Homework 4

Yiying Wu (yw3996)

R packages

```
library(tidyverse)
library(caret)
library(tidymodels)
library(rpart)
library(rpart.plot)
library(ranger)
library(gbm)
```

1. College data

In this exercise, we will build tree-based models using the College data (see "Col- lege.csv" in Homework 2). The response variable is the out-of-state tuition (Outstate). Partition the dataset into two parts: training data (80%) and test data (20%).

```
dat1<-read_csv("./data/College.csv")
dat1 <- na.omit(dat1)%>% select(-College)
```

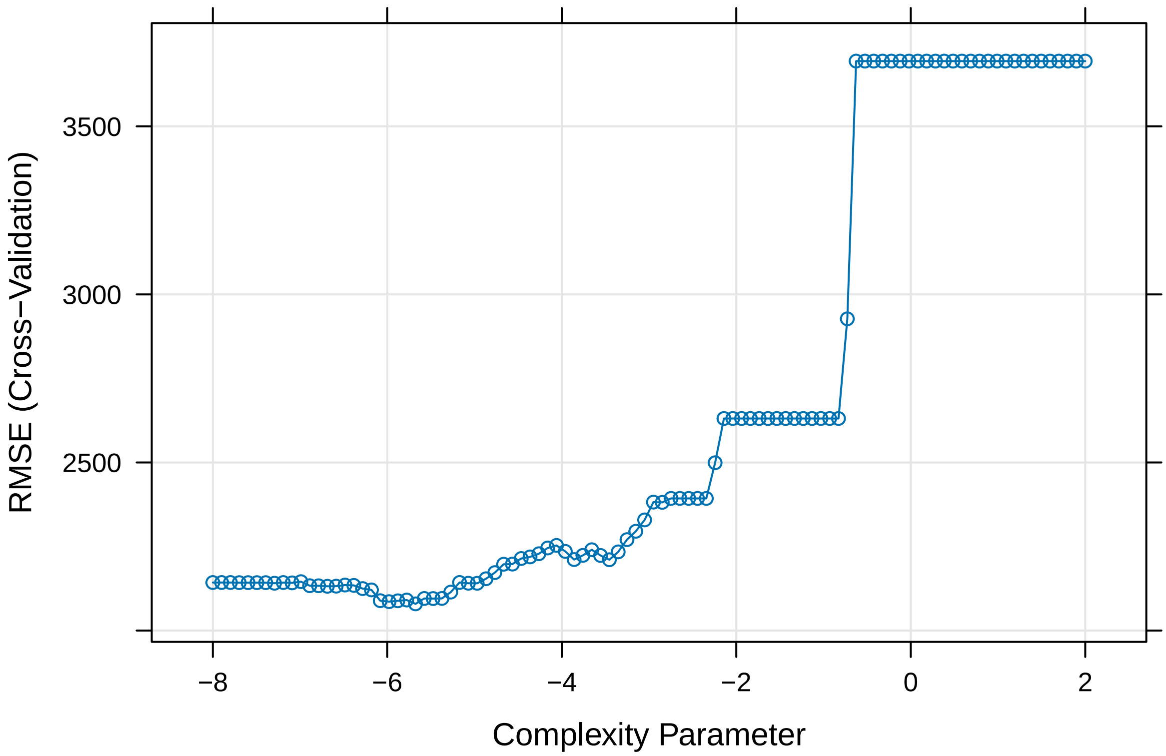
Partition the dataset into two parts: training data (80%) and test data (20%).

```
set.seed(1)
data_split1 <- initial_split(dat1, prop = 0.80)

# Extract the training and test data
training_data1 <- training(data_split1)
x_train1 <- training_data1 %>% select(-Outstate)
y_train1 <- training_data1$Outstate

testing_data1 <- testing(data_split1)
x_test1 <- testing_data1 %>% select(-Outstate)
y_test1 <- testing_data1$Outstate

# ctrl
ctrl1 <- trainControl(method = "cv", number = 10)
```

