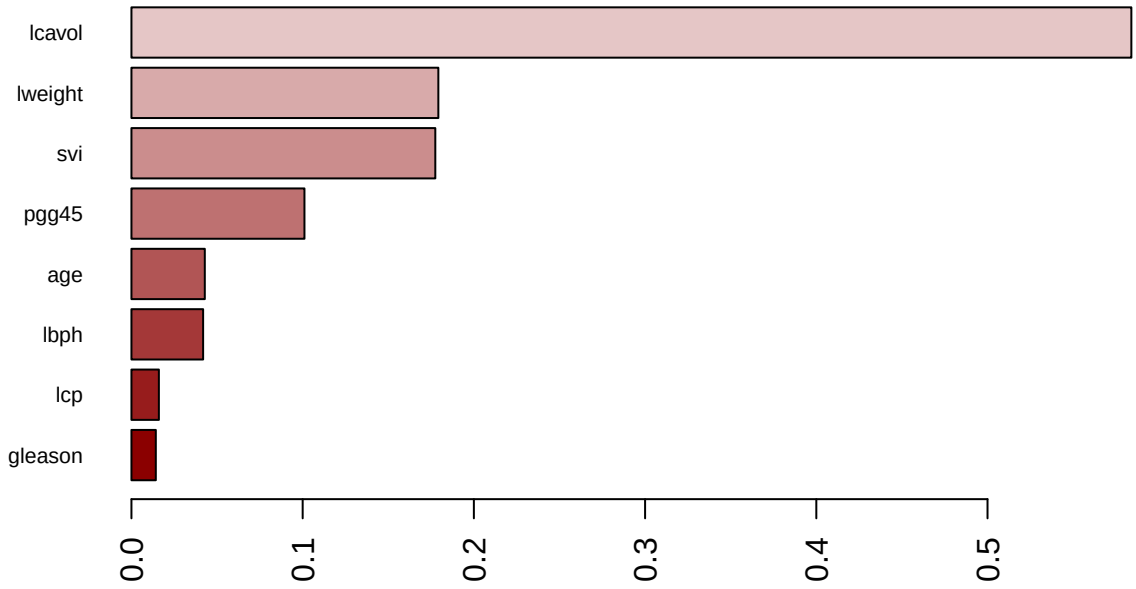
Interpretation: When the log lcavol is less than 2.4 and equal or greater than -0.48, the mean observation values of Lpsa in this terminal is 2.1. And this terminal nodes contains 50% of the training observations.

```
set.seed(1)

bagging.grid = expand.grid(mtry = 8,
                           splitrule = "variance",
                           min.node.size = 1:20)
bagging = train(lpsa ~ . ,
                prostate.df,
                subset = trRows,
                  method = "ranger",
                  tuneGrid = bagging.grid,
                  trControl = control1,
                  metric = "RMSE",
                  importance = "permutation")

bagging$result[which.min(bagging$results[,5]),]
```

(c) Perform bagging and report the variable importance.

```
##    mtry splitrule min.node.size      RMSE  Rsquared       MAE    RMSESD
## 16    8  variance            16 0.7223026 0.5724365 0.5918636 0.2168013
##    RsquaredSD     MAESD
## 16  0.1616386 0.1574604
```

```
barplot(sort(ranger::importance(bagging$finalModel), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(19))
```



