P8106 Data ScienceII HW4

Yueran Zhang (yz4188)

2023-04-11

1. 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%).

```
library(tidyverse)
library(ISLR)
library(mlbench)
library(caret)
library(rpart)
library(rpart.plot)
library(party)
library(partykit)
library(pROC)
library(randomForest)
library(ranger)
library(gbm)
library(pdp)
```

Import Dataset

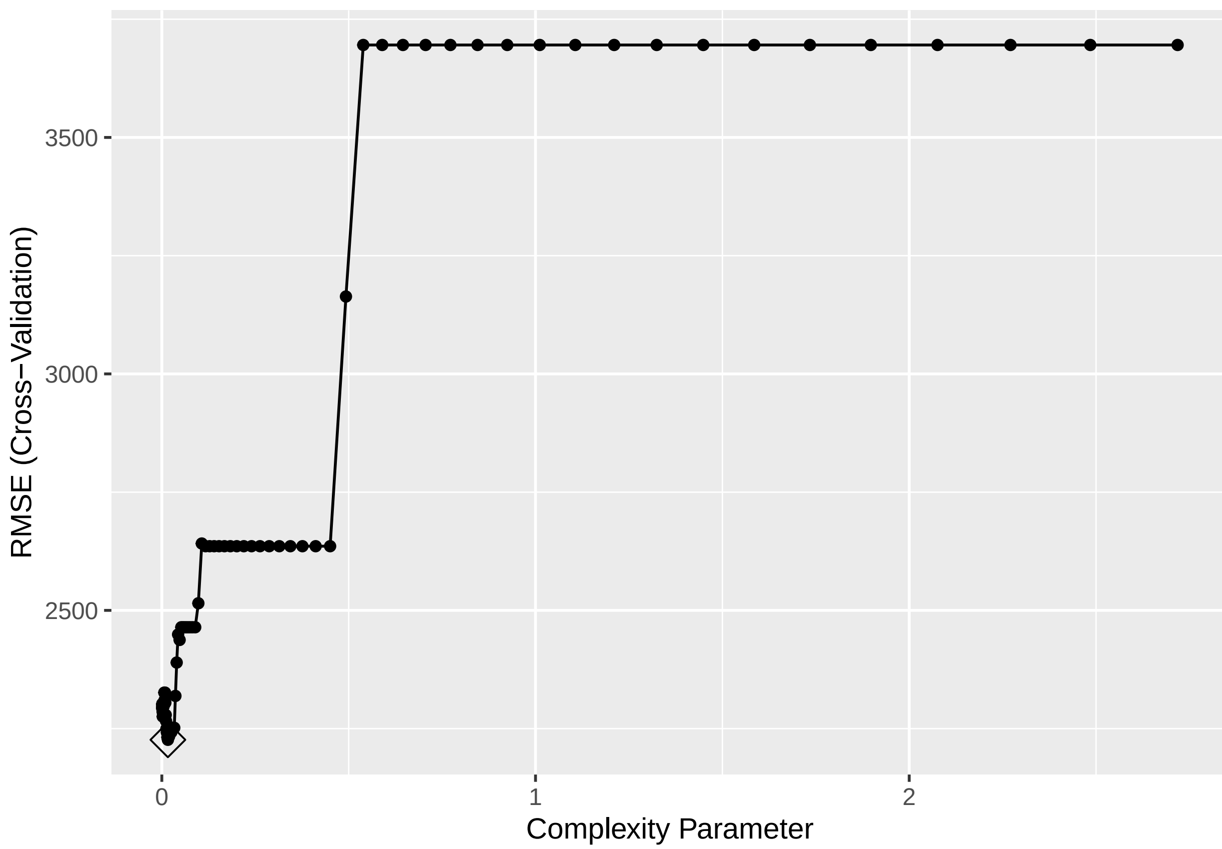
```
set.seed(1234)

# Load dataset + clean data
College = read.csv("/Users/yueranzhang/Desktop/DSII/HW4/DataSet/College.csv")[-1] %>%
janitor::clean_names() %>%
na.omit()

# Data Partition
RowTrain <- createDataPartition(y = College$outstate,
                                p = 0.8,
                                list = FALSE)

training.data <- College[RowTrain,]
test.data <- College[-RowTrain,]
```

```r
# training data
x <- model.matrix(outstate ~. , training.data) [,-1]
y <- training.data$outstate

# test data
x2 <- model.matrix(outstate ~. , test.data)[,-1]
y2 <- test.data$outstate
```

