P8106-hw4

Renjie Wei rw2844

4/10/2022

```
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)
library(ggplot2)
```

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

```
# I removed the id variable `College`
data <- read.csv("College.csv")[,-1]
data <- na.omit(data)
set.seed(2022)
# sum(is.na(data)) = 0
rowTrain <- createDataPartition(y = data$Outstate, p = 0.8, list = FALSE)
```

Problem 1

Part A

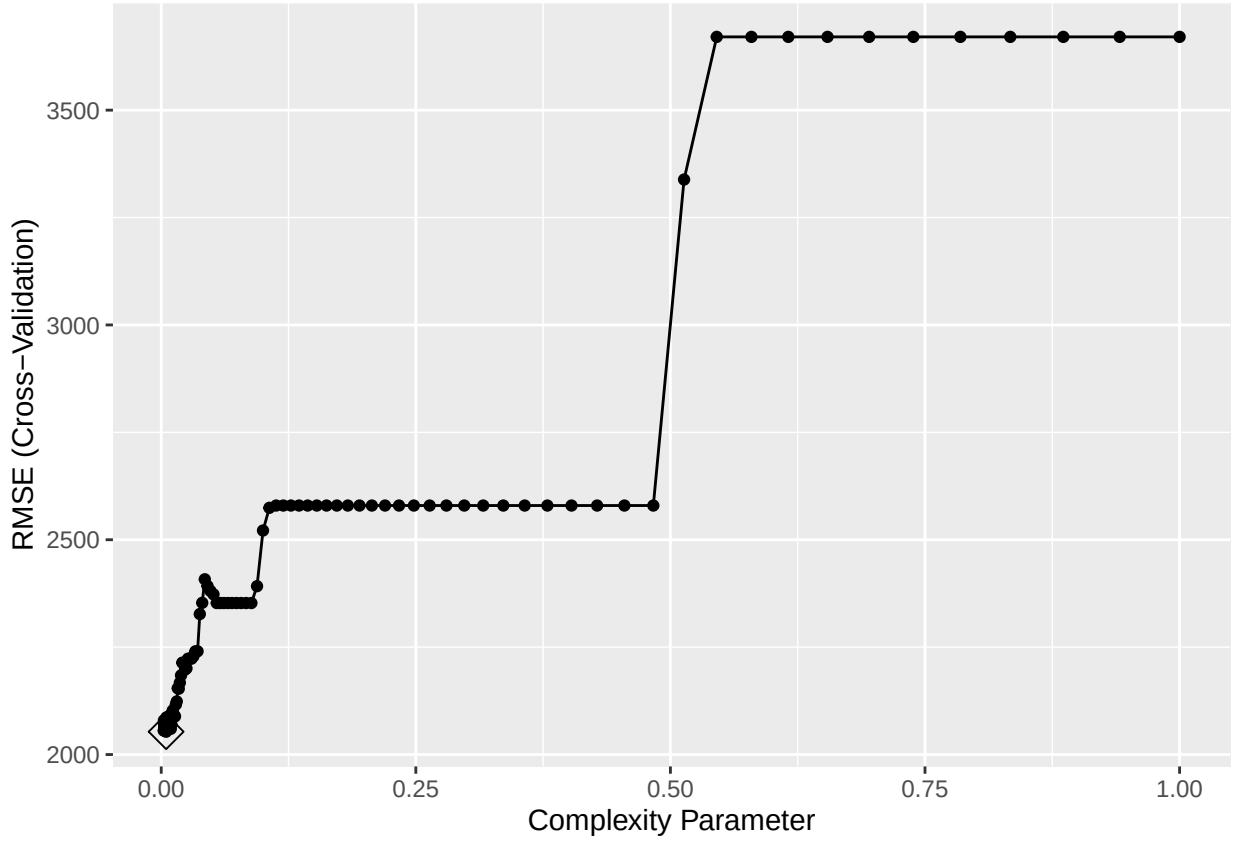
Build a regression tree on the training data to predict the response. Create a plot of the tree.

