p8106_hw4

Yueyi Xu

2024-04-19

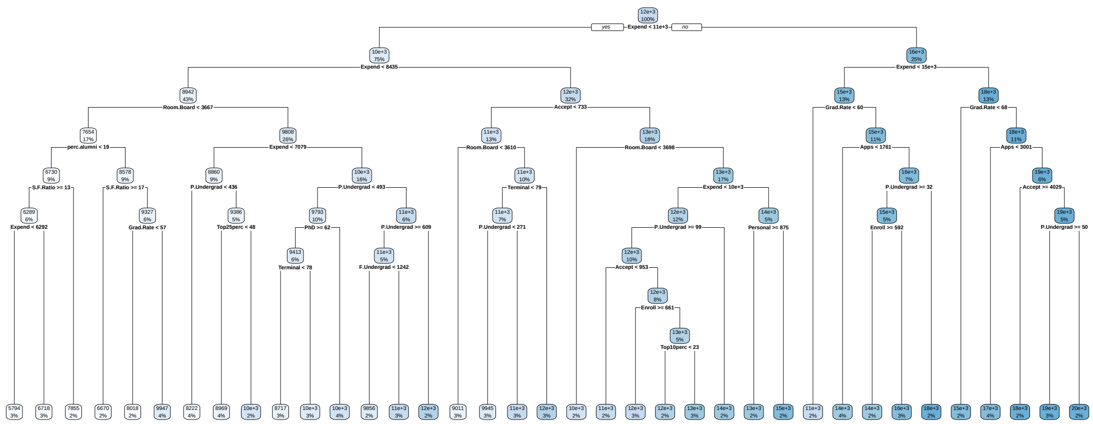
Problem 1

```
# Import data, clean the data, drop na values
college_data <- read.csv("College.csv")

college_data <- college_data %>%
  select(-College)

# Data Partition
set.seed(1)

# Split the dataset into training (80%) and test (20%) sets
data_split <- initial_split(college_data, prop = 0.8)

# Extract the training and test data
training_data <- training(data_split)
testing_data <- testing(data_split)
```

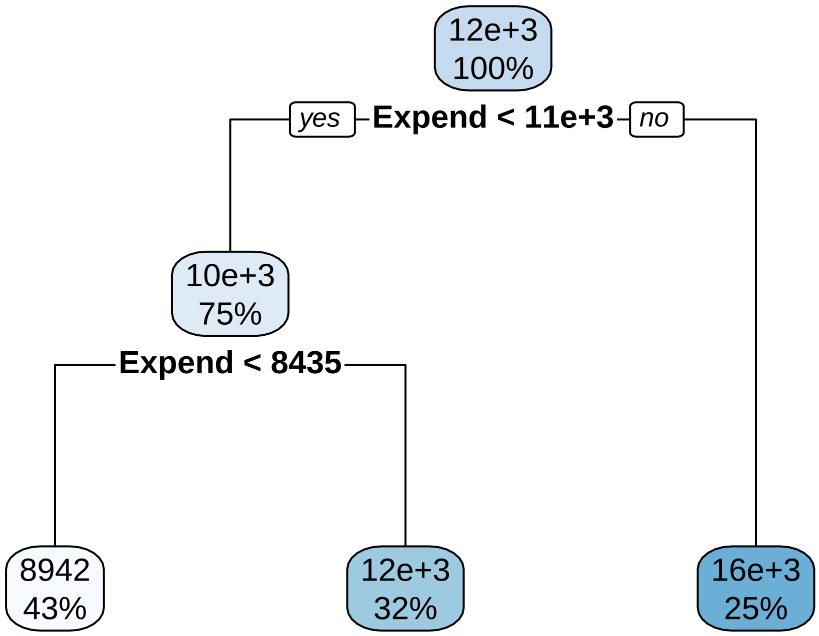
(a)

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

```
set.seed(1)
tree1 <- rpart(formula = Outstate ~.,
               data = training_data,
               control = rpart.control(cp = 0))
rpart.plot(tree1)
```



```r
set.seed(1)
tree2 <- rpart(Outstate ~.,
               data = training_data,
               control = rpart.control(cp = 0.1))
rpart.plot(tree2)
```