Question A

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

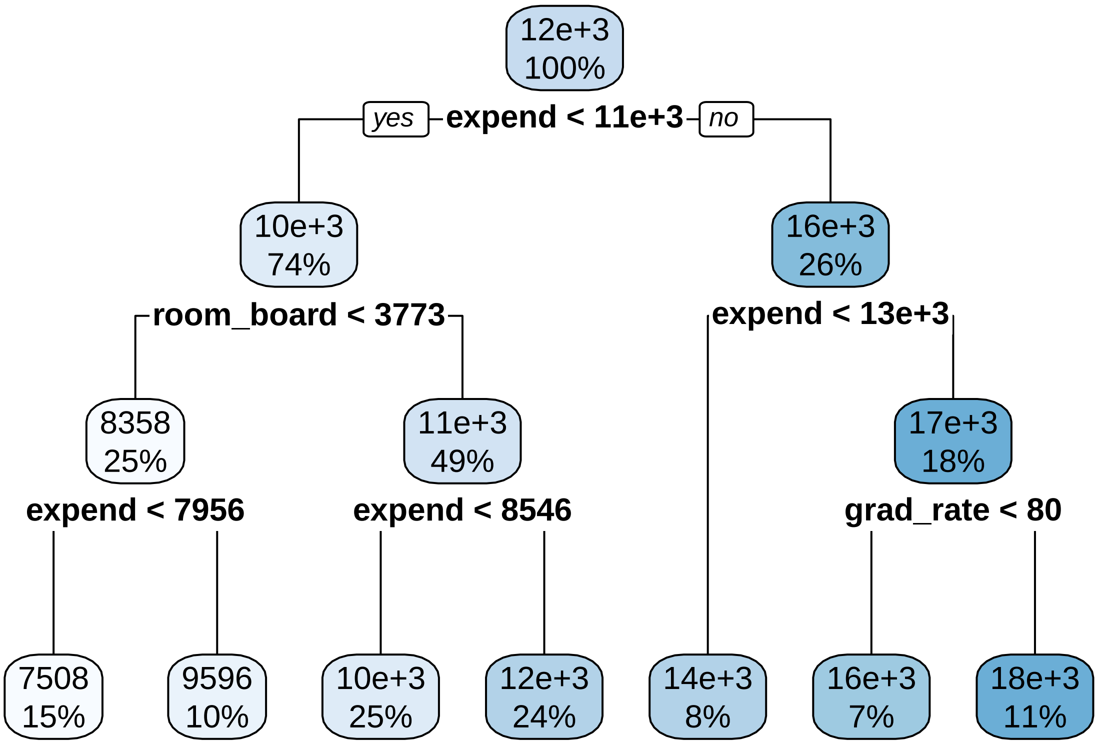
```
ctrl <- trainControl(method = "cv")
set.seed(1234)

# Using Package 'caret' to built the regression tree model

rpart.fit <- train(outstate ~ . ,
                   College[RowTrain,],
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-7,1, length = 90))),
                   trControl = ctrl)


# Report the tuning parameter
ggplot(rpart.fit, highlight = TRUE)
```




```
rpart.fit$bestTune
```

```
##            cp
## 33 0.01618621
```

```r
# Plot the tree
rpart.plot(rpart.fit$finalModel)
```



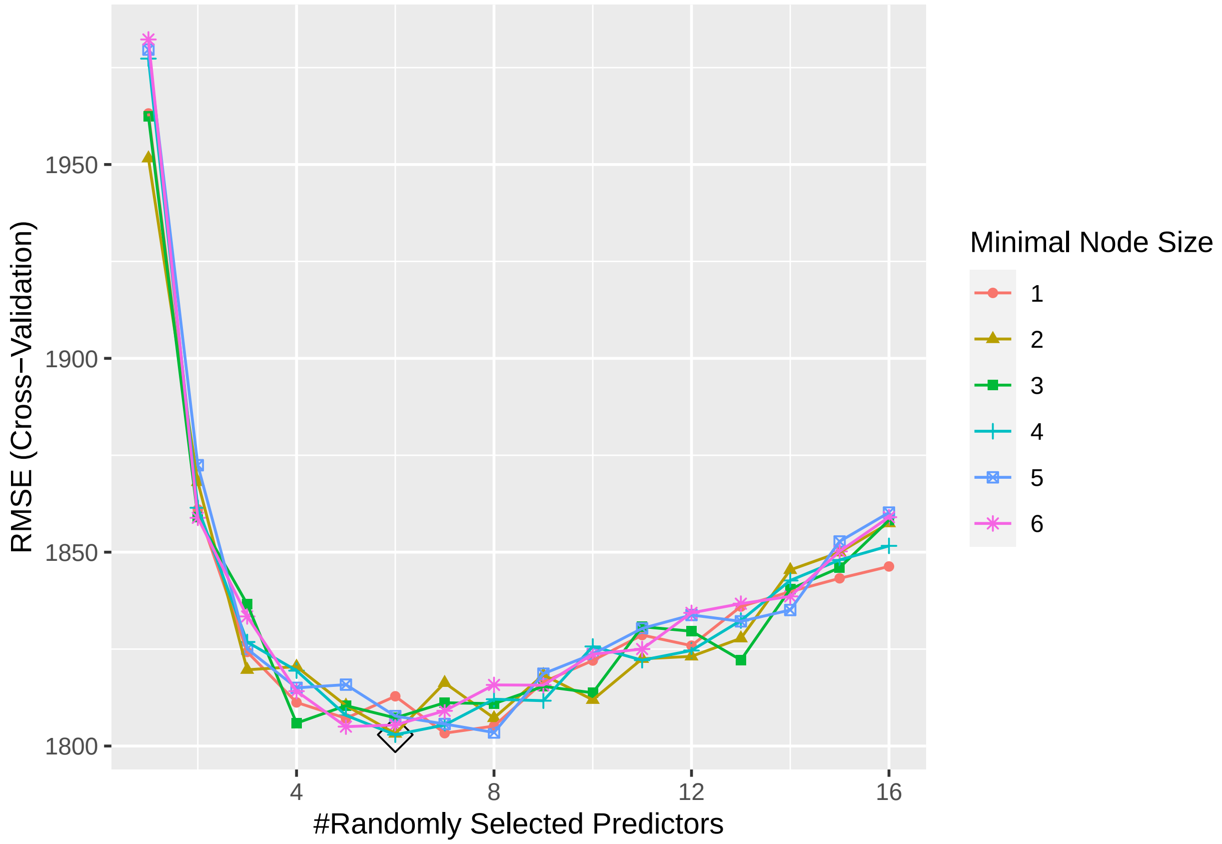
* From the final model, we can report the best tuning parameter `cp` is 0.0161862120750658.

