8106hw4

Ze Li

Contents


```
library(ISLR)
library(caret)
library(mgcv)
library(tidymodels)
library(rpart)
library(rpart.plot)
library(randomForest)
library(ranger)
library(gbm)
library(pROC)
```

Problem 1

```
college=read.csv("/Users/zeze/Library/Mobile Documents/com~apple~CloudDocs/2024/24S BIST P8106 DS II/hw
indexTrain <- createDataPartition(y = college$Outstate, p = 0.8, list = FALSE)
train <- college[indexTrain, ][-1]
test <- college[-indexTrain, ][-1]
train <- na.omit(train)
```

```r
test <- na.omit(test)
head(train)
```

```
##   Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad Outstate
## 1 1660   1232    721        23        52        2885         537     7440
## 2 2186   1924    512        16        29        2683        1227    12280
## 3 1428   1097    336        22        50        1036          99    11250
## 5  193    146     55        16        44         249         869     7560
## 6  587    479    158        38        62         678          41    13500
## 7  353    340    103        17        45         416         230    13290
##   Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni Expend Grad.Rate
## 1       3300   450     2200  70       78      18.1          12   7041        60
## 2       6450   750     1500  29       30      12.2          16  10527        56
## 3       3750   400     1165  53       66      12.9          30   8735        54
## 5       4120   800     1500  76       72      11.9           2  10922        15
## 6       3335   500      675  67       73       9.4          11   9727        55
## 7       5720   500     1500  90       93      11.5          26   8861        63
```

```r
# matrix of predictors
x_train <- model.matrix(Outstate ~ ., train)[, -1]
head(x_train)
```

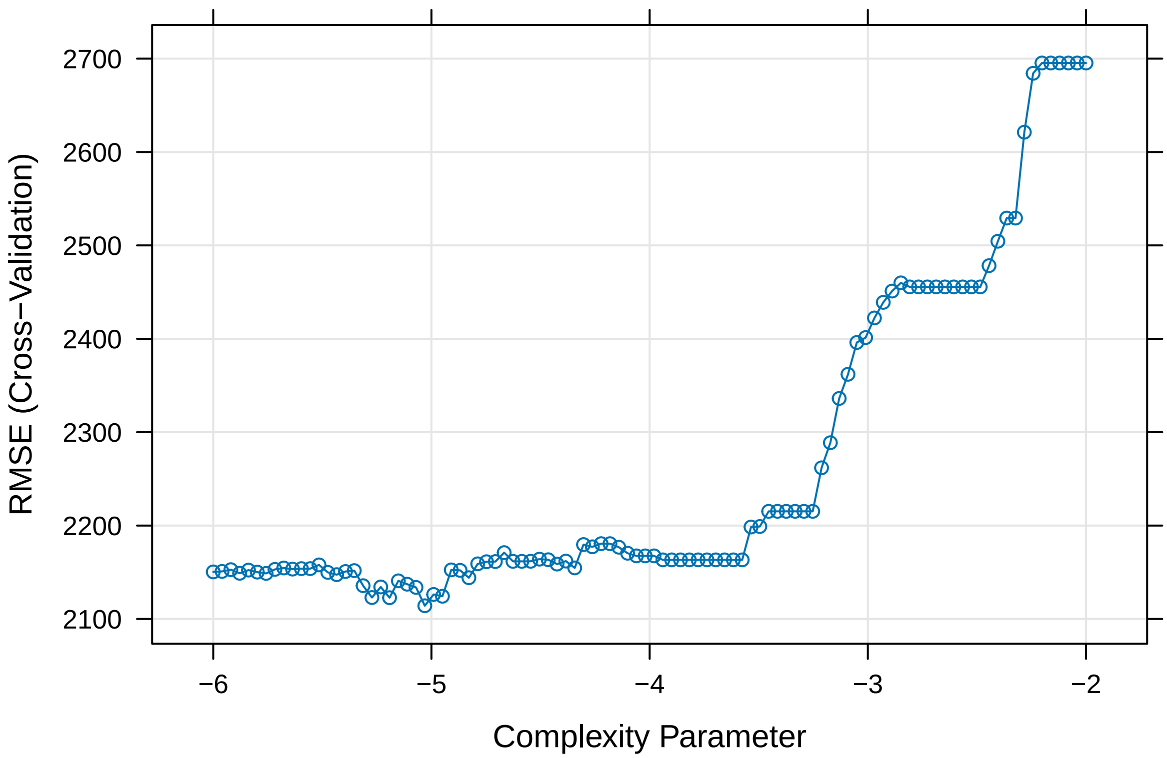
```
##   Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad Room.Board
## 1 1660   1232    721        23        52        2885         537       3300
## 2 2186   1924    512        16        29        2683        1227       6450
## 3 1428   1097    336        22        50        1036          99       3750
## 5  193    146     55        16        44         249         869       4120
## 6  587    479    158        38        62         678          41       3335
## 7  353    340    103        17        45         416         230       5720
##   Books Personal PhD Terminal S.F.Ratio perc.alumni Expend Grad.Rate
## 1   450     2200  70       78      18.1          12   7041        60
## 2   750     1500  29       30      12.2          16  10527        56
## 3   400     1165  53       66      12.9          30   8735        54
## 5   800     1500  76       72      11.9           2  10922        15
## 6   500      675  67       73       9.4          11   9727        55
## 7   500     1500  90       93      11.5          26   8861        63
```

```r
# vector of response
y_train <- train$Outstate
# matrix of predictors
x_test <- model.matrix(Outstate ~ ., test)[, -1]
# vector of response
y_test <- test$Outstate
```