Answers

First, I built the regression tree model using `caret`.

```
ctrl <- trainControl(method = "cv")

set.seed(2022)
rpart.fit <- train(Outstate ~ . ,
                   data[rowTrain,],
                   method = "rpart",
```

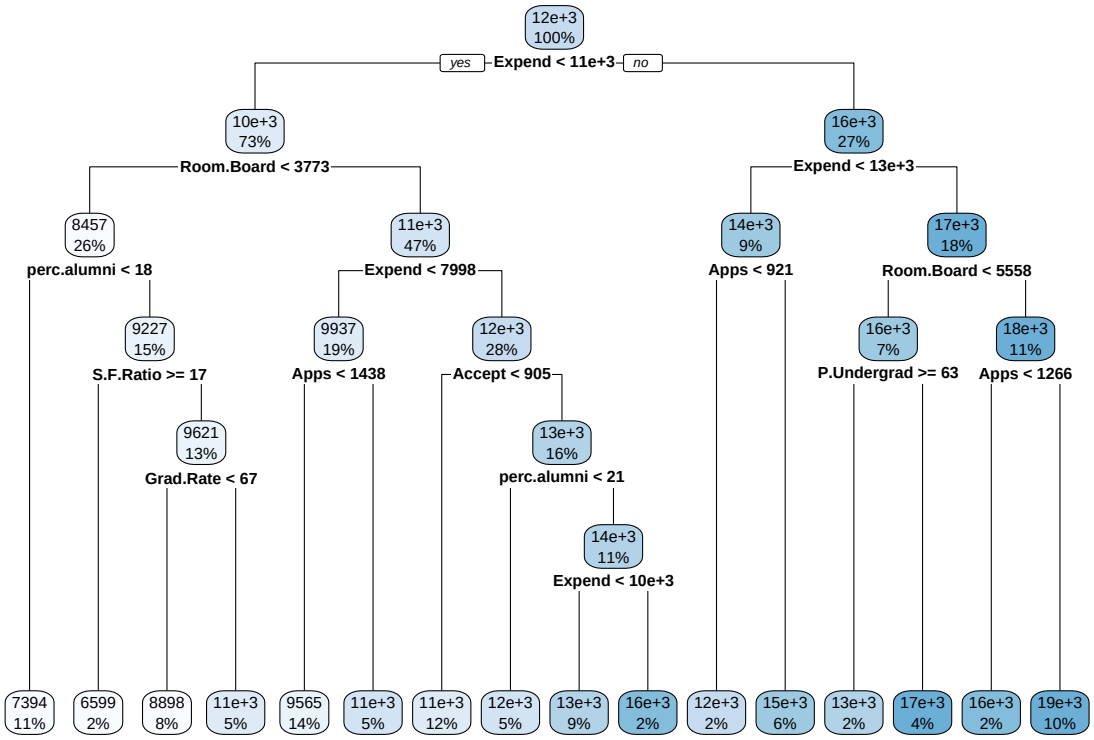
```
                    tuneGrid = data.frame(cp = exp(seq(-6,0, length = 100))),
                    trControl = ctrl)
ggplot(rpart.fit, highlight = TRUE)
```




```
# cp = 0.00482795
```

From the final model, we know that the best tuning parameter `cp` is 0.0048279. And the following plot shows the final tree.