Question B

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

```r
set.seed(1234)

# Using Package 'caret'  to perform random forest

rf.grid <- expand.grid(mtry = 1:16,
                       splitrule = "variance",
                       min.node.size = 1:6)

rf.fit <- train(outstate ~ . ,
                College[RowTrain,],
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl)
```

```r
# Report the tuning parameter
ggplot(rf.fit, highlight = TRUE)
```



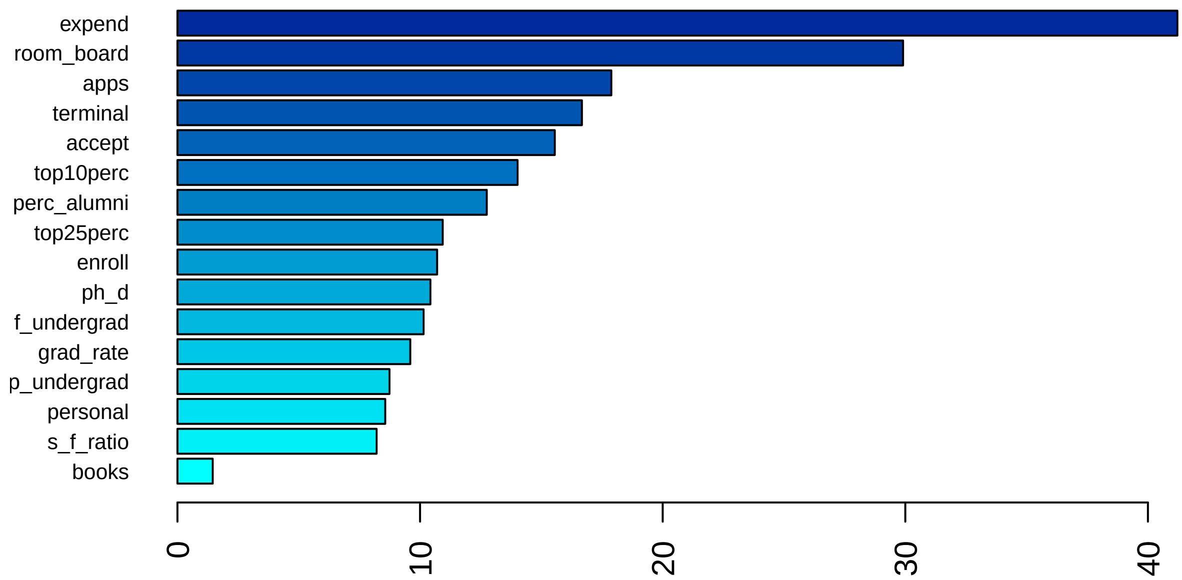

```r
rf.fit$bestTune
```

```
##    mtry splitrule min.node.size
## 34    6  variance             4
```

- From the above output, ww can know that the best tuning parameters selected via CV are `mtry = 6`, `splitrule = variance` and `min.node.size = 4`.

```r
set.seed(1234)

# We can extract the variable importance from the fitted models.

rf.final.per <- ranger(outstate ~ . ,
                       College[RowTrain,],
                        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","darkblue"))(19))
```



- We see that the variables `expend(Instructional expenditure per student)`, `Room_Board(Room and board costs)` and `apps(Number of applications received)` are the top 3 from the variable importance.

```
set.seed(1234)

pred.rf <- predict(rf.fit, newdata = College[-RowTrain,])
test.error.rf <- RMSE(pred.rf, College$outstate[-RowTrain])
test.error.rf
```