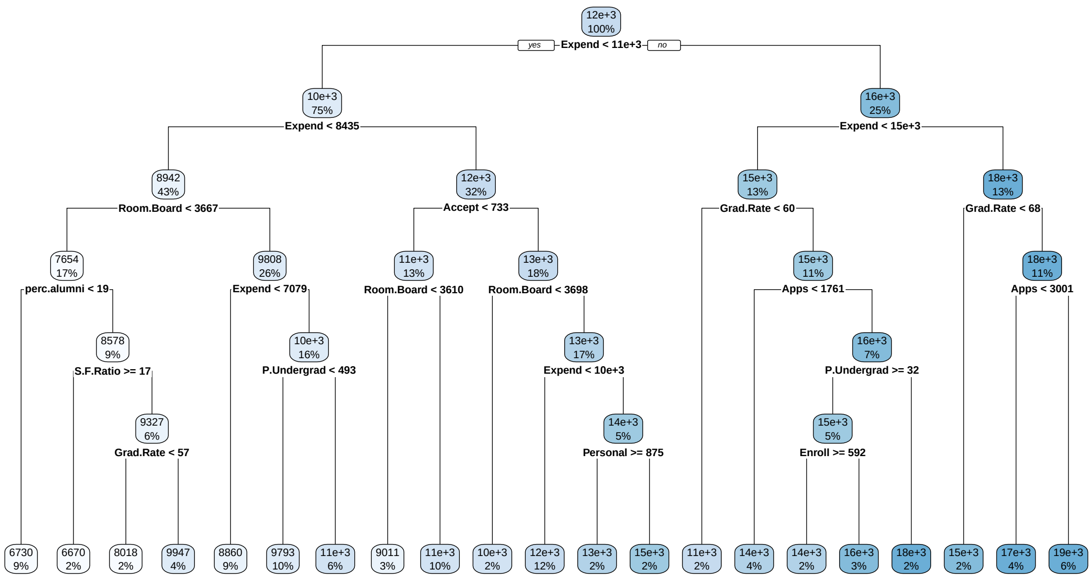
Outcome variable: Outstate

(a) Build a regression tree on the training data to predict the response. Create a plot of the tree.

```
set.seed(1)
rpart.fit <- train(Outstate ~ . ,
                   training_data1,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-8,2, length = 100))),
                   trControl = ctrl1)


plot(rpart.fit, xTrans = log)
```

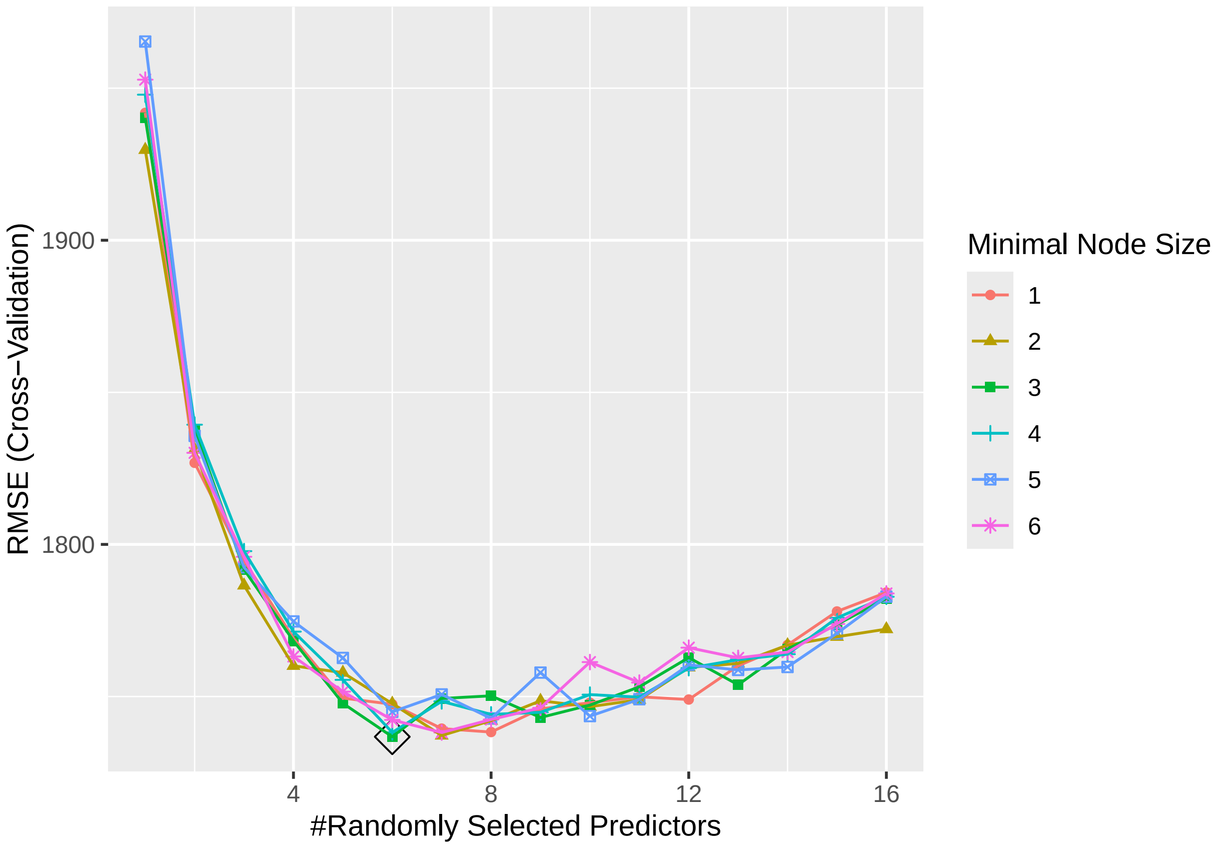



```
rpart.plot(rpart.fit$finalModel)
```