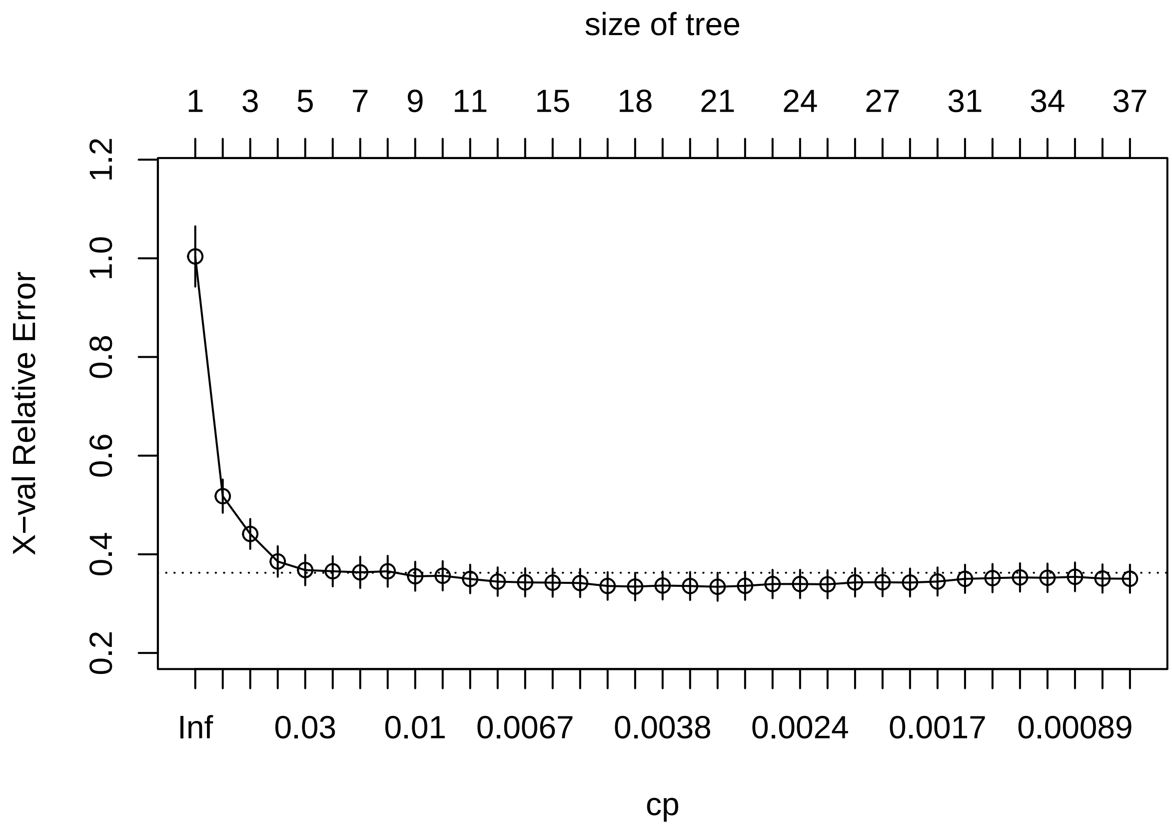



```r
printcp(tree1)
```

```
##
## Regression tree:
## rpart(formula = Outstate ~ ., data = training_data, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
##  [1] Accept      Apps        Enroll      Expend      F.Undergrad Grad.Rate
##  [7] P.Undergrad perc.alumni Personal    PhD         Room.Board  S.F.Ratio
## [13] Terminal    Top10perc   Top25perc
##
## Root node error: 6.173e+09/452 = 13657034
##
## n= 452
##
##            CP nsplit rel error  xerror     xstd
## 1  0.49687912      0   1.00000 1.00379 0.061129
## 2  0.10633322      1   0.50312 0.51785 0.033714
## 3  0.04791493      2   0.39679 0.44127 0.030355
## 4  0.03504974      3   0.34887 0.38544 0.031017
## 5  0.02506138      4   0.31382 0.36806 0.030846
## 6  0.01516529      5   0.28876 0.36565 0.030603
## 7  0.01094019      6   0.27360 0.36347 0.031752
## 8  0.01077909      7   0.26266 0.36555 0.031353
## 9  0.00959109      8   0.25188 0.35551 0.029394
```

```
## 10 0.00903094      9   0.24229 0.35649 0.029724
## 11 0.00759360     10   0.23326 0.35001 0.028943
## 12 0.00738206     11   0.22566 0.34463 0.028955
## 13 0.00599129     13   0.21090 0.34313 0.028631
## 14 0.00569487     14   0.20491 0.34252 0.028619
## 15 0.00517513     15   0.19921 0.34177 0.028597
## 16 0.00476813     16   0.19404 0.33579 0.028054
## 17 0.00399063     17   0.18927 0.33458 0.028089
## 18 0.00368136     18   0.18528 0.33666 0.028242
## 19 0.00365828     19   0.18160 0.33573 0.028439
## 20 0.00313676     20   0.17794 0.33417 0.028424
## 21 0.00281892     21   0.17480 0.33610 0.028489
## 22 0.00244723     22   0.17198 0.33976 0.028780
## 23 0.00228629     23   0.16953 0.33979 0.028707
## 24 0.00216557     24   0.16725 0.33917 0.028520
## 25 0.00209368     25   0.16508 0.34306 0.028713
## 26 0.00192948     26   0.16299 0.34334 0.028466
## 27 0.00191531     28   0.15913 0.34274 0.028410
## 28 0.00148413     29   0.15721 0.34481 0.028587
## 29 0.00147516     30   0.15573 0.35026 0.028641
## 30 0.00144328     31   0.15426 0.35172 0.028700
## 31 0.00128466     32   0.15281 0.35314 0.028802
## 32 0.00096241     33   0.15153 0.35236 0.028903
## 33 0.00082504     34   0.15057 0.35435 0.029177
## 34 0.00053977     35   0.14974 0.35088 0.028738
## 35 0.00000000     36   0.14920 0.35046 0.028559
```