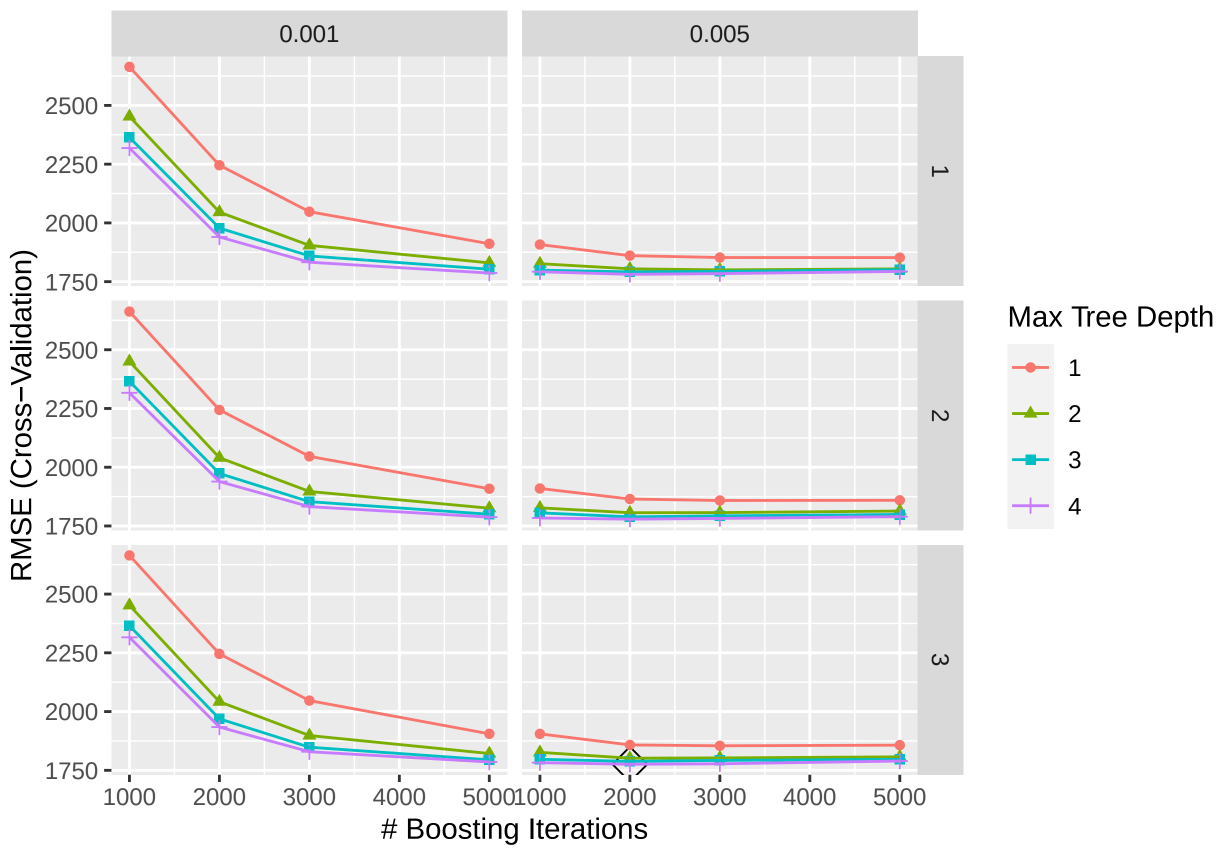
```
## [1] 1565.881
```

- The test error is 1565.880541.

Question C

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

```r
# Use Package "caret" to implement boosting framework

set.seed(1234)

gbm.grid <- expand.grid(n.trees = c(1000, 2000, 3000, 5000),
                        interaction.depth = 1:4,
                        shrinkage = c(0.005,0.001),
                        n.minobsinnode = c(1:3))


gbm.fit <- train(outstate ~ . ,
                 College[RowTrain,],
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 verbose = FALSE)

# Report the tuning parameter
ggplot(gbm.fit, highlight = TRUE)
```

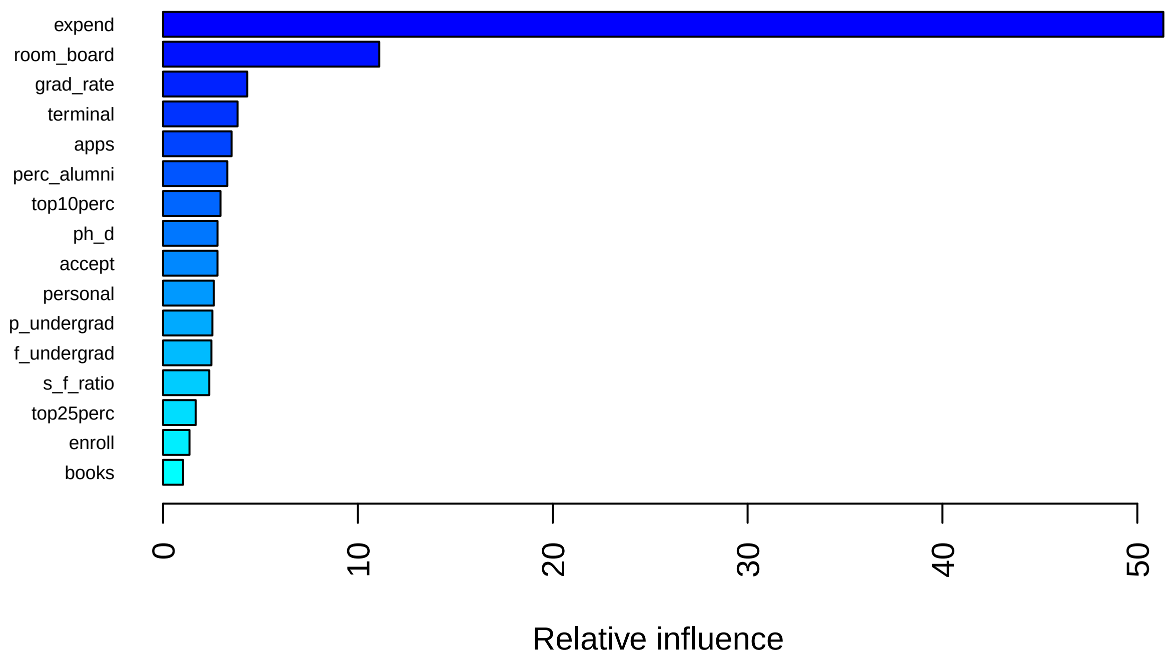



```r
gbm.fit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 94    2000                 4     0.005              3
```

- The best tuning parameters are `n.trees = 2000`, `interaction.depth = 4`, `shrinkage = 0.005` and `nminobsinode = 3`.

```
set.seed(1234)

# Variable importance from boosting can be obtained using the `summary()` function.

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




```
##                     var   rel.inf
## expend           expend 51.335795
## room_board   room_board 11.099189
## grad_rate     grad_rate  4.323892
## terminal       terminal  3.823519
## apps               apps  3.517553
## perc_alumni perc_alumni  3.299188
## top10perc     top10perc  2.948734
## ph_d               ph_d  2.795676
## accept           accept  2.794902
## personal       personal  2.609288
## p_undergrad p_undergrad  2.532933
## f_undergrad f_undergrad  2.481886
## s_f_ratio     s_f_ratio  2.372804
## top25perc     top25perc  1.679169
## enroll           enroll  1.357956
## books             books  1.027515
```

- We see that the variables `expend(Instructional expenditure per student)`, `Room_Board(Room and board costs)` and `grad_rate(Graduation rate)` are the top 3 from the variable importance, which are slightly different from what we got from random forest model.

```
set.seed(1234)

pred.glm <- predict(gbm.fit, newdata = College[-RowTrain,])
test.error.glm <- RMSE(pred.glm, College$outstate[-RowTrain])
test.error.glm
```

```
## [1] 1453.169
```

- TheThe test error is 1453.1694807.

2. This problem involves the OJ data in the ISLR package. The data contains 1070 purchases where the customers either purchased Citrus Hill or Minute Maid Orange Juice. A number of characteristics of customers and products are recorded. Create a training set containing a random sample of 700 observations, and a test set containing the remaining observations.