(b) Perform random forest on the training data. Report the variable importance and the test error.

```
rf.grid <- expand.grid(mtry = 1:16,
                       splitrule = "variance",
                       min.node.size = 1:6)
set.seed(1)
rf.fit <- train(Outstate ~ . ,
                training_data1,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl1)
ggplot(rf.fit, highlight = TRUE)
```

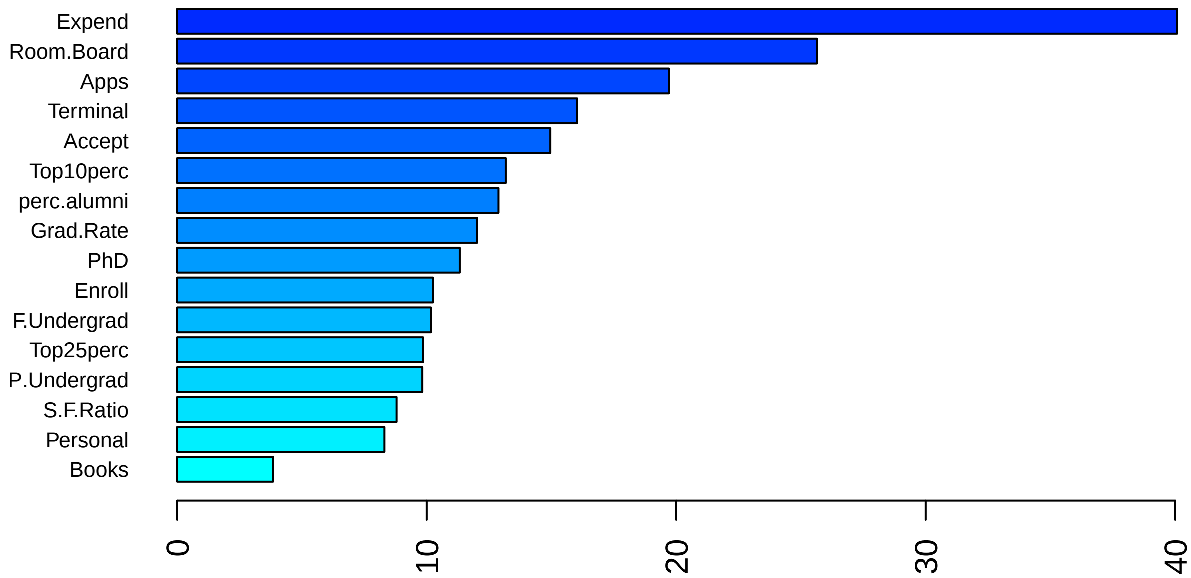



```
rf.fit$bestTune
```

```
##    mtry splitrule min.node.size
## 33    6  variance             3
```

variable importance

```
set.seed(1)
rf.final.per <- ranger(Outstate ~ . ,
                        training_data1,
                        mtry = rf.fit$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = rf.fit$bestTune[[3]],
                        importance = "permutation",
                        scale.permutation.importance = TRUE)
barplot(sort(ranger::importance(rf.final.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(19))
```