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



Part B

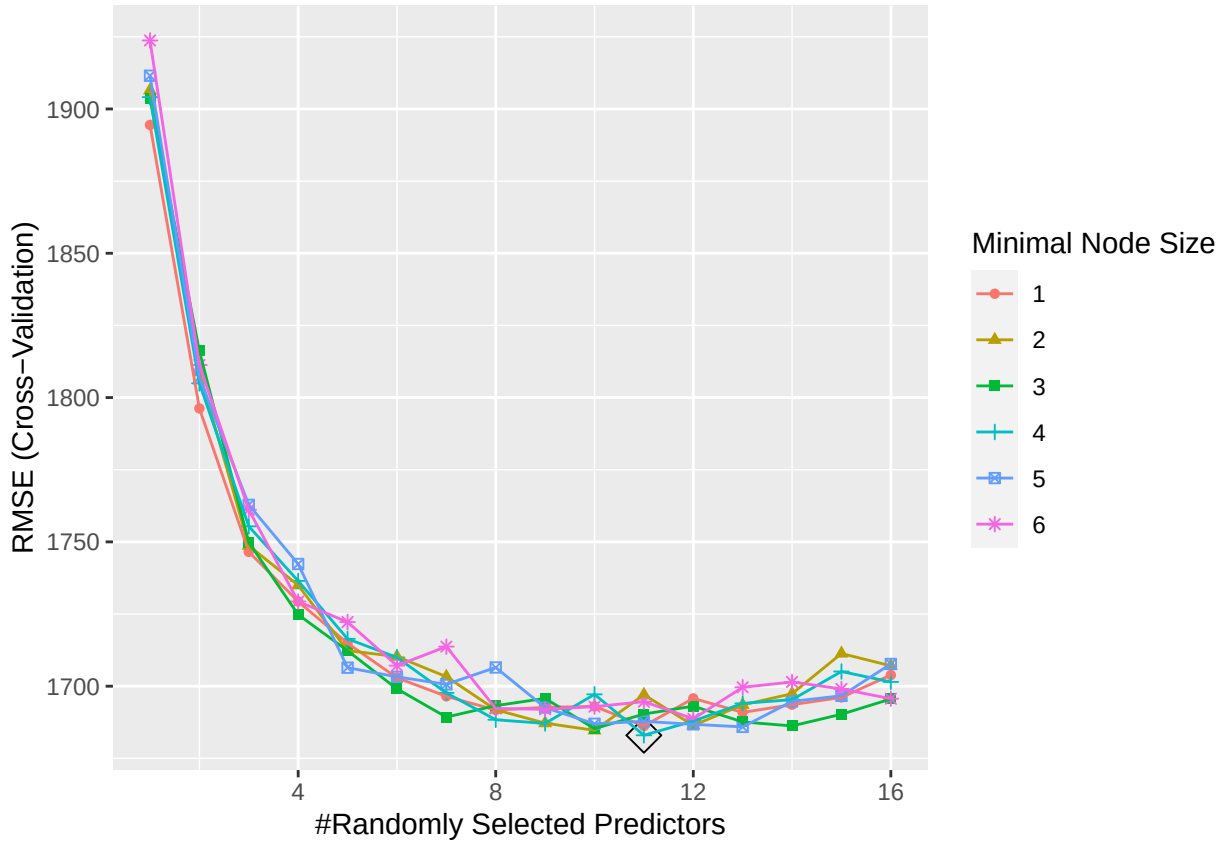
Perform random forest on the training data. Report the variable importance and the test error.

Answers

I built the random forest model using `caret`.

```
rf.grid <- expand.grid(mtry = 1:16,
                       splitrule = "variance",
                       min.node.size = 1:6)
set.seed(2022)

library(parallel)
no_cores <- detectCores() - 1
library(doParallel)
cl <- makePSOCKcluster(no_cores)
registerDoParallel(cl)

rf.fit <- train(Outstate ~ . ,
                data,
                subset = rowTrain,
                method = "ranger",
                tuneGrid = rf.grid,
```

```
                trControl = ctrl)

stopCluster(cl)
registerDoSEQ()
ggplot(rf.fit, highlight = TRUE)
```