```
set.seed(1234)

# Build a classification tree using the training data, with Purchase as the response and the other vari

data(OJ)
OJ <- na.omit(OJ)

OJ$Purchase <- factor(OJ$Purchase, c("CH","MM"))


RowTrain.OJ <- createDataPartition(y = OJ$Purchase,
                                   p = 0.653,
                                   list = FALSE)

# training dataset
training.data.OJ <- OJ[RowTrain.OJ, ]

# test dataset
test.data.OJ <- OJ[-RowTrain.OJ, ]
```

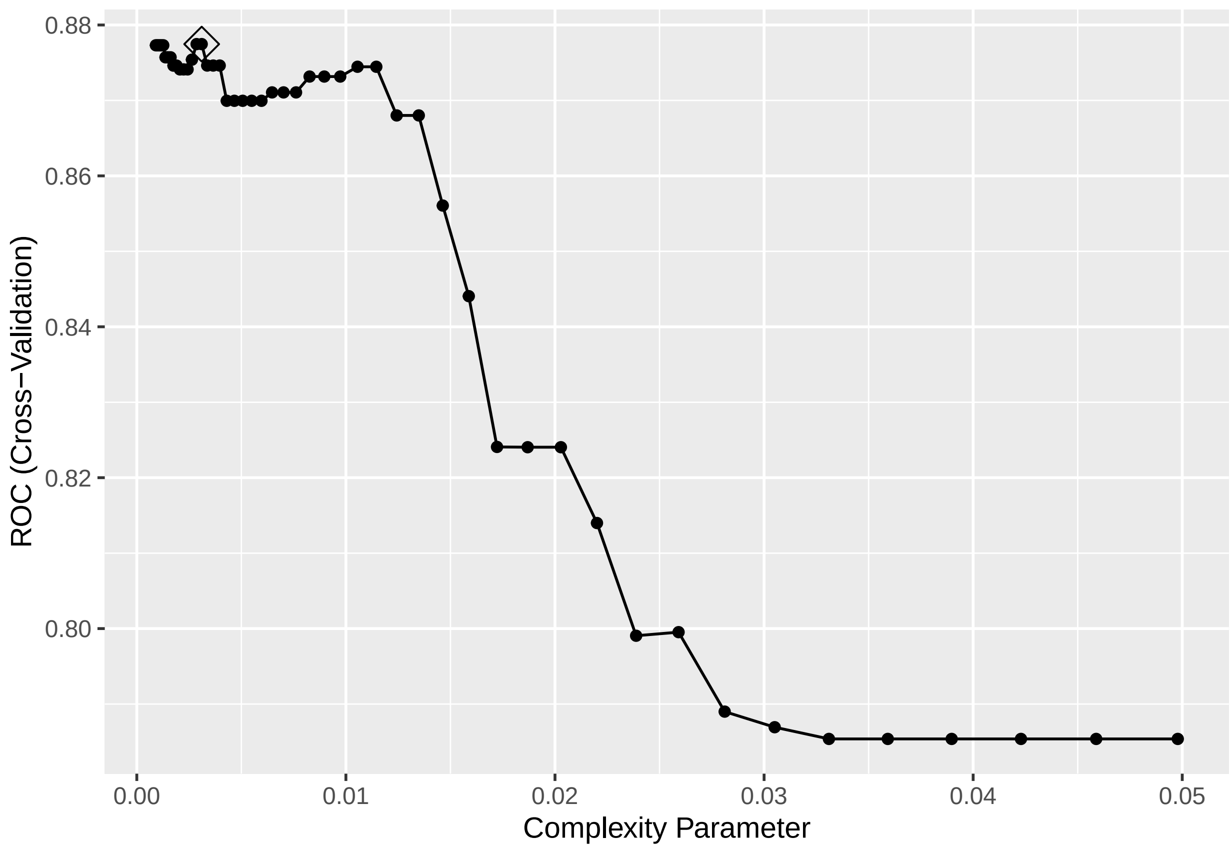
Question A

Build a classification tree using the training data, with Purchase as the response and the other variables as predictors. Which tree size corresponds to the lowest cross-validation error? Is this the same as the tree size obtained using the 1 SE rule?

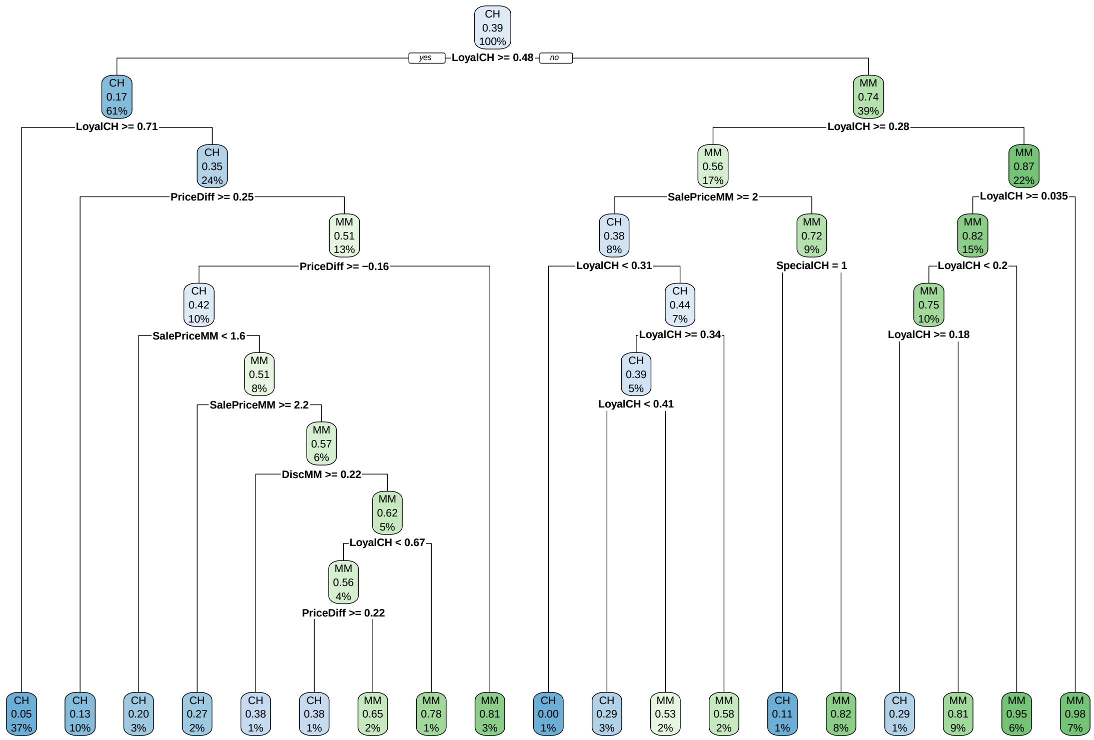
```