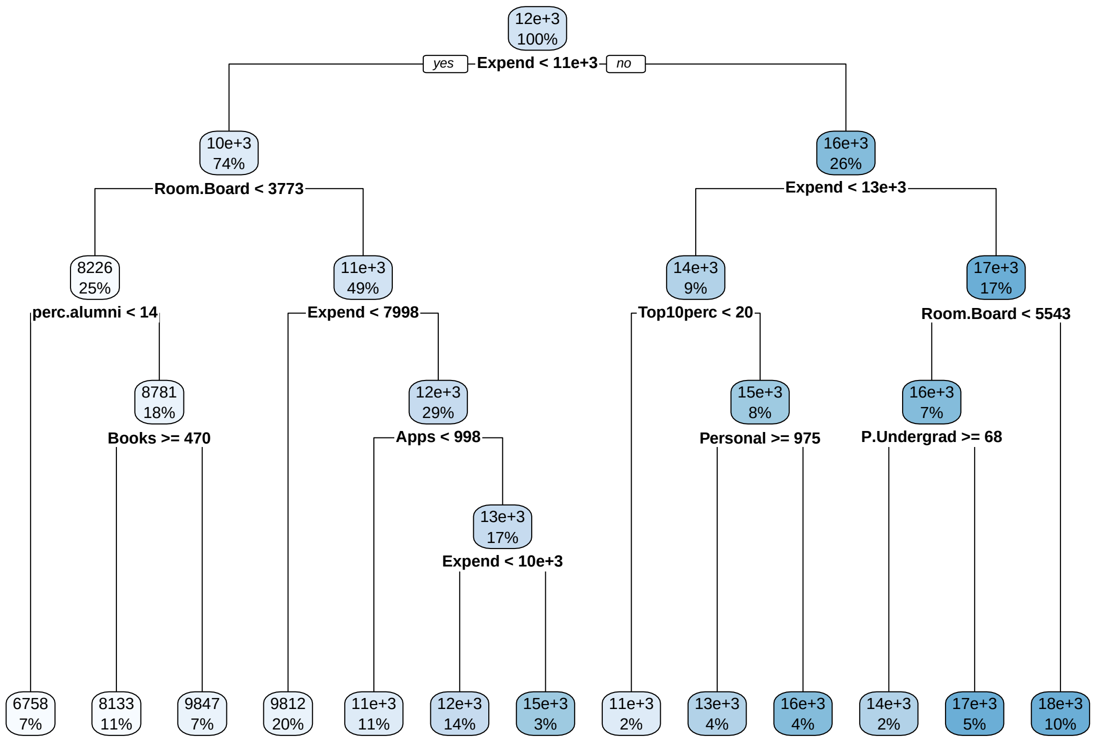
(a) Build a regression tree on the training data to predict the response. Create a plot of the tree.

```r
ctrl <- trainControl(method = "cv")
set.seed(1)
rpart.fit <- train(Outstate ~ . , train,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 100))),
```

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



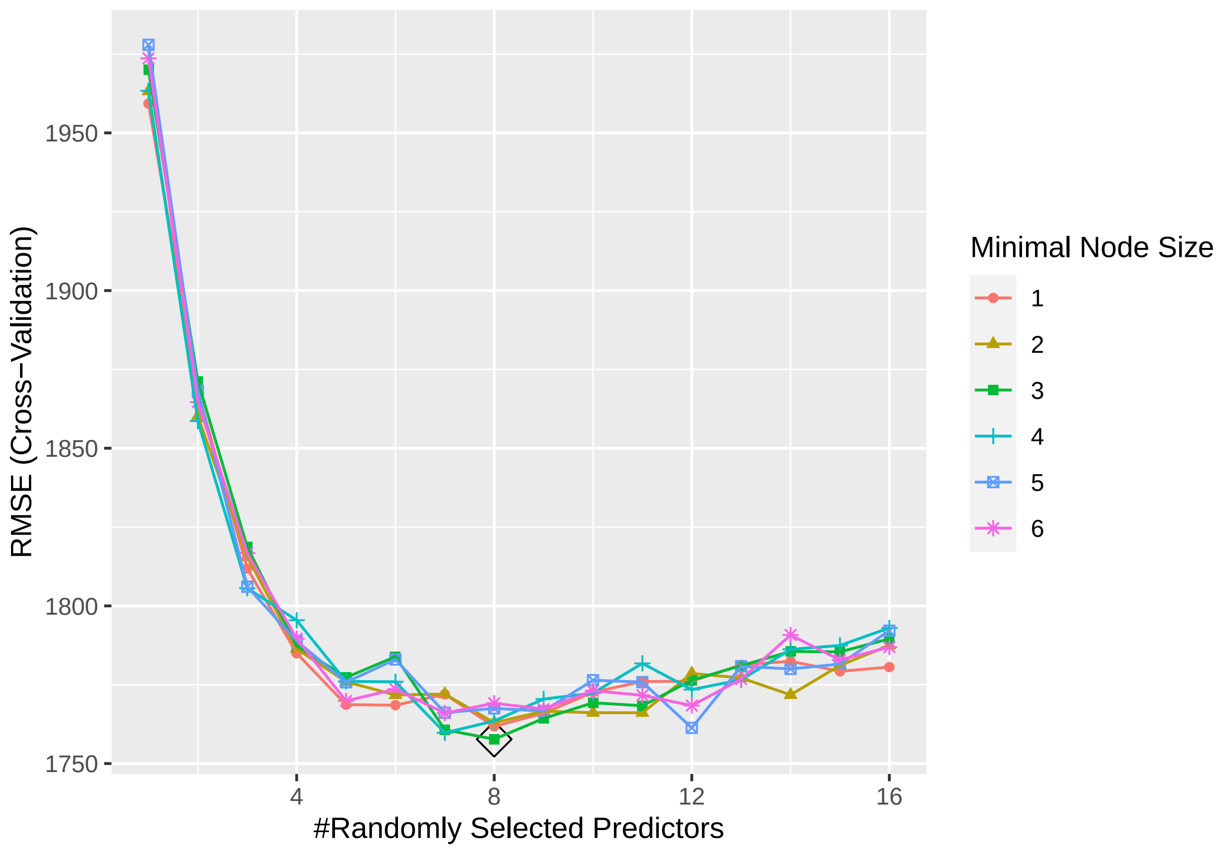

```
rpart.plot(rpart.fit$finalModel)
```