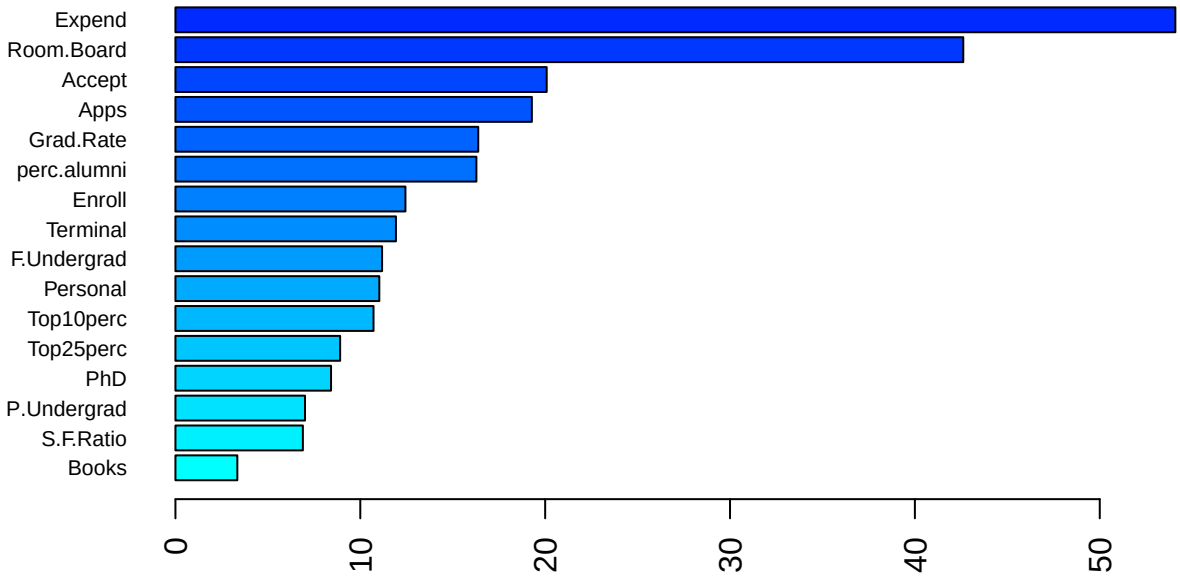

```
rf.fit$bestTune
```

```
##    mtry splitrule min.node.size
## 64   11  variance             4
```

The best tuning parameters are `mtry = 11`, `splitrule = variance` and `min.node.size = 4`.

I extracted the variable importance from the fitted model by permutation.

```
rf.final.per <- ranger(Outstate ~ . ,
                       data[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","blue"))(19))
```



From the variable importance plot we can see that `Expend`, `Room.Board` and `Accept` are the top 3 most important variables.

```
rf.predict <- predict(rf.fit, newdata = data[-rowTrain,])
rf.RMSE <- RMSE(rf.predict, data$Outstate[-rowTrain])
```

The RMSE of test set is 1960.044.

Part C

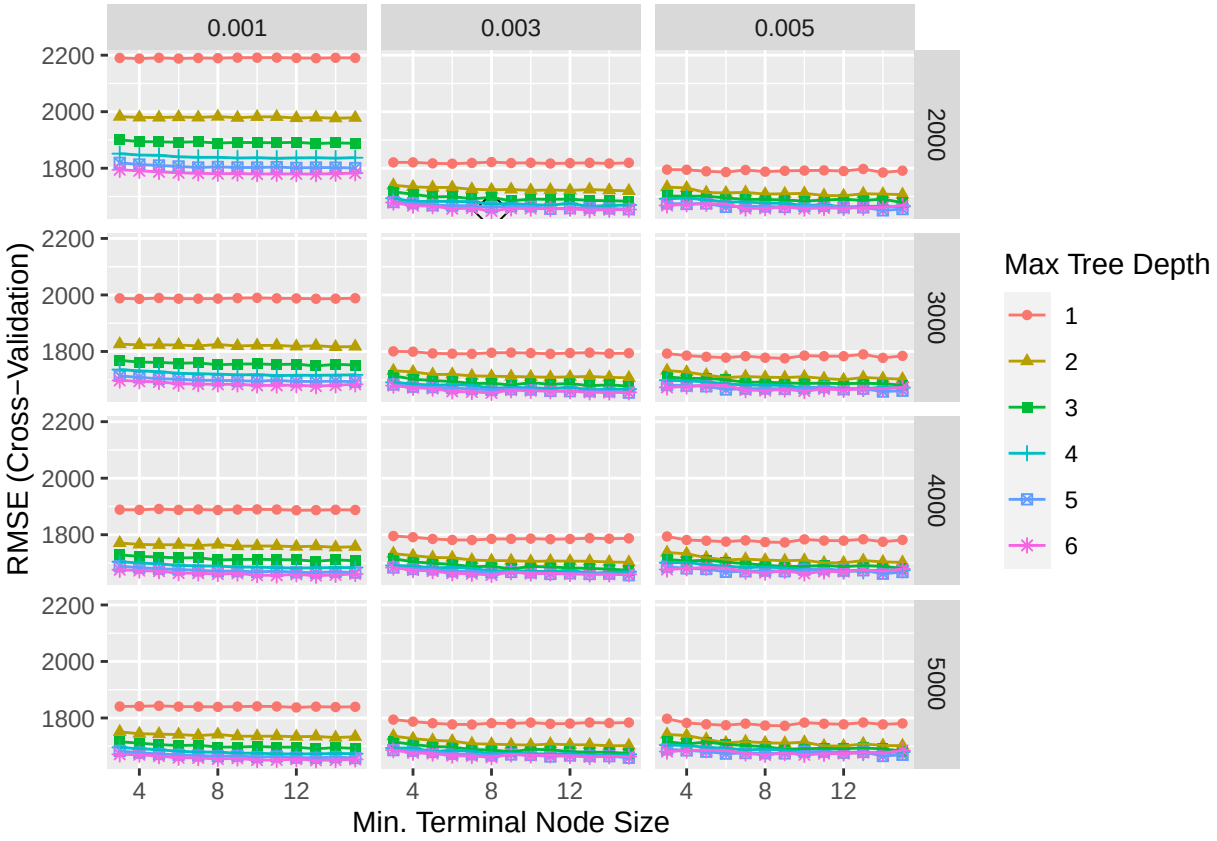
Perform boosting on the training data. Report the variable importance and the test error.