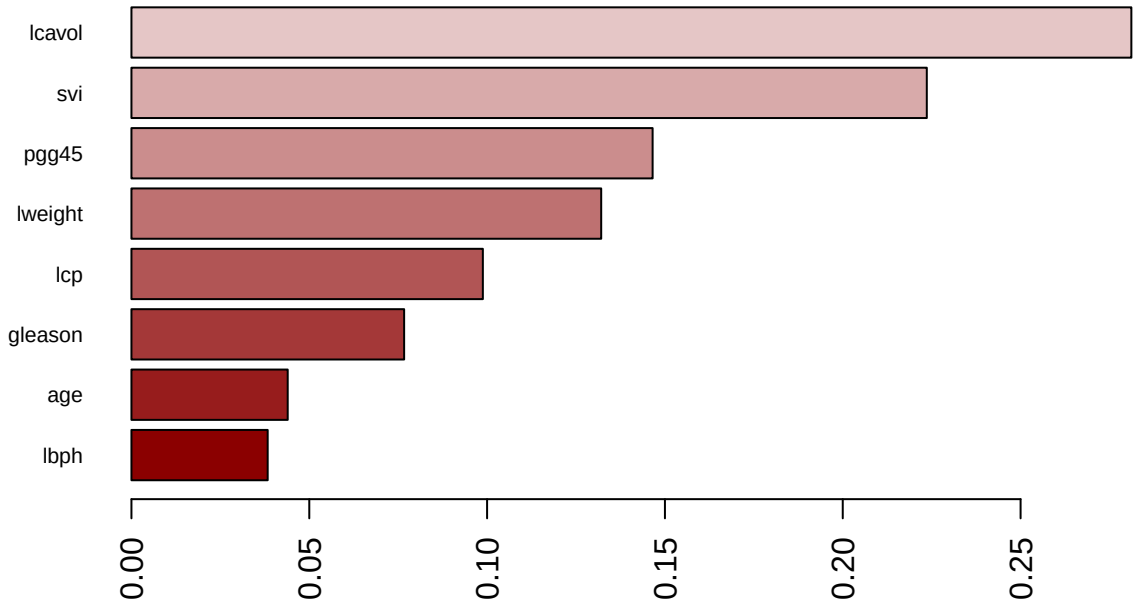
The importance from highest to lowest: lcavol, svi, lweight, lcp, pgg45, lbph, gleason, age

```r
set.seed(1)

rf.grid = expand.grid(mtry = 1:7,
                      splitrule = "variance",
                      min.node.size = 1:20)

rf = train(lpsa ~ . ,
              prostate.df,
              subset = trRows,
           method = "ranger",
           tuneGrid = rf.grid,
           trControl = control1,
           importance = "permutation")

rf$result[which.min(rf$results[,5]),]
```

(d) Perform random forest and report the variable importance.

```
##     mtry splitrule min.node.size      RMSE  Rsquared       MAE    RMSESD
## 135    7  variance            15 0.7163409 0.5827014 0.5861489 0.2110015
##     RsquaredSD     MAESD
## 135  0.1670867 0.1540447
```

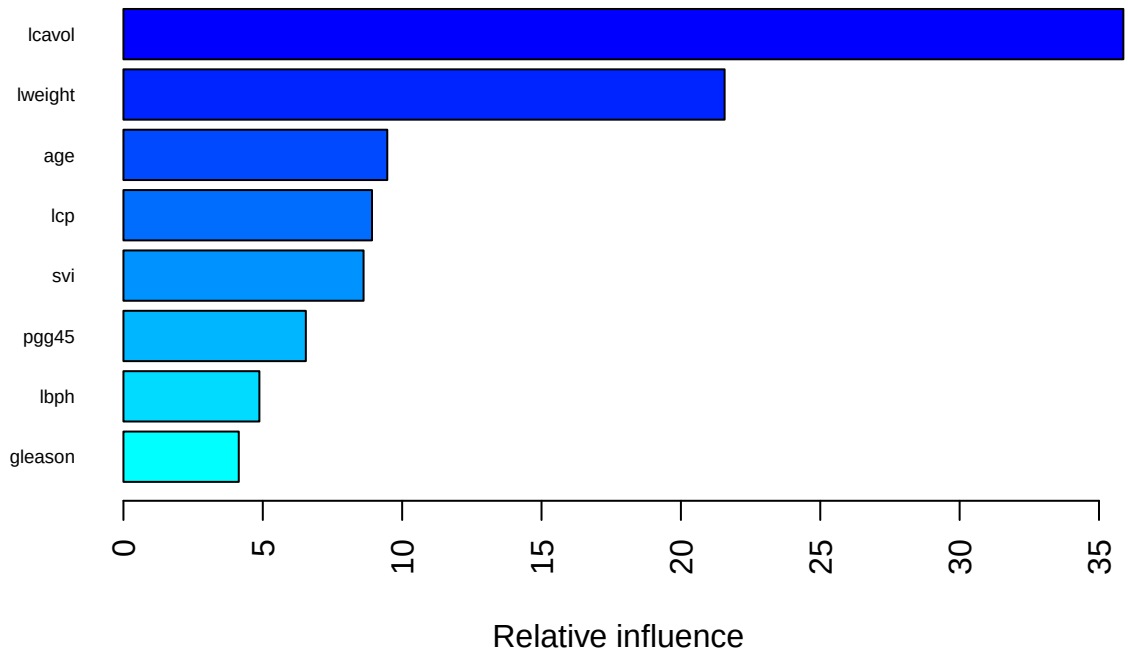
```r
barplot(sort(ranger::importance(rf$finalModel), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(19))
```