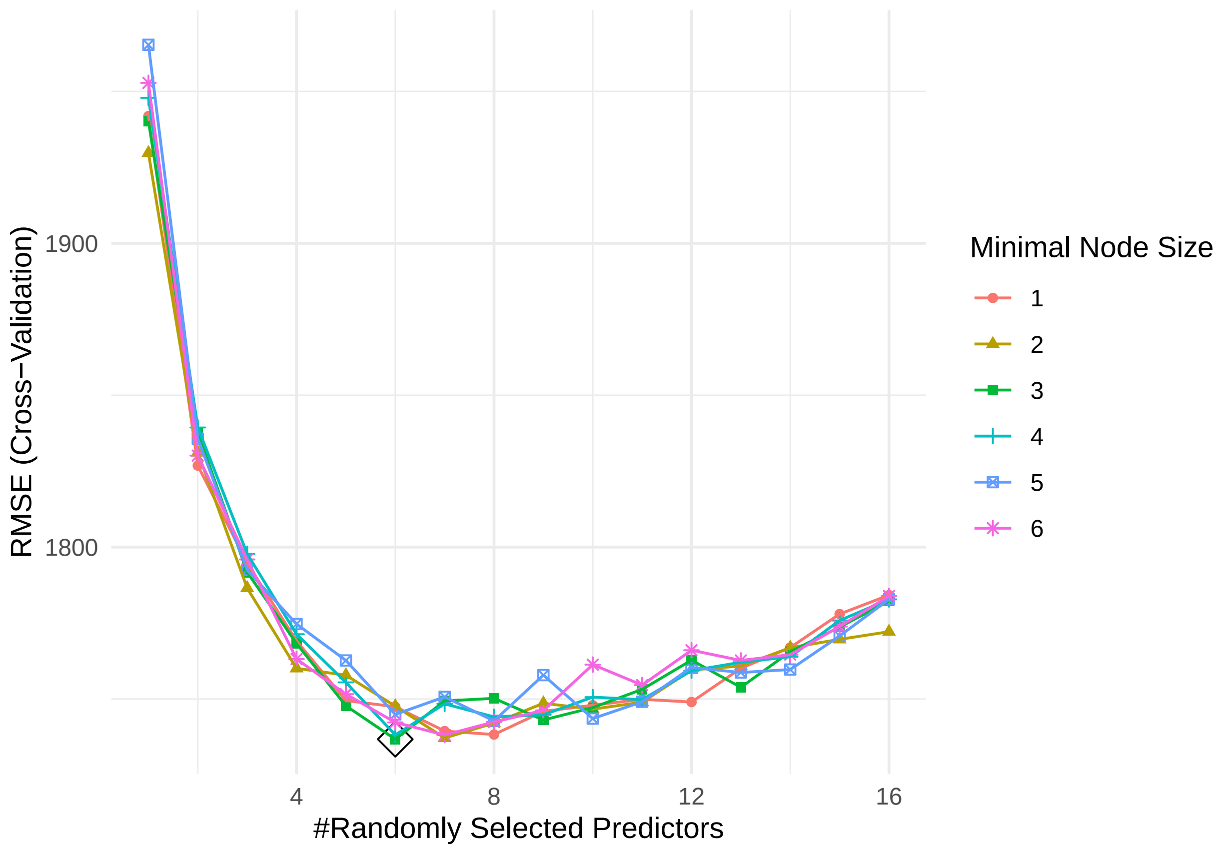
```r
cpTable <- tree1$cptable
plotcp(tree1)
```



(b)

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

```
# Perform random forest
ctrl <- trainControl(method = "cv")

rf.grid <- expand.grid(mtry = 1:16,
                       splitrule = "variance",
                       min.node.size = 1:6)
set.seed(1)
rf.fit <- train(Outstate ~ . ,
                data = training_data,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl,
                importance = "permutation")

ggplot(rf.fit, highlight = TRUE) +
  theme_minimal() +
  labs(captain = "Random Forest")
```