Answers

I built the boosting model using `caret`.

```
gbm.grid <- expand.grid(n.trees = c(2000, 3000, 4000, 5000),
                        interaction.depth = 1:6,
                        shrinkage = seq(0.001, 0.005, by = 0.002),
                        n.minobsinnode = c(3:15))

set.seed(2022)

no_cores <- detectCores() - 1
```

```
cl <- makePSOCKcluster(no_cores)
registerDoParallel(cl)

gbm.fit <- train(Outstate ~ . ,
                 data[rowTrain,],
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 verbose = FALSE)
stopCluster(cl)
registerDoSEQ()
ggplot(gbm.fit, highlight = TRUE)
```



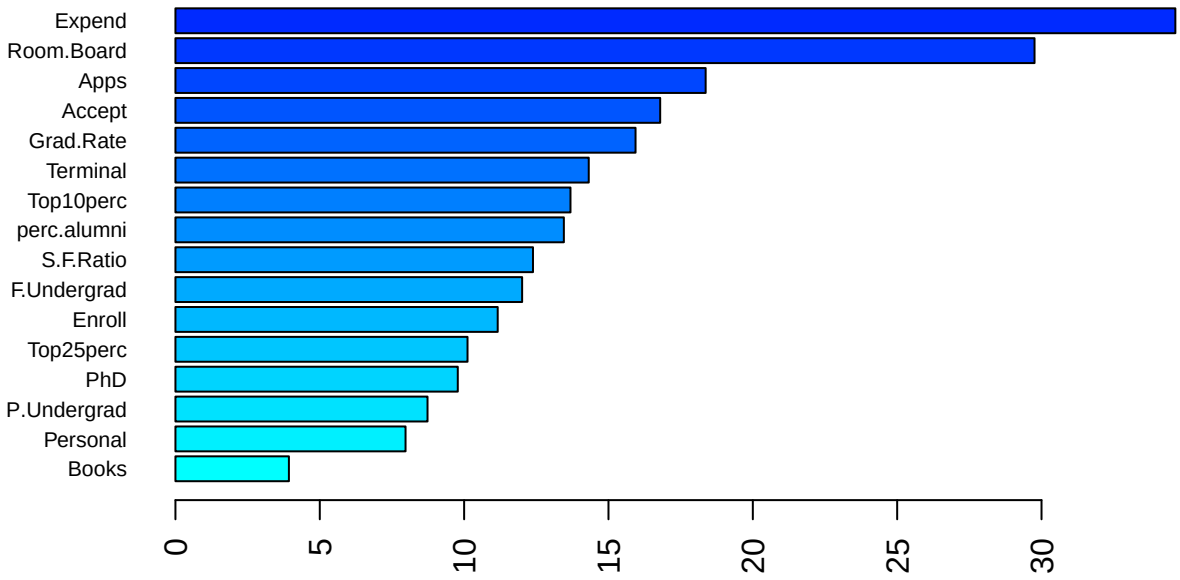

```
gbm.fit$bestTune
```

```
##     n.trees interaction.depth shrinkage n.minobsinnode
## 593    2000                 6     0.003              8
```