The importance from highest to lowest: lcavol, svi, lweight, lcp, pgg45, gleason, lbph, age

```
boosting.grid = expand.grid(n.trees = c(2000, 3000),
                            interaction.depth = 1:5,
                            shrinkage = seq(0.01, 0.05, len = 3),
                            n.minobsinnode = 1:10)

boosting = train(lpsa ~ . ,
                 prostate.df,
                 subset = trRows,
                    method = "gbm",
                    tuneGrid = boosting.grid,
                    trControl = control1,
                    verbose = FALSE)
summary(boosting$finalModel,
        las = 2,
        cBars = 19,
        cex.names = 0.6)
```

(e) Perform boosting and report the variable importance.




```
##             var   rel.inf
## lcavol   lcavol 35.874301
## lweight lweight 21.568129
## age         age  9.465646
```

```
## lcp         lcp  8.917328
## svi         svi  8.615708
## pgg45     pgg45  6.543921
## lbph       lbph  4.877417
## gleason gleason  4.137550
```

The importance from highest to lowest: lcavol, lweight, svi, age, lcp, lbph, pgg45, gleason

```
resample = resamples(list(bagging = bagging,
                          randomforest = rf,
                          boosting = boosting))
summary(resample)
```

(f) Which of the above models will you select to predict PSA level?

```
##
## Call:
## summary.resamples(object = resample)
##
## Models: bagging, randomforest, boosting
## Number of resamples: 10
##
## MAE
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## bagging      0.3984566 0.4866116 0.5079147 0.5746832 0.5934488 0.9340593    0
## randomforest 0.3516963 0.4823104 0.5042799 0.5580368 0.5906832 0.8662464    0
## boosting     0.4322930 0.5172649 0.5642106 0.5949657 0.6312477 0.8561852    0
##
## RMSE
##                   Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## bagging      0.4346588 0.5696529 0.6126163 0.7054637 0.7433606 1.139006    0
## randomforest 0.3983901 0.5522512 0.6696398 0.6868840 0.6980039 1.050708    0
## boosting     0.4653923 0.6404675 0.7632973 0.7488086 0.8654685 1.026401    0
##
## Rsquared
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## bagging      0.2834417 0.5377027 0.5718267 0.6041678 0.6440544 0.8715307    0
## randomforest 0.3801693 0.5259919 0.6376655 0.6403641 0.7720660 0.9006568    0
## boosting     0.4677919 0.4955857 0.6405875 0.6354862 0.7727994 0.7958831    0
```