##############################
# Lowest cross-validation error
##############################

set.seed(1234)

ctrl <- trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

rpart.fit <- train(Purchase ~ . ,
                   OJ,
                   subset = RowTrain.OJ,
                   method = "rpart",
```

```
                    tuneGrid = data.frame(cp = exp(seq(-7,-3, len = 50))),
                    trControl = ctrl,
                    metric = "ROC")

#Plot for Complexity Parameter selection
ggplot(rpart.fit, highlight = TRUE, height = 30, width = 21, units = "cm")
```

We use Package caret to fit the classification tree, and plot with the final tree.




```
# Tree plot with the lowest cross-validation error
rpart.plot(rpart.fit$finalModel)
```



- From the above using lowest cross-validation error tree plot, the tree size is 19.

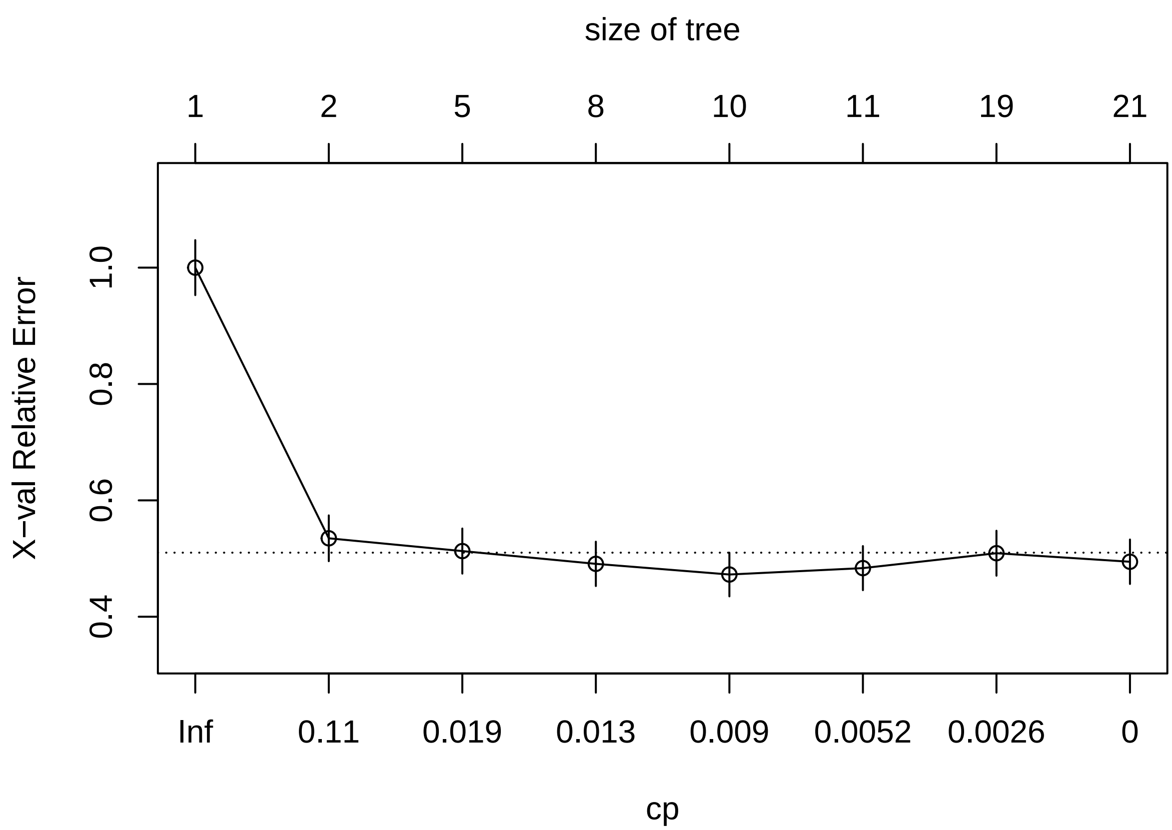
```r
###############################
# 1 SE Rule
###############################

set.seed(1234)
tree.1se <- rpart(formula = Purchase ~ . ,
                  data = OJ,
                  subset = RowTrain.OJ,
                  control = rpart.control(cp = 0))