The best tuning parameters are `n.trees = 2000`, `interaction.depth = 6`, `shrinkage = 0.003` and `nminobsinode = 8`.

I extracted the variable importance from the fitted model by permutation.

```
gbm.final.per <- ranger(Outstate ~ . ,
                        data[rowTrain,],
                        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))
```



From the variable importance plot we can see that `Expend`, `Room.Board` and `Apps` are the top 3 most important variables, which are slightly different from what we got from random forest model.

```
gbm.predict <- predict(gbm.fit, newdata = data[-rowTrain,])
gbm.RMSE <- RMSE(gbm.predict, data$Outstate[-rowTrain])
```

The RMSE of test set is 1898.73.

Problem 2

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

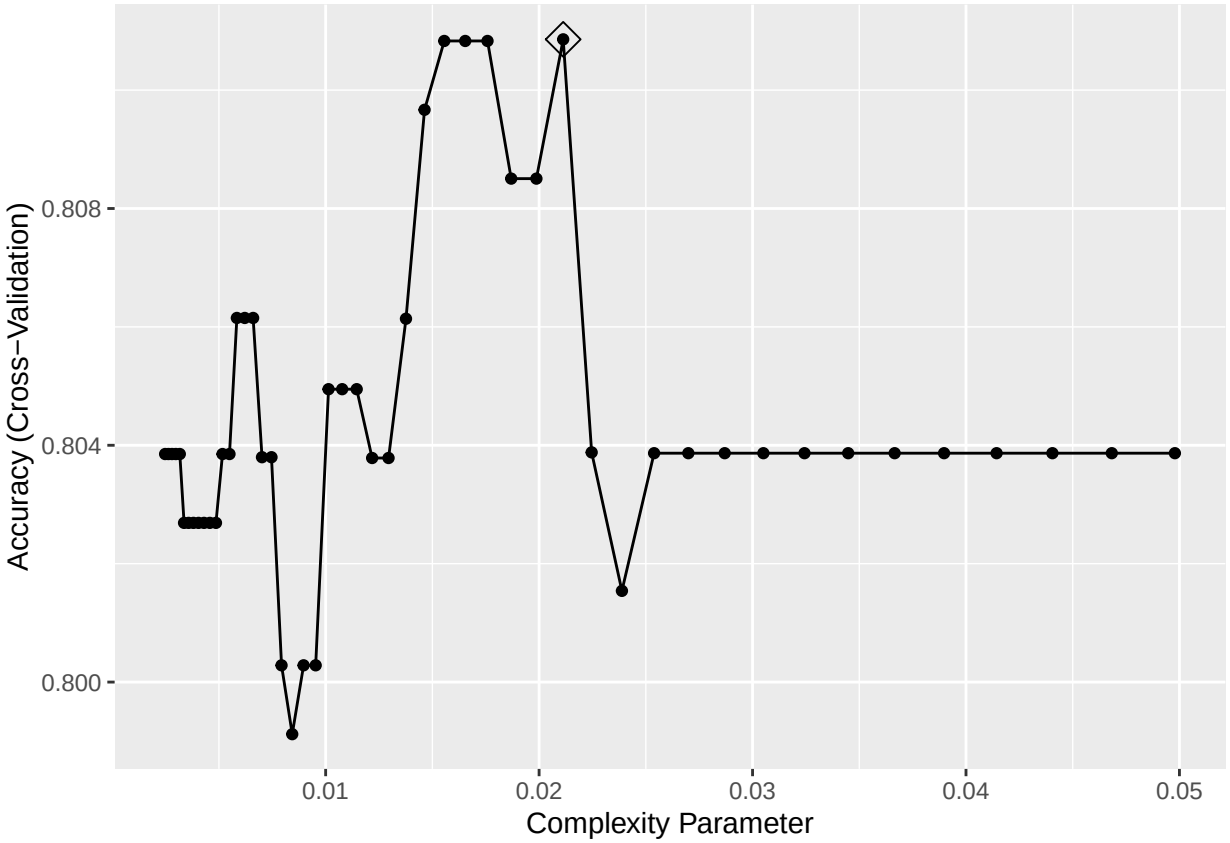
```
data(OJ)
OJ <- na.omit(OJ)
set.seed(2022)
# sum(is.na(data)) = 0
rowTrain1 <- createDataPartition(y = OJ$Purchase, p = 0.8, list = FALSE)
```