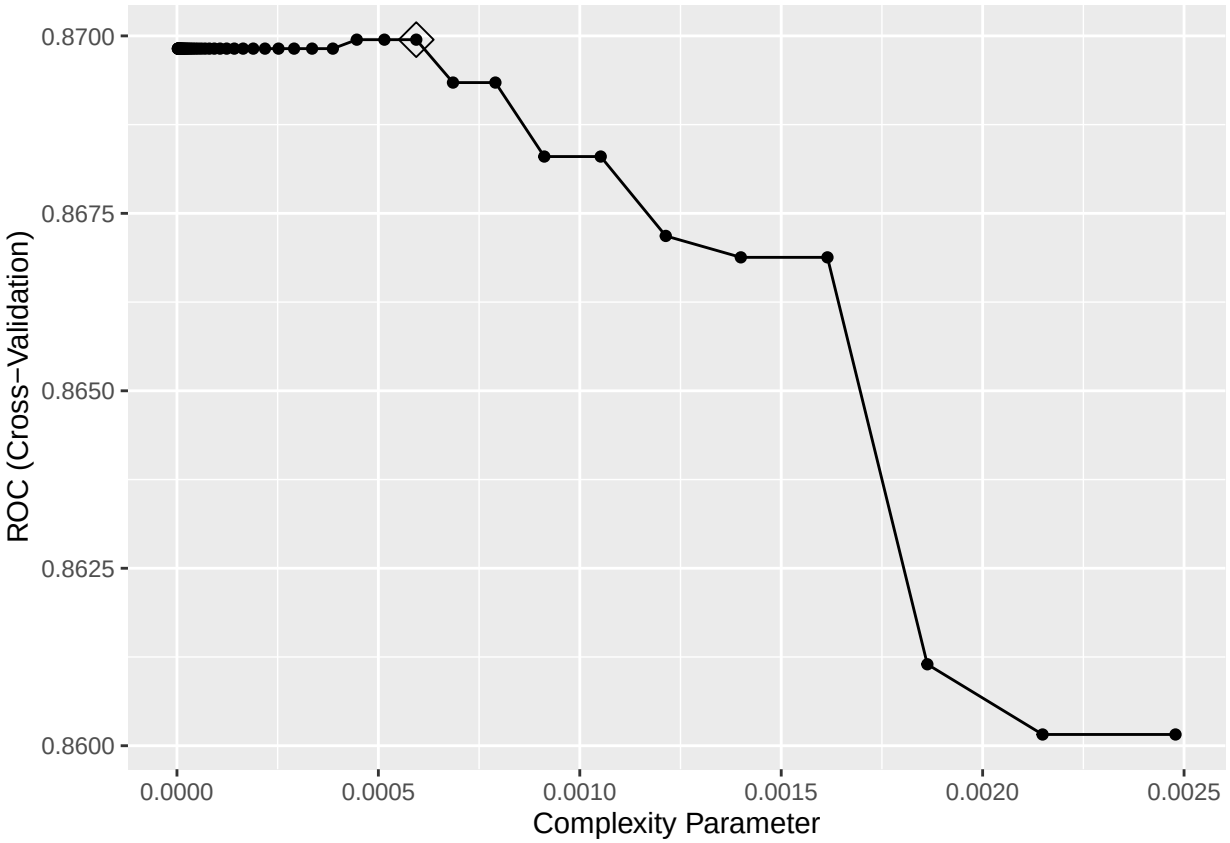
I would choose boosting method, as it has the lowest mean and median for MAE and RMSE.

Problem 2

```
data("OJ")
oj.df = OJ

trRows.cl = createDataPartition(oj.df$Purchase,
                                p = 0.75,
                                list = FALSE)

control = trainControl(method = "cv",
                       summaryFunction = twoClassSummary,
                       classProbs = TRUE)
```

```
set.seed(1)
cl.tree = train(Purchase ~ .,
                oj.df,
                method = "rpart",
                tuneGrid = data.frame(cp = exp(seq(-13, -6, len = 50))),
                trControl = control,
                metric = "ROC"
                )

ggplot(cl.tree, highlight = TRUE)
```

(a) Fit a classification tree to the training set, with Purchase as the response and the other variables as predictors. Use cross-validation to determine the tree size and create a plot of the final tree. Predict the response on the test data. What is the test classification error rate?