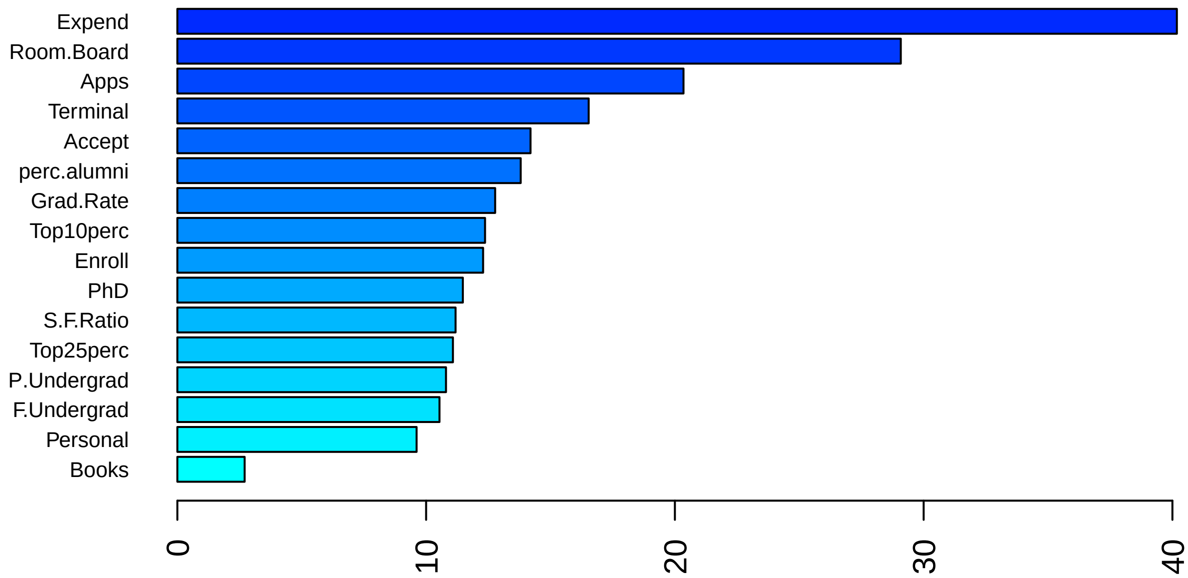



```
rf.fit$bestTune %>%
  knitr::kable(caption = "Best tune")
```

Table 1: Best tune

	mtry	splitrule	min.node.size
33	6	variance	3

```
rf2.final.per <- ranger(Outstate ~ . ,
                        training_data,
                        mtry = rf.fit$bestTune[[1]],
                        min.node.size = rf.fit$bestTune[[3]],
                        splitrule = "variance",
                        importance = "permutation",
                        scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf2.final.per),
             decreasing = FALSE),
        las = 2,
        horiz = TRUE,
        cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(19))
```




```r
# Variable Importance
df <- as.data.frame(ranger::importance(rf2.final.per))
colnames(df) <- "Importance"
df %>% arrange(desc(Importance))
```

```
##             Importance
## Expend       40.178942
## Room.Board   29.081572
## Apps         20.346272
## Terminal     16.534106
## Accept       14.196601
## perc.alumni  13.800836
## Grad.Rate    12.774939
## Top10perc    12.368317
## Enroll       12.290758
## PhD          11.475855
## S.F.Ratio    11.183291
## Top25perc    11.077537
## P.Undergrad  10.798701
## F.Undergrad  10.537615
## Personal      9.616744
## Books         2.704511
```