test error

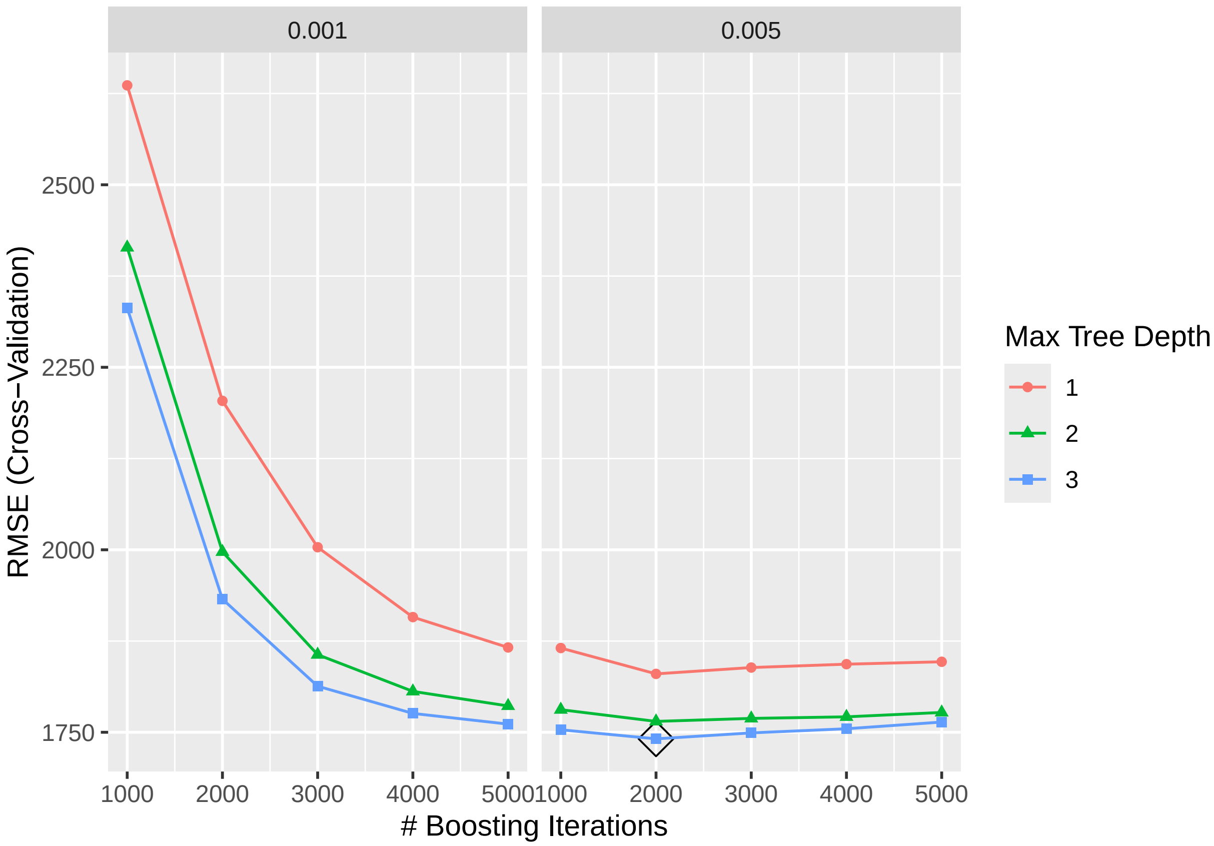
```
rf.predict <- predict(rf.fit, newdata = training_data1)
rf.RMSE <- RMSE(rf.predict, y_test1)
rf.RMSE
```

```
## [1] 5040.468
```

The RMSE for random forest is 5040.468.

(c) Perform boosting on the training data. Report the variable importance and the test error.

```
gbm.grid <- expand.grid(n.trees = c(1000, 2000, 3000, 4000, 5000),
                        interaction.depth = 1:3,
                        shrinkage = c(0.001, 0.005),
                        n.minobsinnode = c(1))

set.seed(1)
gbm.fit <- train(Outstate ~ . ,
                 training_data1,
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 trControl = ctrl1,
                 verbose = FALSE)
ggplot(gbm.fit, highlight = TRUE)
```

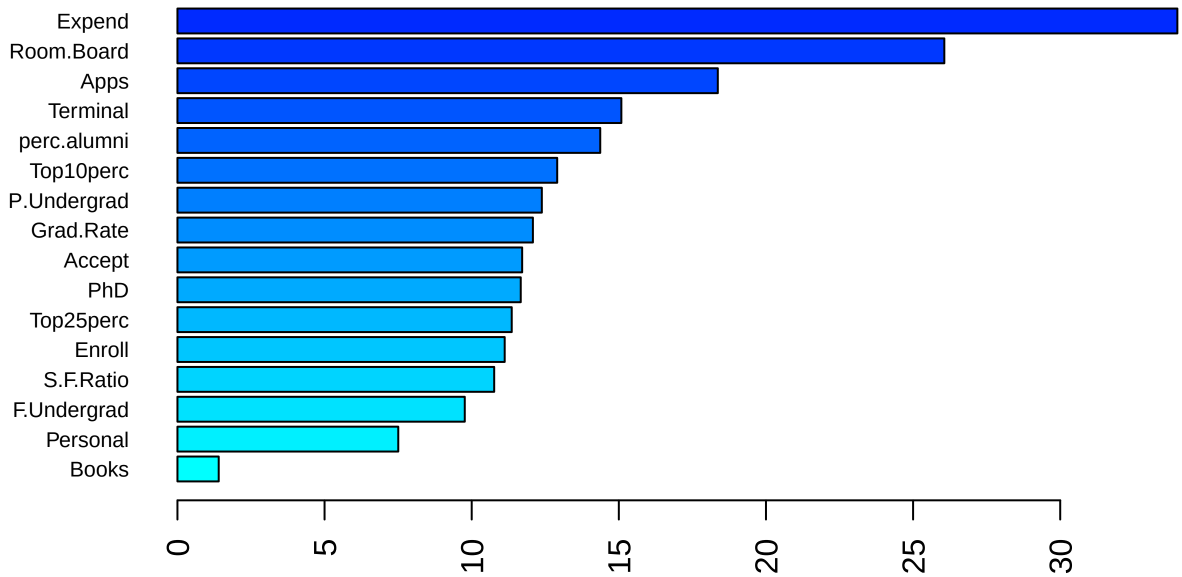



```
gbm.fit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 27    2000                 3     0.005              1
```