cpTable <- printcp(tree.1se)
```

We use Package rpartfor 1SE rule.

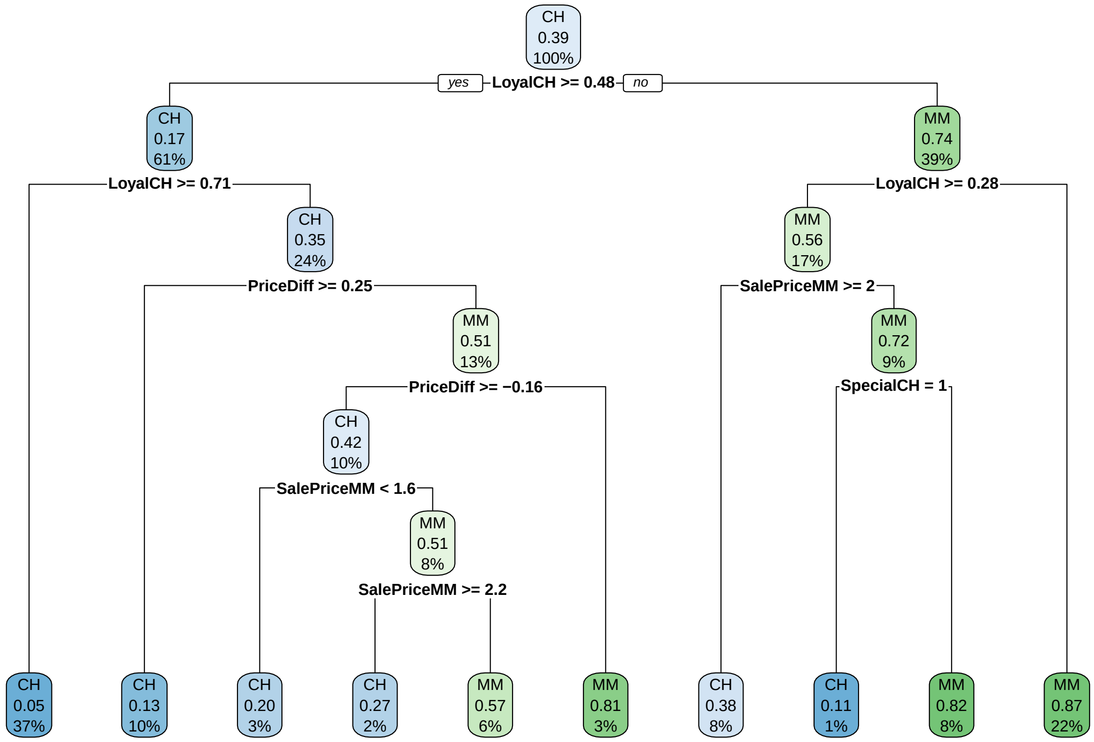
```
##
## Classification tree:
## rpart(formula = Purchase ~ ., data = OJ, subset = RowTrain.OJ,
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
```

```
## [1] DiscMM      LoyalCH     PriceDiff    SalePriceMM SpecialCH    StoreID
##
## Root node error: 273/700 = 0.39
##
## n= 700
##
##          CP nsplit rel error  xerror     xstd
## 1 0.4761905      0   1.00000 1.00000 0.047270
## 2 0.0238095      1   0.52381 0.53480 0.039375
## 3 0.0158730      4   0.45055 0.51282 0.038766
## 4 0.0109890      7   0.40293 0.49084 0.038128
## 5 0.0073260      9   0.38095 0.47253 0.037575
## 6 0.0036630     10   0.37363 0.48352 0.037910
## 7 0.0018315     18   0.34432 0.50916 0.038661
## 8 0.0000000     20   0.34066 0.49451 0.038237
```