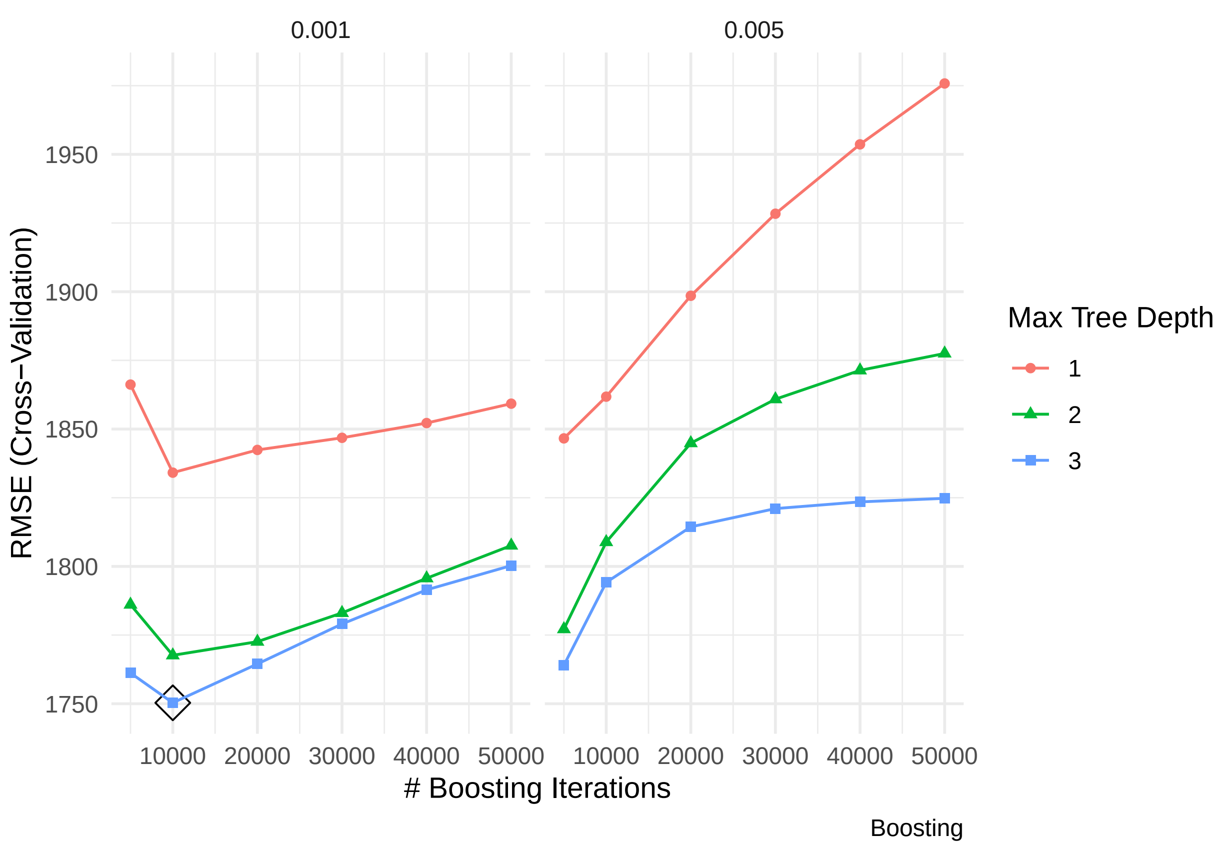
```r
# Test Error
pred.rf <- predict(rf.fit, newdata = testing_data)
RMSE.rf <- RMSE(pred.rf, testing_data$Outstate)
```

The variable "expend" is the most significant variable with 40.178942 value for predicting out-of-state tuition. The test error of RMSE for the random forest model is 1741.258.

(c)

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

```r
# Boosting
gbm.grid <- expand.grid(n.trees = c(5000,10000,20000,30000,40000,50000),
                        interaction.depth = 1:3,
                        shrinkage = c(0.001,0.005),
                        n.minobsinnode = c(1))
set.seed(1)
gbm.fit <- train(Outstate ~ . ,
                 data = training_data,
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 verbose = FALSE)

ggplot(gbm.fit, highlight = TRUE) +
  theme_minimal() +
  labs(caption = "Boosting")
```



```r
gbm.fit$bestTune %>%
  knitr::kable(caption = "Best tune for boostin")
```

Table 2: Best tune for boostin

	n.trees	interaction.depth	shrinkage	n.minobsinnode
14	10000	3	0.001	1

```r
# Variable Importance
summary(gbm.fit$finalModel, plot = FALSE)
```

```
##                     var    rel.inf
## Expend           Expend 53.244888
## Room.Board   Room.Board 10.230882
## Terminal       Terminal  5.940624
## Grad.Rate     Grad.Rate  4.278428
## perc.alumni perc.alumni  3.502031
## Apps               Apps  3.387694
## F.Undergrad F.Undergrad  2.410000
## PhD                 PhD  2.313853
## Personal       Personal  2.292650
## P.Undergrad P.Undergrad  2.164553
## Accept           Accept  2.145142
## S.F.Ratio     S.F.Ratio  2.019490
## Top25perc     Top25perc  1.921880
## Top10perc     Top10perc  1.694928
## Books             Books  1.251389
## Enroll           Enroll  1.201568
```

```r
# Test Error
gbm.pred <- predict(gbm.fit, newdata = testing_data)
RMSE.boosting <- RMSE(gbm.pred, testing_data$Outstate)
```

The variable "expend" is the most significant variable with 53.244888 value for predicting out-of-state tuition. The test error of RMSE for the boosting model is 1649.905.