variable importance

```
set.seed(1)
gbm.final.per <- ranger(Outstate ~ . ,
                        training_data1,
                        n.trees = gbm.fit$bestTune[[1]],
                        splitrule = "variance",
                        interaction.depth = gbm.fit$bestTune[[2]],
                        shrinkage = gbm.fit$bestTune[[3]],
                        n.minobsinnode = gbm.fit$bestTune[[4]],
                        importance = "permutation",
                        scale.permutation.importance = TRUE)
barplot(sort(ranger::importance(gbm.final.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(19))
```



test error

```
gbm.predict <- predict(gbm.fit, newdata = testing_data1)
gbm.RMSE <- RMSE(gbm.predict, y_test1)
gbm.RMSE
```

```
## [1] 1649.232
```

The RMSE for gbm model is 1649.232.

2. auto data

This problem is based on the data "auto.csv" in Homework 3. Split the dataset into two parts: training data (70%) and test data (30%).

```
dat2<-read_csv("./data/auto.csv")%>%
  mutate(
    mpg_cat = as.factor(mpg_cat),
    origin = as.factor(origin))
dat2 <- na.omit(dat2)
```

Outcome variable: mpg_cat

```
contrasts(dat2$mpg_cat)
```

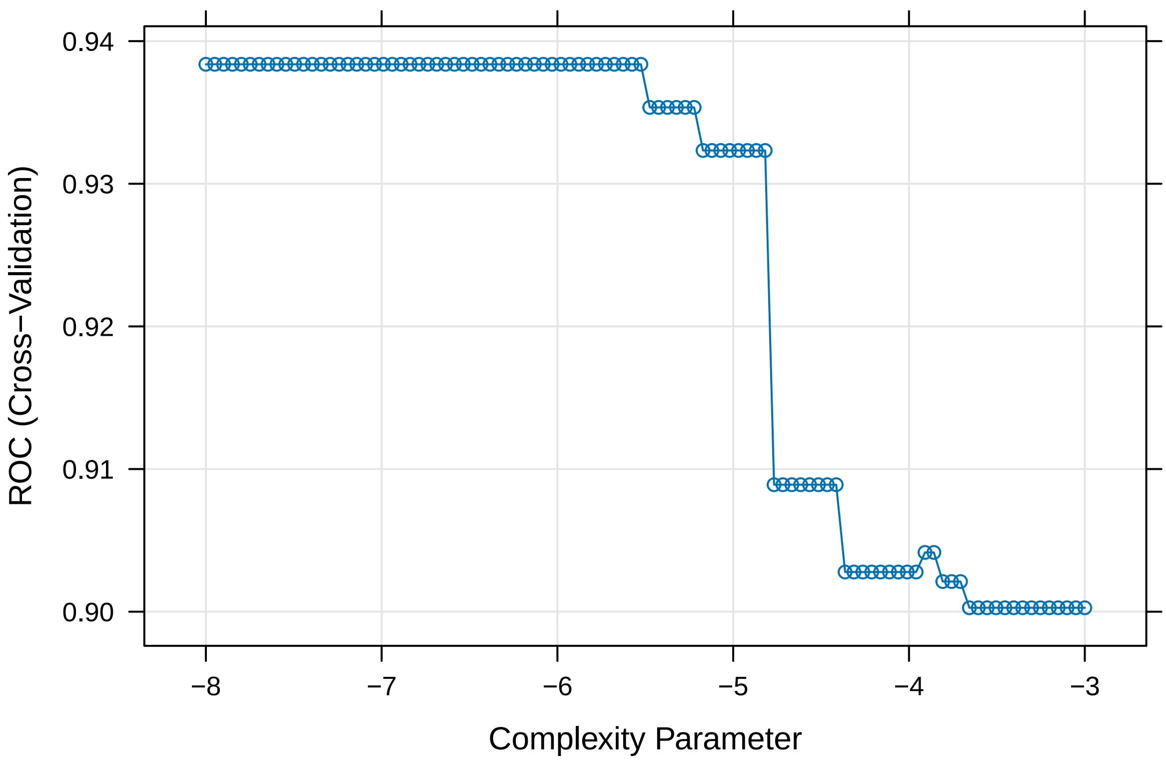
```
##      low
## high   0
## low    1
```

Split the dataset into two parts: training data (70%) and test data (30%).

```
set.seed(1)
data_split2 <- initial_split(dat2, prop = 0.7)

# Extract the training and test data
training_data2 <- training(data_split2)
testing_data2 <- testing(data_split2)

ctrl2 <- trainControl(method = "cv", number = 10,
                      summaryFunction = twoClassSummary,
                      classProbs = TRUE)
```