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

Random Forest

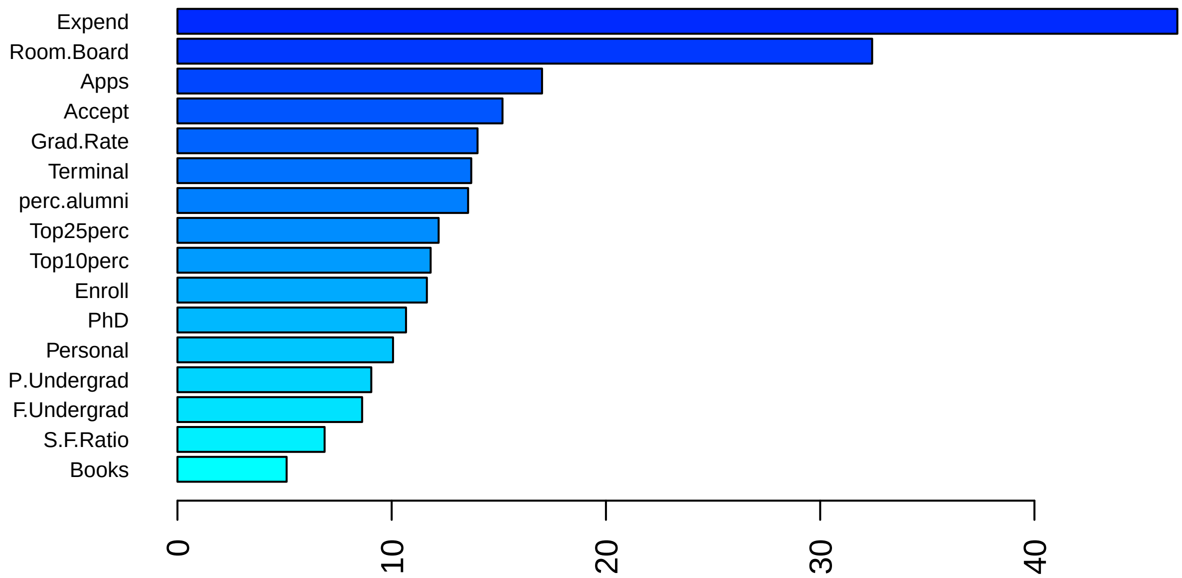
```
ctrl <- trainControl(method = "cv")
rf.grid <- expand.grid(mtry = 1:16, splitrule = "variance", min.node.size = 1:6)
set.seed(1)
rf.fit <- train(Outstate ~ . , data = train,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl)
ggplot(rf.fit, highlight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



Random Forest - Variable Importance

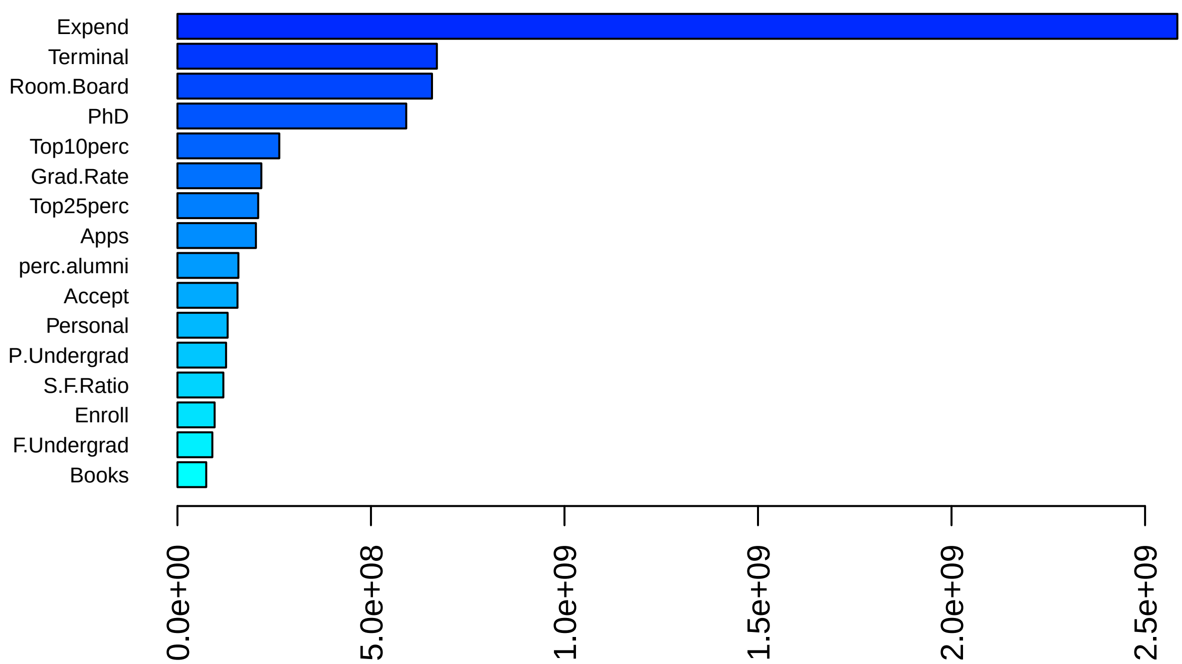
```
set.seed(1)
rf2.final.per <- ranger(Outstate ~ . ,
                        data = train,
                        mtry = rf.fit$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = rf.fit$bestTune[[3]],
                        importance = "permutation",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




```
set.seed(1)
rf2.final.imp <- ranger(Outstate ~ . ,
                        data = train,
                        mtry = rf.fit$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = rf.fit$bestTune[[3]],
                        importance = "impurity")

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



Random Forest - Test Error