```
plotcp(tree.1se)
```




```
#Tree plot with 1SE
minErr <- which.min(cpTable[,4])
tree.1se2<- prune(tree.1se, cp = cpTable[minErr,1])
rpart.plot(tree.1se2)
```



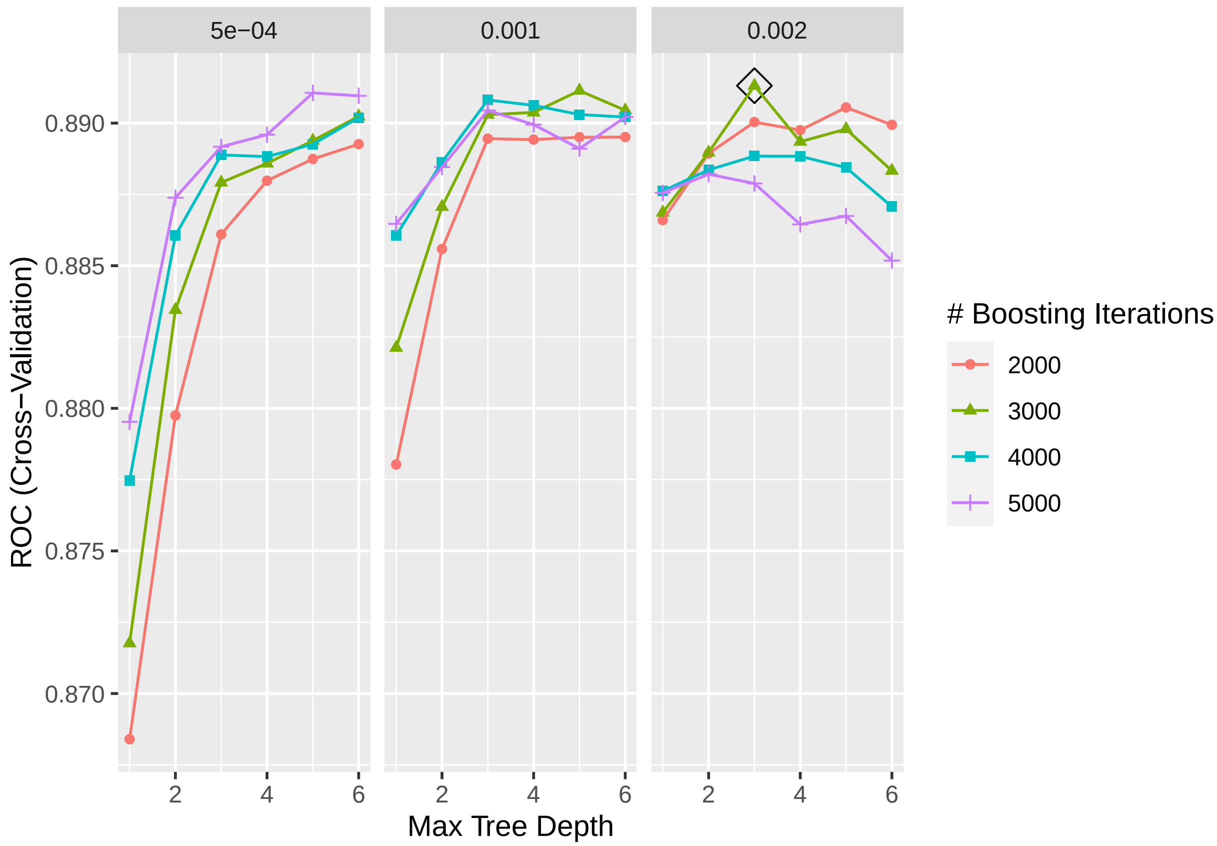
- Using the 1 SE rule, we can see the tree size now is 10. The tree size obtained by using cross-validation is different from the tree size obtained by using 1 SE rule.

Question B

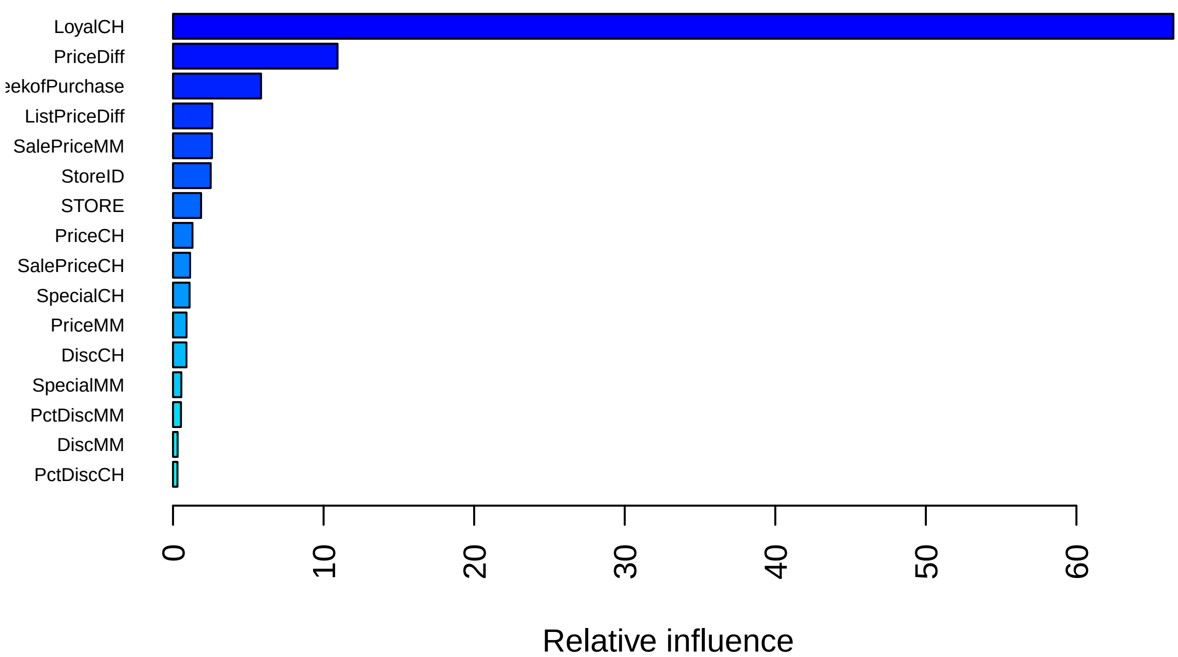
Perform boosting on the training data and report the variable importance. What is the test error rate?

```
set.seed(1234)

gbmA.grid <- expand.grid(n.trees = c(2000,3000,4000,5000),
                         interaction.depth = 1:6,
                         shrinkage = c(0.0005,0.001,0.002),
                         n.minobsinnode = 1)

gbmA.fit <- train(Purchase ~ . ,
                  OJ,
                  subset = RowTrain.OJ,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
```

```
                 verbose = FALSE)

ggplot(gbmA.fit, highlight = TRUE)
```




```
#Variable Importance
summary(gbmA.fit$finalModel, las = 2, cBars = 16, cex.names = 0.6)
```




```
##                           var    rel.inf
## LoyalCH               LoyalCH 66.4381784
## PriceDiff           PriceDiff 10.9258779
## WeekofPurchase WeekofPurchase  5.8452419
## ListPriceDiff   ListPriceDiff  2.6085816
## SalePriceMM       SalePriceMM  2.5849111
## StoreID               StoreID  2.5027718
## STORE                   STORE  1.8622146
## PriceCH               PriceCH  1.2915286
## SalePriceCH       SalePriceCH  1.1325304
## SpecialCH           SpecialCH  1.0984586
## PriceMM               PriceMM  0.8951909
## DiscCH                 DiscCH  0.8890760
## SpecialMM           SpecialMM  0.5504536
## PctDiscMM           PctDiscMM  0.5264604
## DiscMM                 DiscMM  0.3056835
## PctDiscCH           PctDiscCH  0.2941778
## Store7Yes           Store7Yes  0.2486631
```

- We see that the variables `LoyalCH`, `PriceDiff` and `ekofPyrchase` are the top 3 from the variable importance.

```
gbmA.pred <- predict(gbmA.fit, newdata = OJ[-RowTrain.OJ,], type = "raw")
error.rate.gbmA <- mean(gbmA.pred != OJ$Purchase[-RowTrain.OJ])
```

- The test error rate is 0.1432432.