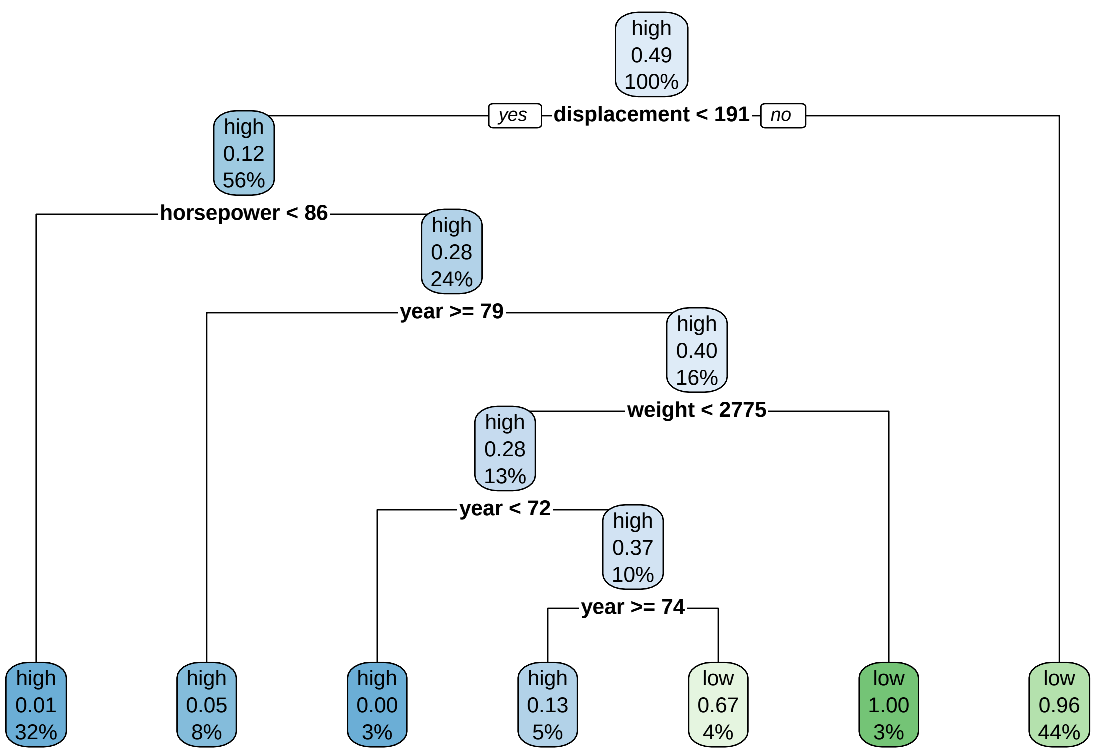
Problem 2

```r
auto_data <- read.csv("auto.csv")

auto_data <- auto_data %>%
  mutate(mpg_cat = as.factor(mpg_cat))

set.seed(1)
data_split_auto <- initial_split(auto_data, prop = 0.7)
# Extract the training and testing data
training_data_auto <- training(data_split_auto)
testing_data_auto <- testing(data_split_auto)
```

(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?

```r
# Classification tree
ctrl_2 <- trainControl(method = "cv",
                       summaryFunction = twoClassSummary,
                       classProbs = TRUE)

set.seed(1)
rpart.fit <- train(mpg_cat ~ . ,
                   training_data_auto,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-8,-3, len = 100))),
                   trControl = ctrl_2,
                   metric = "ROC")

best.ct <- prune(rpart.fit$finalModel, cp = rpart.fit$bestTune$cp)
rpart.plot(best.ct)
```

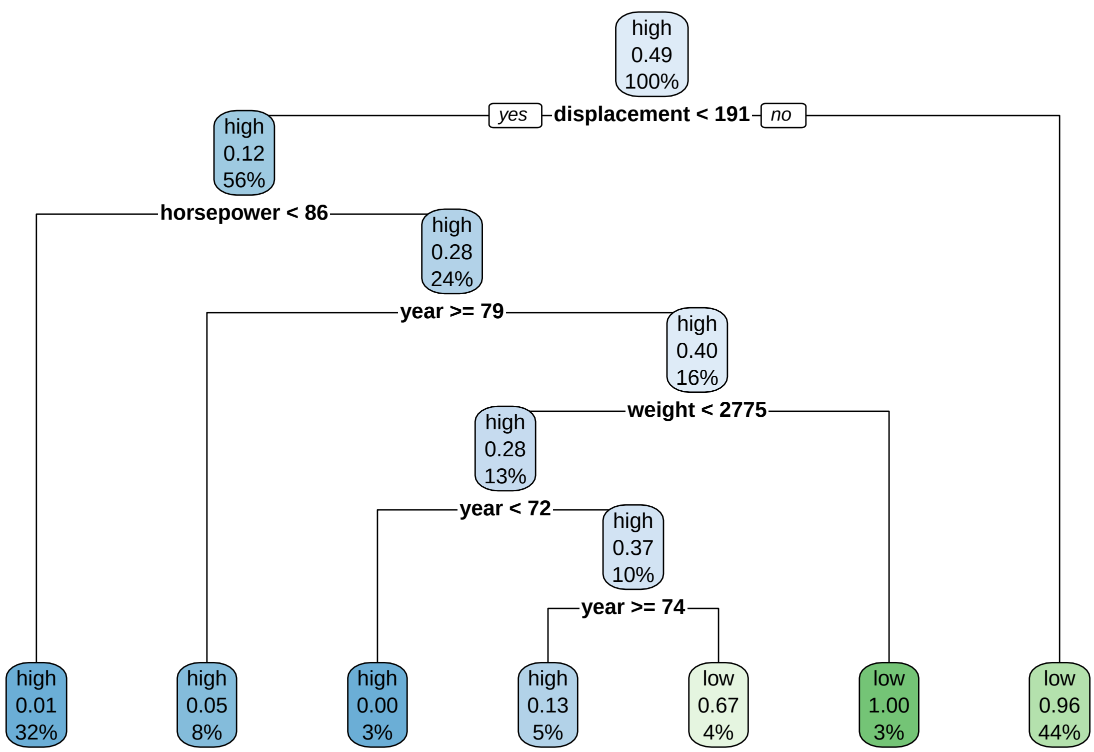



```r
size1 <- nrow(best.ct$frame)
rpart.fit$bestTune %>%
  knitr::kable(caption = "Best tune for classification tree")
```