```
pred.rf <- predict(rf.fit, newdata = test)
RMSE(pred.rf, y_test)
```

```
## [1] 1608.216
```

The two bar plots illustrate the variable importance derived from a random forest model, with the first plot representing permutation importance and the second reflecting impurity importance. In both metrics, 'Expend' stands out as the most influential predictor, indicating that the amount spent per student is a key factor in predicting the response variable 'Outstate'. Academic-related variables such as 'Room.Board', 'Terminal', 'PhD', and 'Top10perc' also rank highly across both importance measures, underscoring the relevance of financial and educational quality factors in the model's predictions.

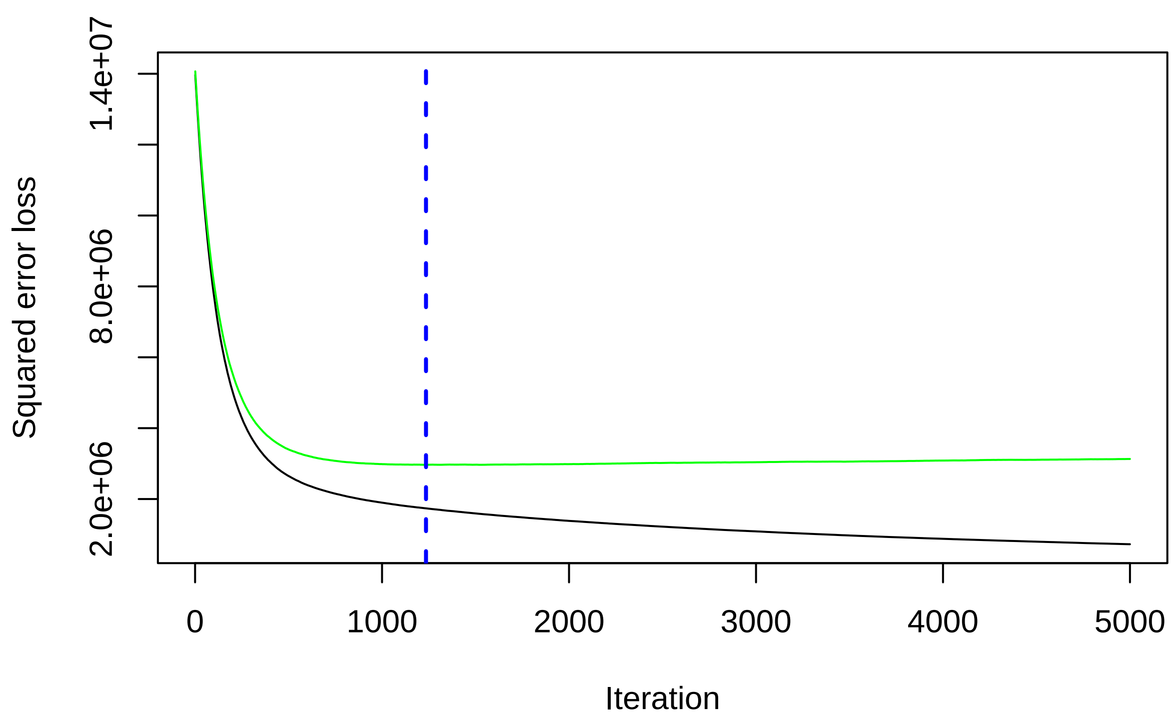
The test error is 1608.215614.

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

Boosting

```
# We first fit a gradient boosting model with Gaussian loss function
set.seed(1)
```

```r
bst <- gbm(Outstate ~ . ,
           data = train,
           distribution = "gaussian",
           n.trees = 5000,
           interaction.depth = 3,
           shrinkage = 0.005,
           cv.folds = 10,
           n.cores = 2)
# We plot loss function as a result of number of trees added to the ensemble
gbm.perf(bst, method = "cv")
```

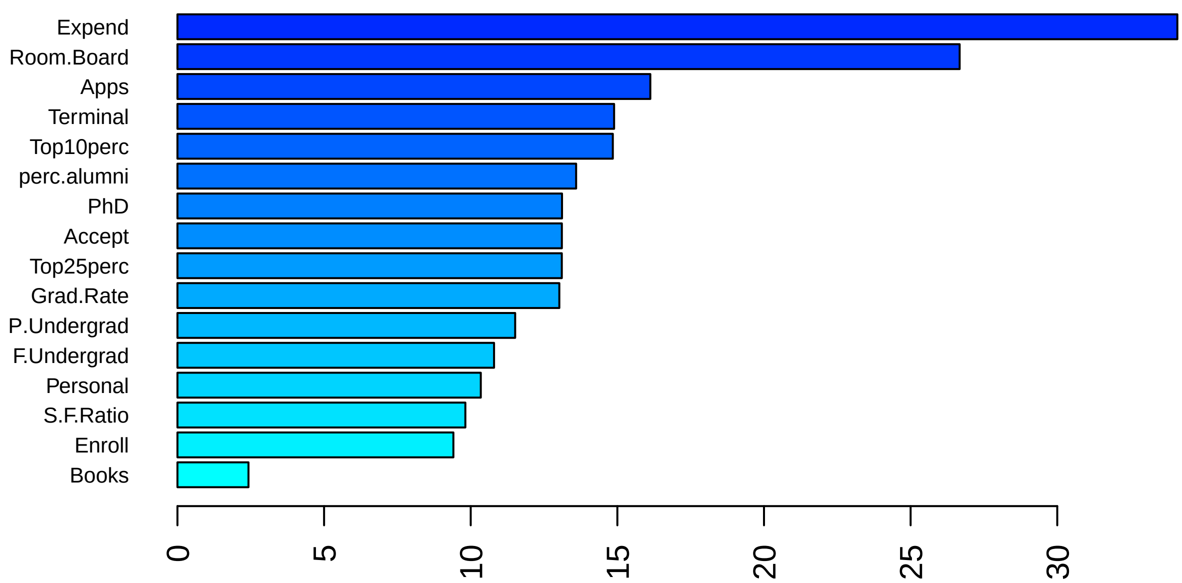