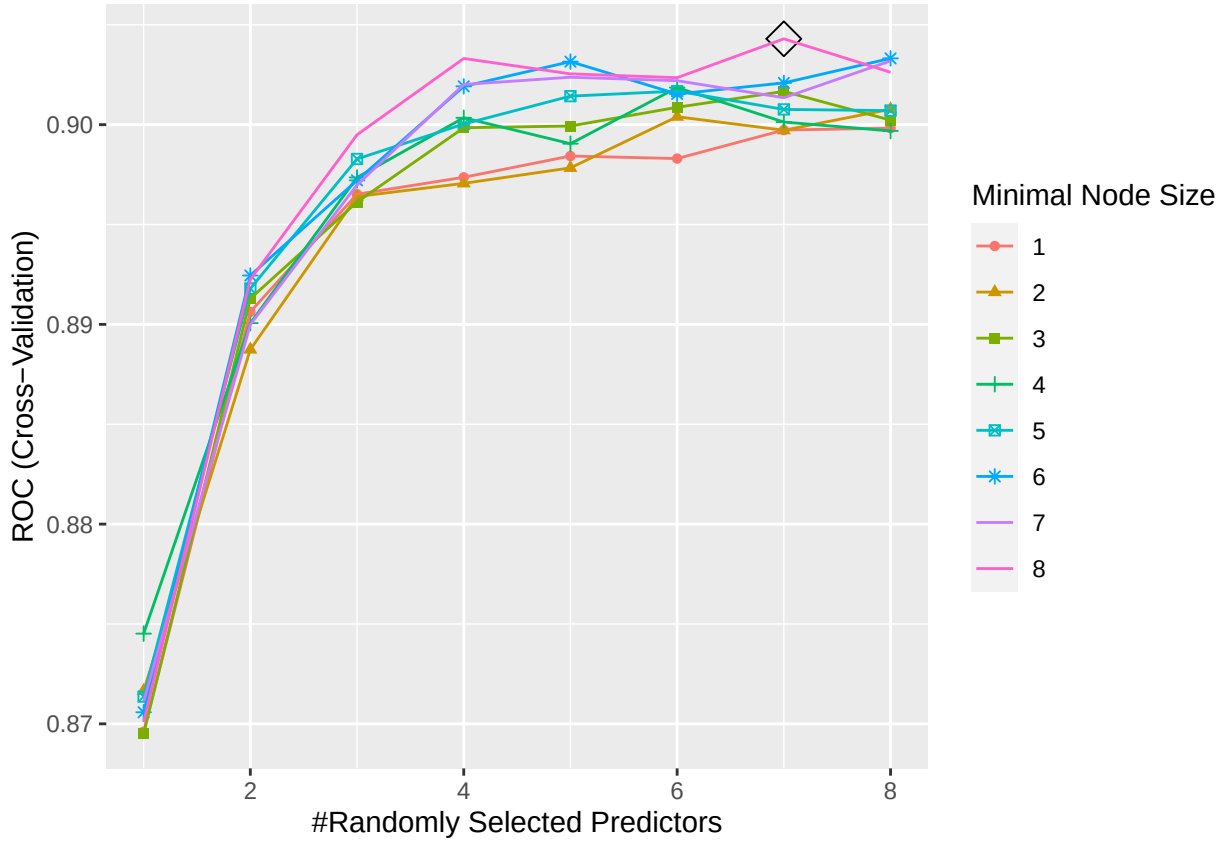

```
cl.tree.pred = predict(cl.tree, newdata = oj.df[-trRows.cl,])
cl.tree.er = mean(cl.tree.pred != oj.df$Purchase[-trRows.cl])
cl.tree.er
```

```
## [1] 0.1460674
```

The error rate is 14.607%.

```
set.seed(1)

cl.rf.grid = expand.grid(mtry = 1:8,
                         splitrule = "gini",
                         min.node.size = 1:8)

cl.rf = train(Purchase ~ .,
              oj.df,
              subset = trRows.cl,
              method = "ranger",
              tuneGrid = cl.rf.grid,
              metric = "ROC",
              trControl = control)

ggplot(cl.rf, highlight = TRUE)
```

(b) Perform random forest on the training set and report variable importance. What is the test error rate?