(a) Build a classification tree using the training data, with mpg cat as the response. Which tree size corresponds to the lowest cross-validation error? Is this the same as the tree size obtained using the 1 SE rule?

```
set.seed(1)
rpart.fit2 <- train(mpg_cat ~ . ,
                    training_data2,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-8,-3, len = 100))),
                    trControl = ctrl2,
                    metric = "ROC")
plot(rpart.fit2, xTrans = log)
```

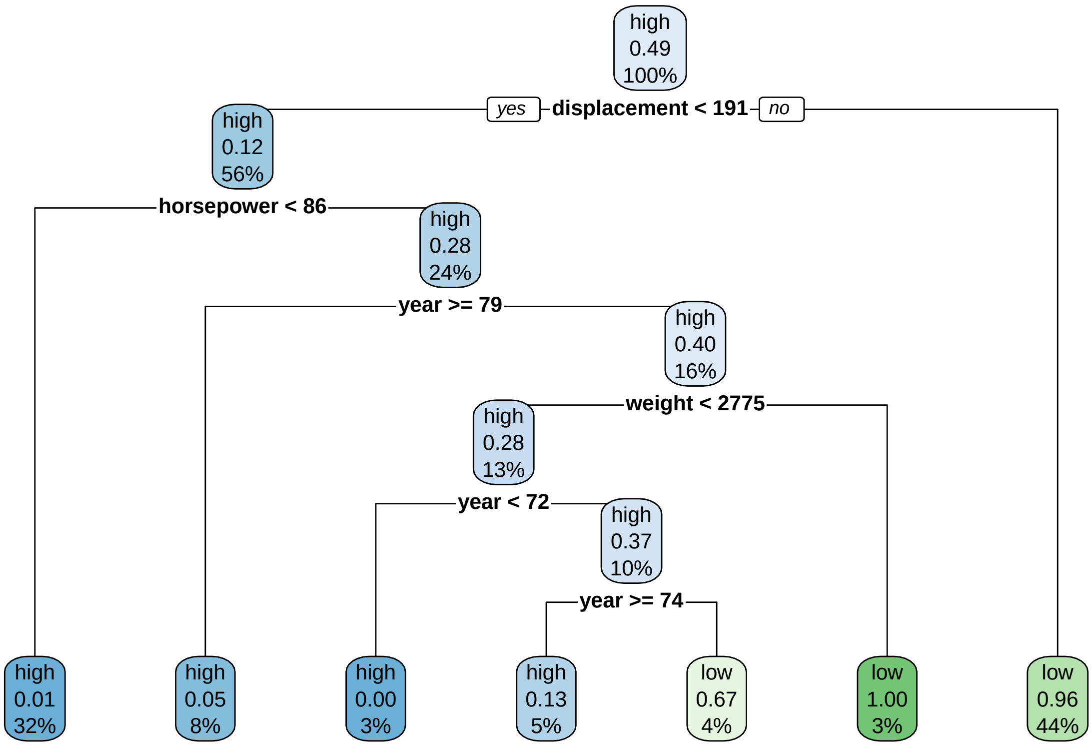



```
rpart.fit2$bestTune
```

```
##              cp
```

```
## 50 0.003984862
```

```
rpart.plot(rpart.fit2$finalModel)
```



The tree size of 7 has the lowest cross validation error. cp=0.00398

```
set.seed(1)
tree1 <- rpart(formula = mpg_cat ~ . ,
               data = training_data2,
               control = rpart.control(cp = 0))
cpTable <- printcp(tree1)
```

```
##
## Classification tree:
## rpart(formula = mpg_cat ~ ., data = training_data2, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] displacement horsepower   weight       year
##
## Root node error: 135/274 = 0.4927
##
## n= 274
##
##          CP nsplit rel error  xerror     xstd
## 1 0.822222      0  1.000000 1.03704 0.061293
## 2 0.017284      1  0.177778 0.20741 0.037140
## 3 0.014815      4  0.125926 0.20000 0.036544
## 4 0.000000      6  0.096296 0.21481 0.037720
```