Table 3: Best tune for classification tree

	cp
50	0.0039849

```
# 1SE
ctrl_1se <- trainControl(method = "cv",
                         summaryFunction = twoClassSummary,
                         selectionFunction = "oneSE",
                         classProbs = TRUE)
set.seed(1)

rpart.fit_1se <- train(mpg_cat ~ . ,
                       training_data_auto,
                       method = "rpart",
                       tuneGrid = data.frame(cp = exp(seq(-8,-3, len = 100))),
                       trControl = ctrl_1se,
                       metric = "ROC")

oneSE.ct <- prune(rpart.fit_1se$finalModel, cp = rpart.fit_1se$bestTune$cp)
rpart.plot(oneSE.ct)
```




```
size2 <- nrow(oneSE.ct$frame)
rpart.fit_1se$bestTune %>%
  knitr::kable(caption = "Best tune for 1SE classification tree")
```

Table 4: Best tune for 1SE classification tree

	cp
64	0.0080815

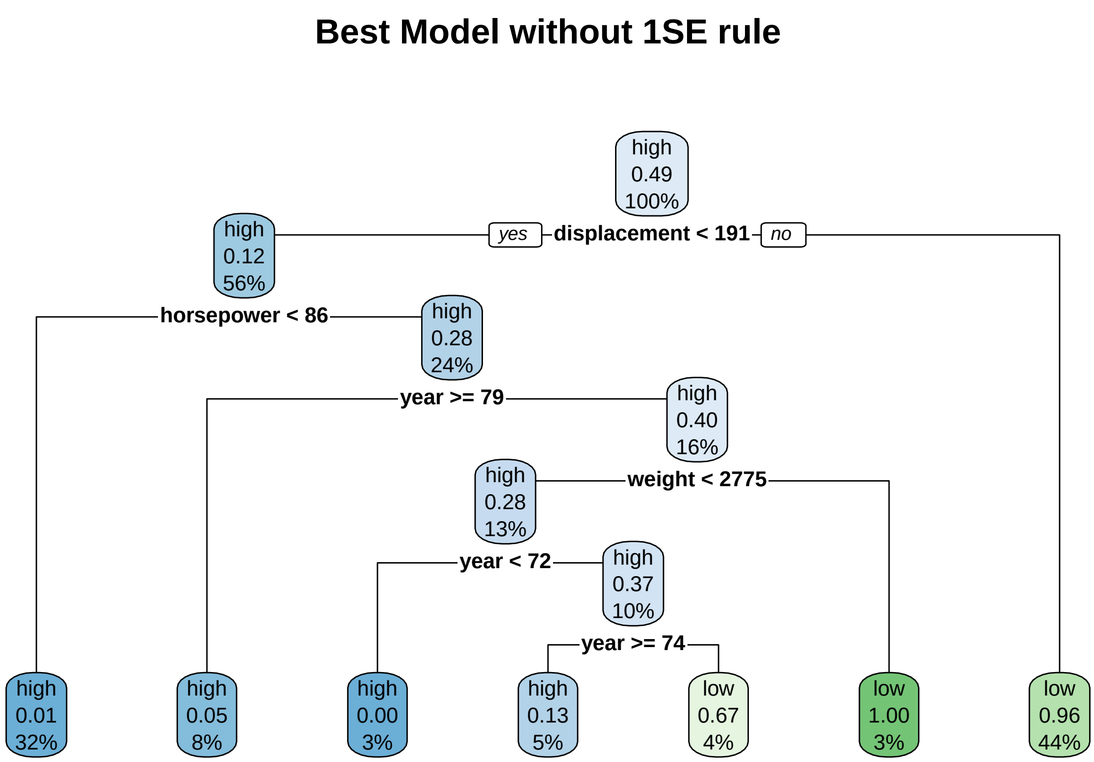
```r
# Displaying best cp values
cat("Best cp without 1SE rule:", rpart.fit$bestTune$cp, "\n")
```

```
## Best cp without 1SE rule: 0.003984862
```

```r
cat("Best cp with 1SE rule:", rpart.fit_1se$bestTune$cp, "\n")
```