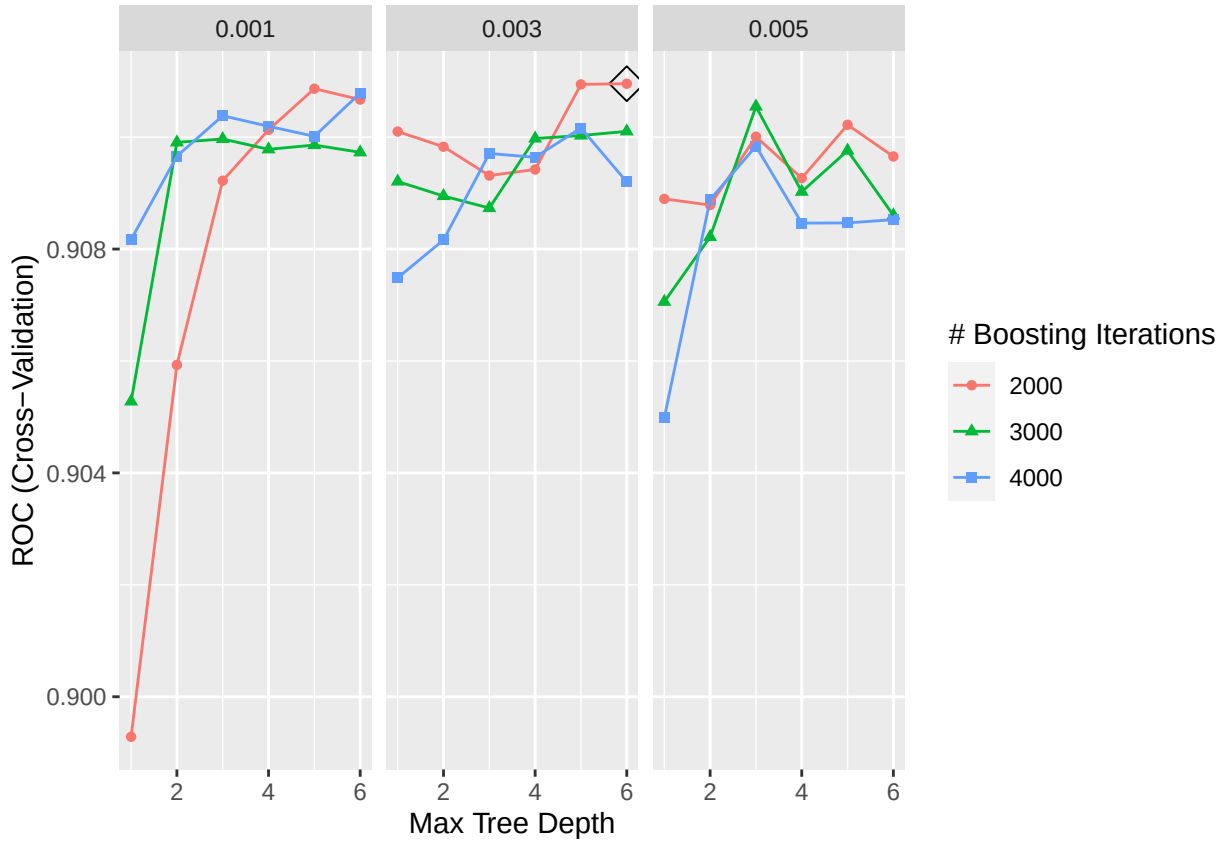

```
cl.rf.pred = predict(cl.rf, newdata = oj.df[-trRows.cl,])
cl.rf.er = mean(cl.rf.pred != oj.df$Purchase[-trRows.cl])
cl.rf.er
```

```
## [1] 0.2359551
```

The error rate is 23.596%.

```
set.seed(1)
cl.boosting.grid = expand.grid(n.trees = c(2000, 3000, 4000),
                               interaction.depth = 1:6,
                               shrinkage = c(0.001, 0.003, 0.005),
                               n.minobsinnode = 1)
cl.boosting = train(Purchase ~ .,
                    oj.df,
                    subset = trRows.cl,
                    tuneGrid = cl.boosting.grid,
                    trControl = control,
                    method = "gbm",
                    distribution = "adaboost",
                    metric = "ROC",
                    verbose = FALSE)

ggplot(cl.boosting, highlight = TRUE)
```

(c) Perform boosting on the training set and report variable importance. What is the test error rate?




```
cl.boosting.pred = predict(cl.boosting, newdata = oj.df[-trRows.cl,])
cl.boosting.er = mean(cl.boosting.pred != oj.df$Purchase[-trRows.cl])
cl.boosting.er
```

```
## [1] 0.2022472
```

The error rate is 20.225%.