```
## [1] 1235
```

Boosting - Variable Importance

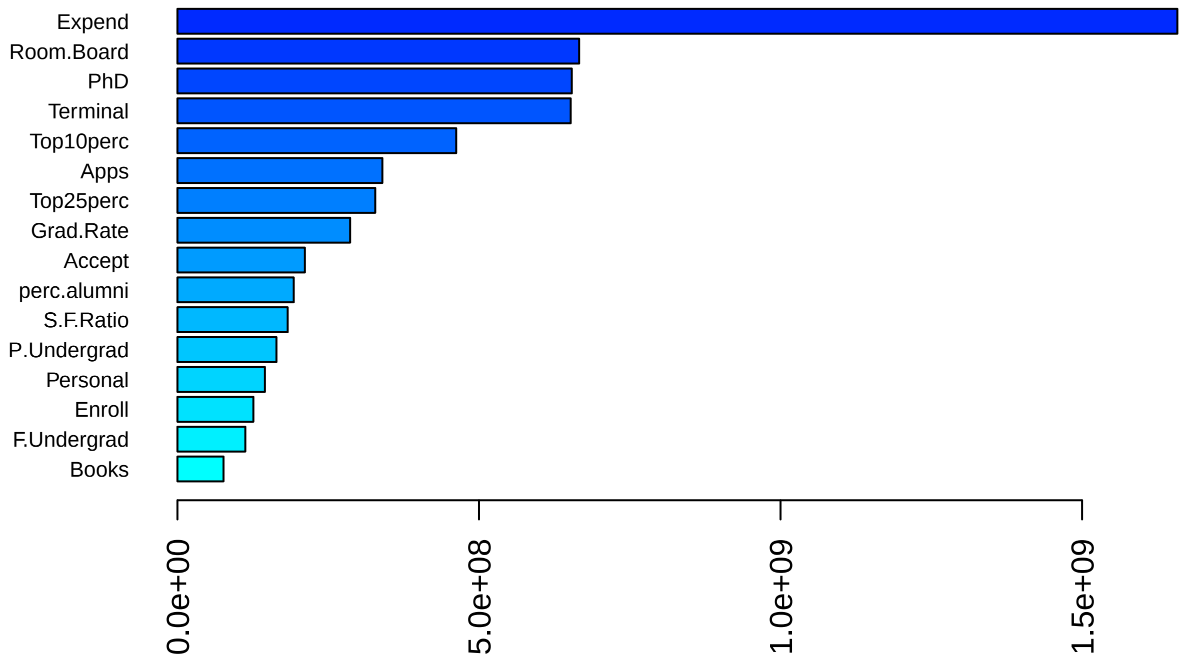
```r
set.seed(1)
gbm.final.per <- ranger(Outstate ~ . ,
                        data = train,
                        mtry = bst$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = bst$bestTune[[3]],
                        importance = "permutation",
                        scale.permutation.importance = TRUE)
```

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




```r
set.seed(1)
gbm.final.imp <- ranger(Outstate ~ . ,
                        data = train,
                        mtry = bst$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = bst$bestTune[[3]],
                        importance = "impurity")

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



Boosting - Test Error

```
pred.gbm <- predict(bst, newdata = test)
```

```
## Using 1235 trees...
```

```
RMSE(pred.gbm, y_test)
```

```
## [1] 1717.951
```

Among predictors, the most significantly one influences the model's ability to predict the 'Outstate' variable. In both measures, 'Expend' emerges as the most influential variable, suggesting that expenditure per student is a dominant predictor. This is followed by academic-related factors such as 'Terminal', 'PhD', and student performance metrics 'Top10perc' which also hold significant importance, reflecting the relevance of academic excellence and resources in predicting 'Outstate'.

The test error is 1717.951023.

Problem 2

```r
auto = read.csv("/Users/zeze/Library/Mobile Documents/com~apple~CloudDocs/2024/24S BIST P8106 DS II/hw4/
auto = auto |>
  drop_na() |>
  mutate(mpg_cat = as.factor(mpg_cat),
         origin = as.factor(origin))
head(auto)
```

```
##   cylinders displacement horsepower weight acceleration year origin mpg_cat
## 1         8          307        130   3504         12.0   70      1     low
## 2         8          350        165   3693         11.5   70      1     low
## 3         8          318        150   3436         11.0   70      1     low
## 4         8          304        150   3433         12.0   70      1     low
## 5         8          302        140   3449         10.5   70      1     low
## 6         8          429        198   4341         10.0   70      1     low
```

```r
data_split <- initial_split(auto, prop = 0.7)
train2 <- training(data_split)
test2 <- testing(data_split)
#indexTrain2 <- createDataPartition(y = auto$mpg_cat, p = 0.7, list = FALSE)
#train2 <- auto[indexTrain2, ]
#test2 <- auto[-indexTrain2, ]
head(train2)
```

```
##   cylinders displacement horsepower weight acceleration year origin mpg_cat
## 1         4           90         48   2085         21.7   80      2    high
## 2         4          140         83   2639         17.0   75      1    high
## 3         4          122         80   2451         16.5   74      1    high
## 4         8          260         90   3420         22.2   79      1    high
## 5         4          156        105   2745         16.7   78      1    high
## 6         8          318        150   4190         13.0   76      1     low
```

(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?