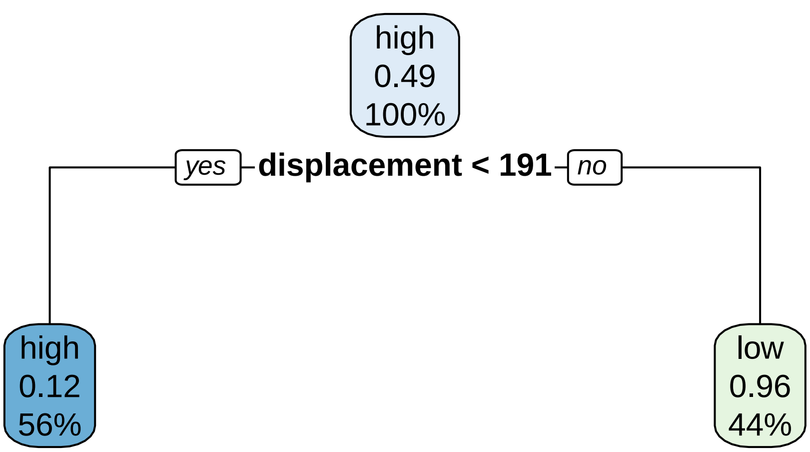
```r
#rpart.plot(tree1)
#plotcp(tree1)
```

1SE rule

```r
set.seed(1)
minErr <- which.min(cpTable[, "xerror"])
oneSE <- cpTable[minErr, "xerror"] + cpTable[minErr, "xstd"]
minErr1SE <- which(cpTable[, "xerror"] <= oneSE)[1]
tree2 <- rpart::prune(tree1, cp = cpTable[minErr1SE, "CP"])
cpTable <- printcp(tree2)
```

```
##
## Classification tree:
## rpart(formula = mpg_cat ~ ., data = training_data2, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] displacement
##
## Root node error: 135/274 = 0.4927
##
## n= 274
##
##         CP nsplit rel error  xerror     xstd
## 1 0.822222      0   1.00000 1.03704 0.061293
## 2 0.017284      1   0.17778 0.20741 0.037140
```

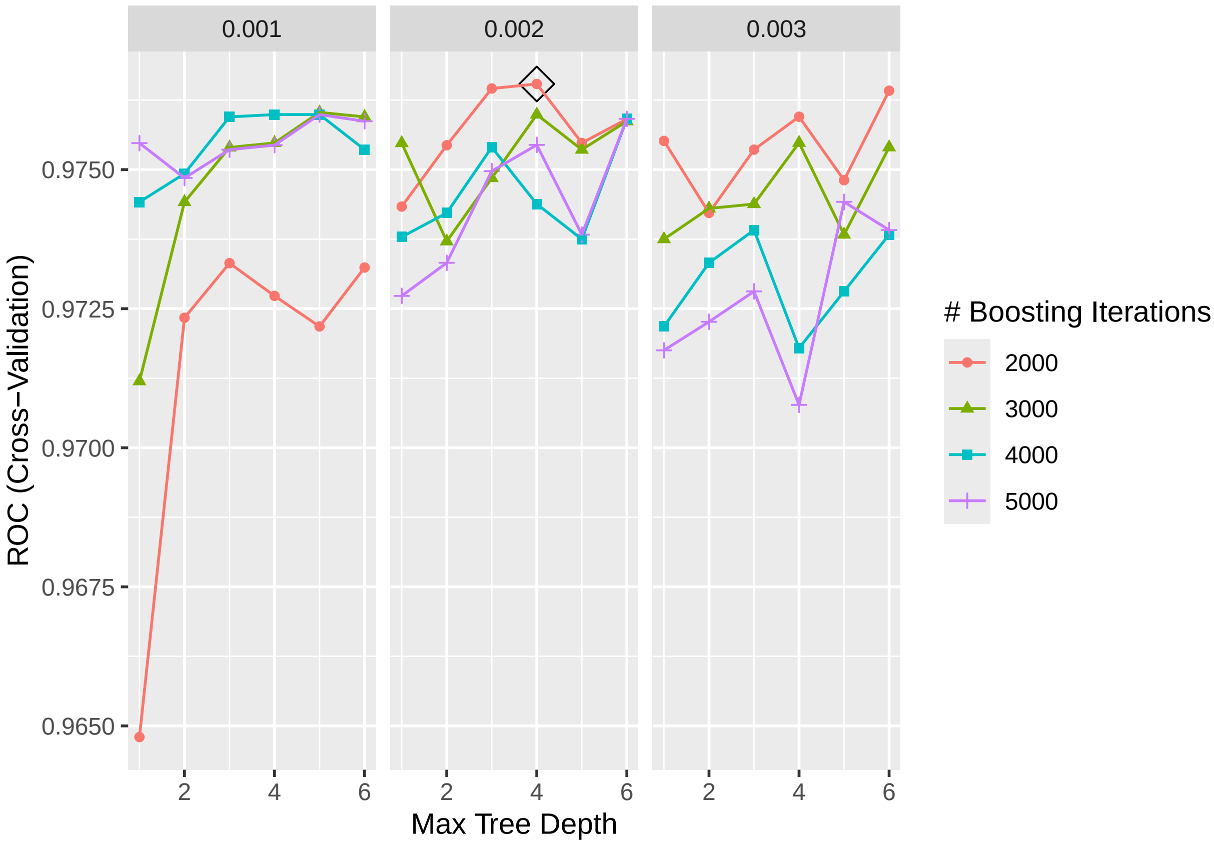
```r
#plotcp(tree2)
rpart.plot(tree2)
```



The tree size obtained using the 1 SE rule is 2. It's different from the tree size corresponds to the lowest cross-validation error.