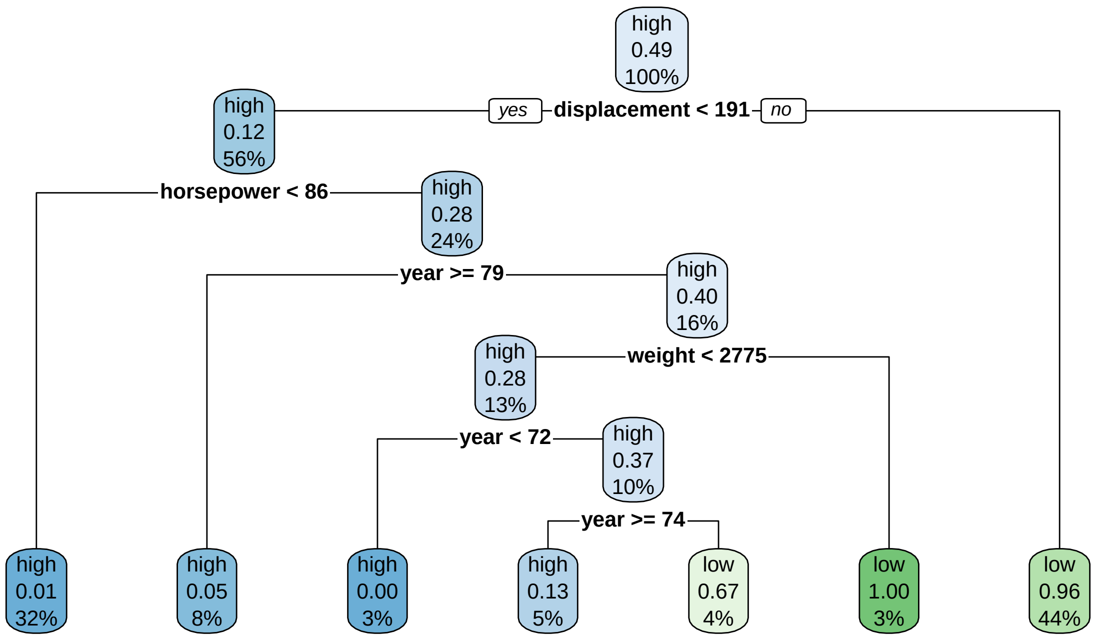
```
## Best cp with 1SE rule: 0.008081467
```

```r
# Compare trees
rpart.plot(rpart.fit$finalModel, main = "Best Model without 1SE rule")
```




```r
rpart.plot(rpart.fit_1se$finalModel, main = "Best Model with 1SE rule")
```

Best Model with 1SE rule



Both without the 1SE rule and with the 1SE rule, the tree size is 13. However, the best model without the 1SE rule has the complexity parameter of 0.003984862 while the model with 1SE has the complexity parameter of 0.008081467.

(b)

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

```r
# Boosting
ctrl_3 <- trainControl(method = "cv",
                       summaryFunction = twoClassSummary,
                       classProbs = TRUE,
                       selectionFunction = "best")

gbm.grid.auto <- expand.grid(n.trees = c(5000,10000,20000,30000,40000,50000),
                             interaction.depth = 1:3,
                             shrinkage = c(0.001,0.005),
                             n.minobsinnode = c(1))
set.seed(1)
gbm.fit.auto <- train(mpg_cat ~ . ,
                      data = training_data_auto,
                      method = "gbm",
                      tuneGrid = gbm.grid,
                      trControl = ctrl_3,
                      verbose = FALSE)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
# Variable Importance
summary(gbm.fit.auto$finalModel, plot = FALSE)
```

```
##                       var     rel.inf
## displacement displacement 39.21302364
## weight             weight 23.20004110
## cylinders       cylinders 18.90377862
## horsepower     horsepower 12.90864349
## year                 year  5.19847041
## acceleration acceleration  0.54678952
## origin             origin  0.02925322
```

```
# Test Error
gbm.pred.auto <- predict(gbm.fit.auto, newdata = testing_data_auto, type = 'prob')[,1]
pROC::roc(testing_data_auto$mpg_cat, gbm.pred.auto)
```

```
## Setting levels: control = high, case = low
```

```
## Setting direction: controls > cases
```

```
##
## Call:
## roc.default(response = testing_data_auto$mpg_cat, predictor = gbm.pred.auto)
##
## Data: gbm.pred.auto in 57 controls (testing_data_auto$mpg_cat high) > 61 cases (testing_data_auto$mpg
## Area under the curve: 0.9781
```

The variable "displacement" is the most significant variable with 39.21302364 value for predicting out-of-state tuition. The AUC value is 0.9781.