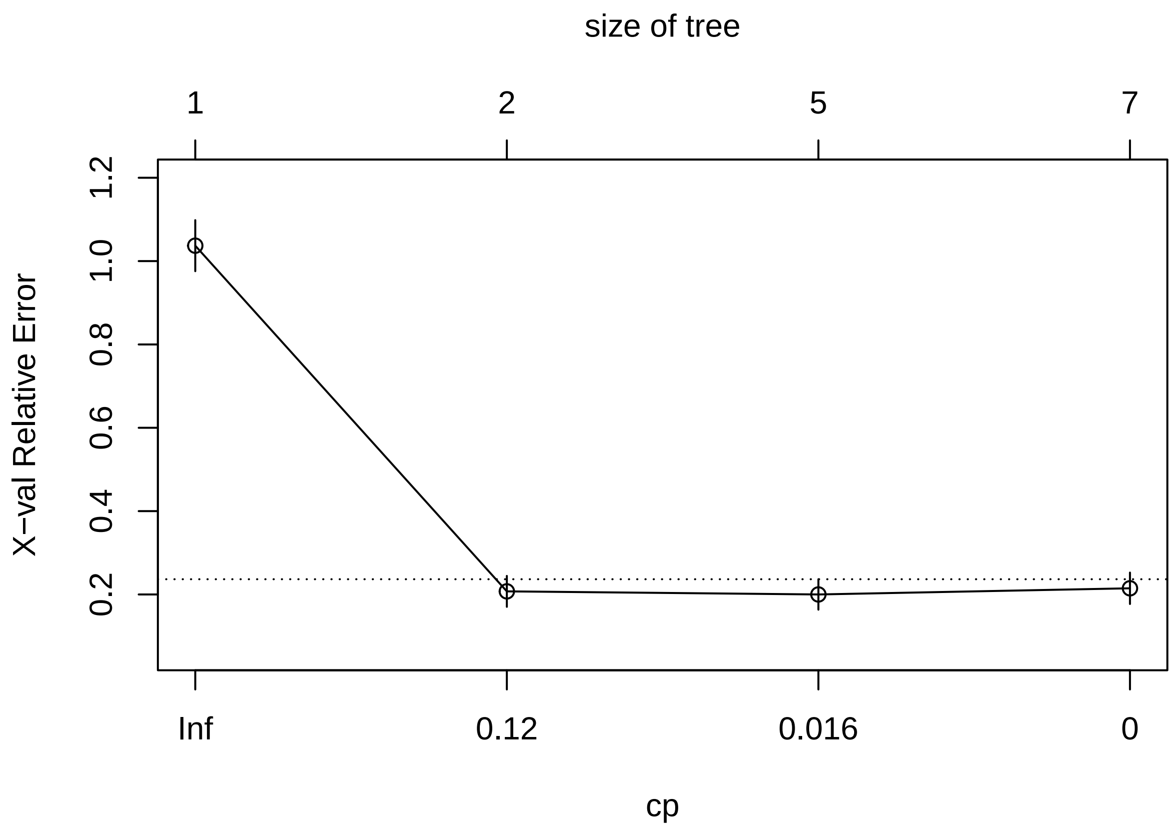
Classification Tree

```r
set.seed(1)
tree1 <- rpart(formula = mpg_cat ~ . , data = train2,
               control = rpart.control(cp=0))
cpTable <- printcp(tree1)
```

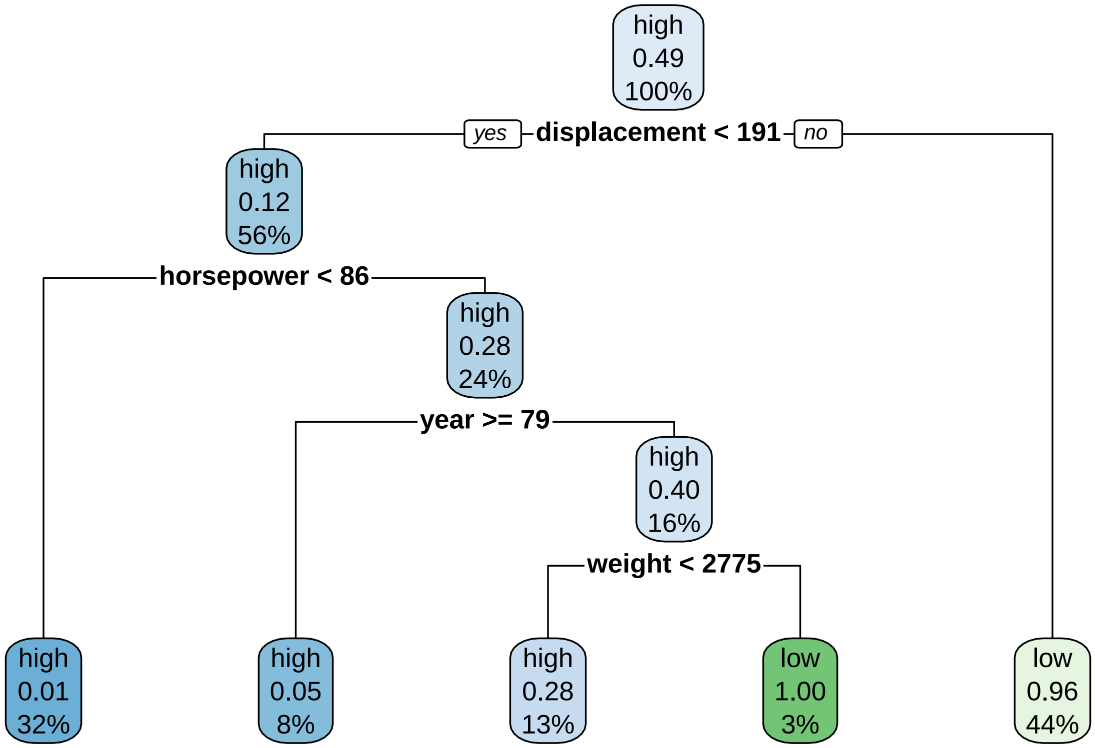
```
##
## Classification tree:
## rpart(formula = mpg_cat ~ ., data = train2, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] displacement horsepower   weight       year
##
```

```
## 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
plotcp(tree1)
```