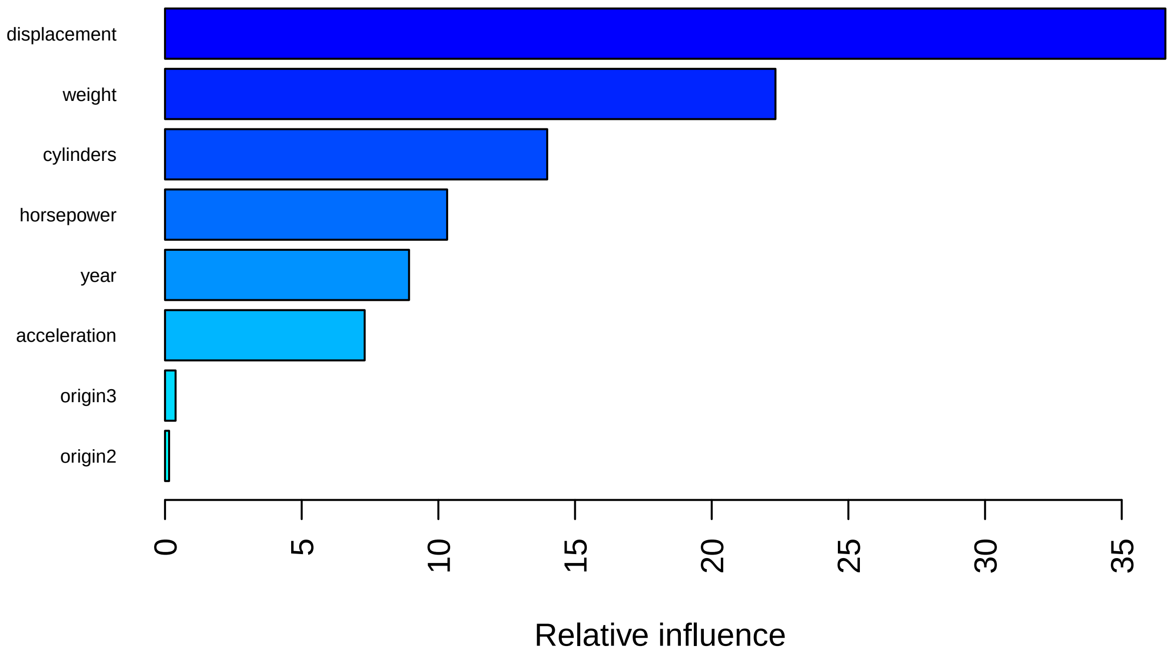
(b) Perform boosting on the training data and report the variable importance. Report the test data performance.

```r
gbmA.grid <- expand.grid(n.trees = c(2000,3000,4000,5000),
                         interaction.depth = 1:6,
                         shrinkage = c(0.001,0.002, 0.003),
                         n.minobsinnode = 1)
set.seed(1)
gbmA.fit <- train(mpg_cat ~ . ,
                  training_data2,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl2,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)
ggplot(gbmA.fit, highlight = TRUE)
```



variable importance

```r
summary(gbmA.fit$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```




```
##                       var    rel.inf
## displacement displacement 36.5969279
## weight             weight 22.3328029
## cylinders       cylinders 13.9822973
## horsepower     horsepower 10.3212955
## year                 year  8.9275619
## acceleration acceleration  7.3034086
## origin3           origin3  0.3864974
## origin2           origin2  0.1492085
```

```
gbmA.pred <- predict(gbmA.fit, newdata = testing_data2,
                     type ="prob")[,1]
resamp <- resamples(list(rf = rpart.fit2,
                         gbmA = gbmA.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: rf, gbmA
## Number of resamples: 10
##
## ROC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf   0.8727811 0.9086538 0.9432889 0.9383770 0.9751276 0.9897959    0
## gbmA 0.9408284 0.9684066 0.9744898 0.9765397 0.9907771 1.0000000    0
##
## Sens
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## rf   0.7857143 0.8008242 0.8571429 0.8774725 0.9285714    1    0
## gbmA 0.8571429 0.9285714 0.9642857 0.9565934 1.0000000    1    0
```

```
##
## Spec
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## rf   0.8461538 0.8489011 0.9258242 0.9104396 0.9285714    1    0
## gbmA 0.7692308 0.8489011 0.9258242 0.9027473 0.9285714    1    0
```

The boosting method has a higher average AUC value (0.9765) than random forest method (0.9384). Therefore, boosting method has better test data performance.