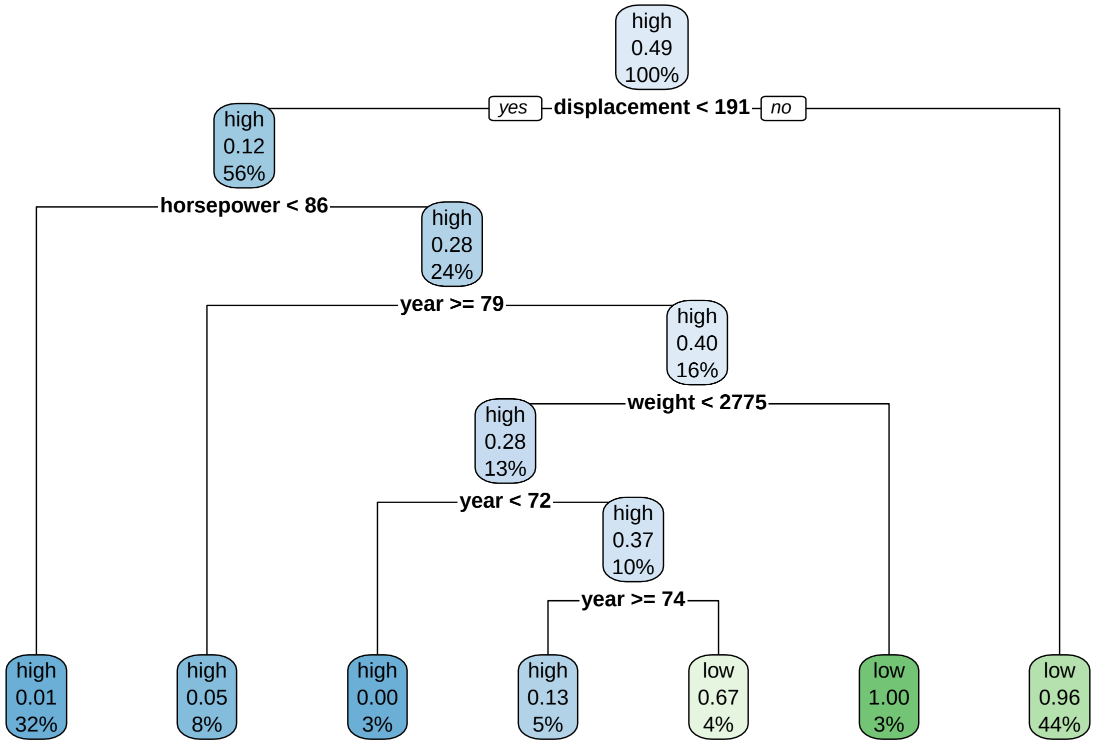



```r
minErr <- which.min(cpTable[,4])
tree2 <- rpart::prune(tree1, cp = cpTable[minErr,1])
rpart.plot(tree2)
```



1se

```
minErr <- which.min(cpTable[, "xerror"])
minCVError <- cpTable[minErr, "xerror"]
minErrSE <- cpTable[minErr, "xstd"]
seIndex <- max(which(cpTable[, "xerror"] <= (minCVError + minErrSE)))
tree3 <- rpart::prune(tree1, cp = cpTable[seIndex, "CP"])
rpart.plot(tree3)
```

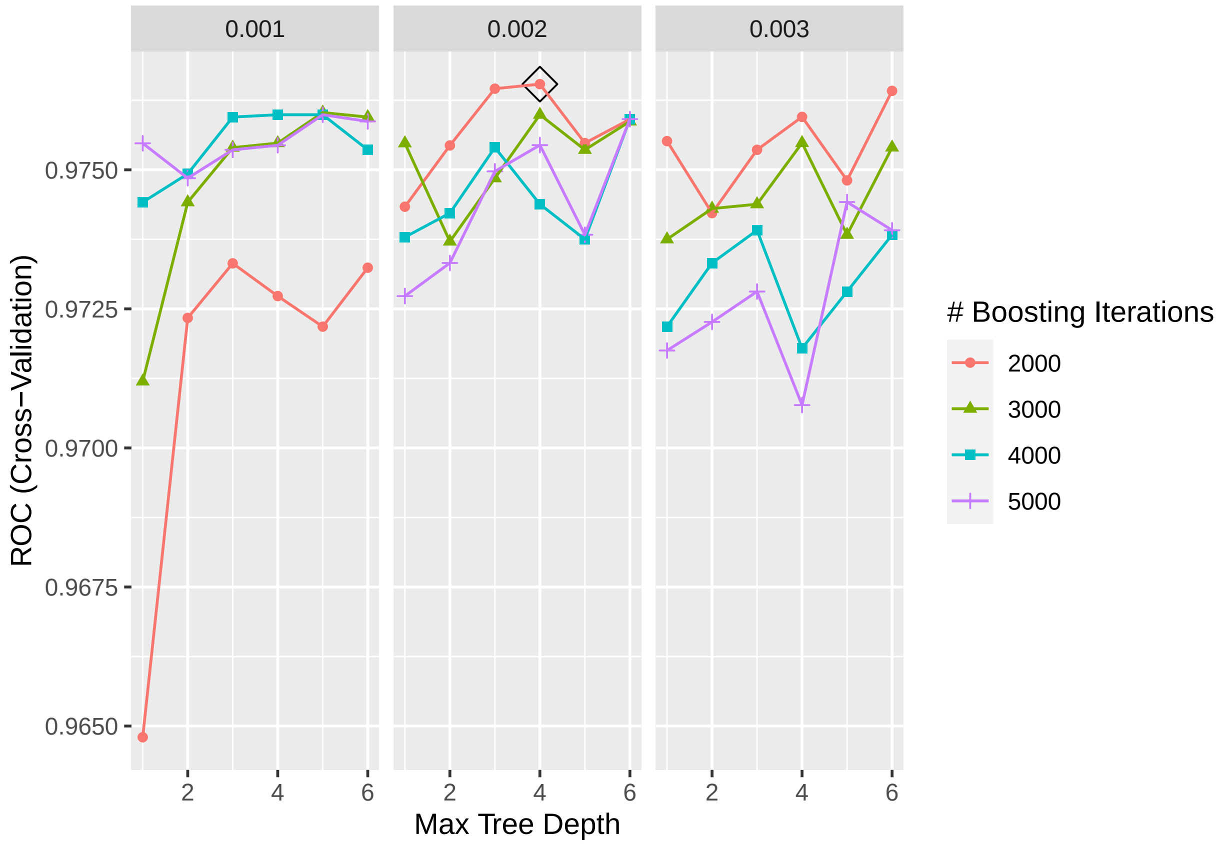


The tree size corresponds to the lowest cross-validation error is different after applying 1se.

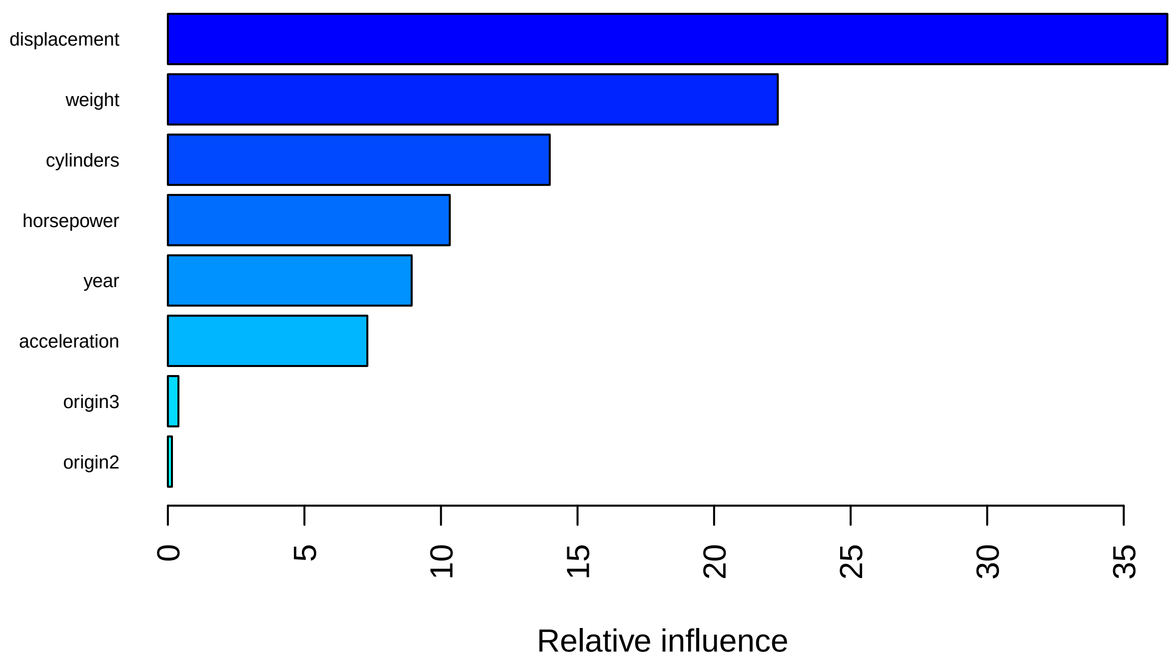
(b) Perform boosting on the training data and report the variable importance. Report the test data performance.

Boosting

```
ctrl=trainControl(method = "cv",
                  classProbs = TRUE,
                  summaryFunction = twoClassSummary)
set.seed(1)
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 ~ . , train2,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)
ggplot(gbmA.fit, highlight = TRUE)
```




```
gbmA.pred <- predict(gbmA.fit, newdata = test2, type = "prob")[,1]
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
```

From the plot, we can see that "Displacement" appears to be the most influential variable, followed by "weight." The variables "origin2" and "origin3" have no bar extending to the right, indicating they have zero or negligible importance in this context. The purpose of the model is not specified, but given the variables, it may be related to vehicles or engines.

Test Performance

```
gbmA.probs <- predict(gbmA.fit, newdata = test2, type = "prob")
roc(response = test2$mpg_cat, predictor = gbmA.probs[, "high"])
```

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

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

```
##
## Call:
## roc.default(response = test2$mpg_cat, predictor = gbmA.probs[,     "high"])
##
## Data: gbmA.probs[, "high"] in 57 controls (test2$mpg_cat high) > 61 cases (test2$mpg_cat low).
## Area under the curve: 0.9888
```

The AUC value is 0.99, which is very close to 1. This indicates an excellent performance of the model on the test data, with high accuracy in differentiating between the 'high' and 'low' categories of the 'mpg_cat' variable. The 'controls' are instances labeled as 'high' and 'cases' as 'low'. An AUC value above 0.9 is typically considered outstanding, suggesting that the model's predicted probabilities (gbmA.probs[, "high"]) are highly effective at ranking the test data instances with a high degree of separation between the two mpg categories.