Part A

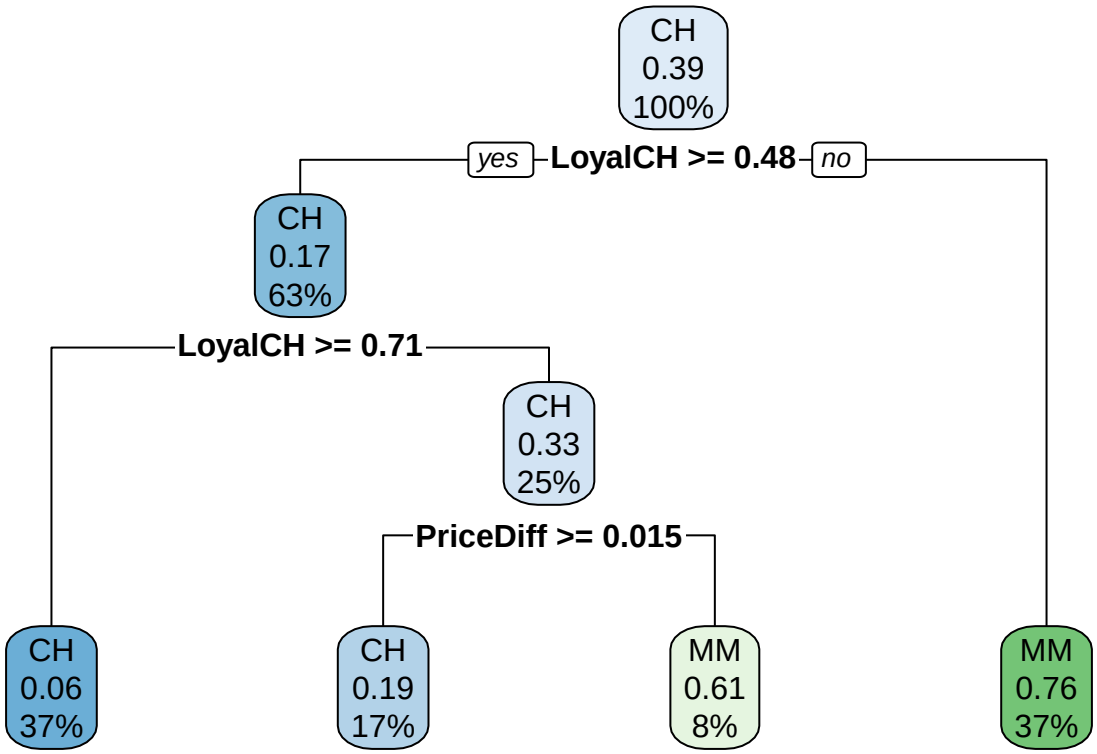
Build a classification tree using the training data, with `Purchase` as the response and the other variables as predictors. Use cross-validation to determine the tree size and create a plot of the final tree. Which tree size corresponds to the lowest cross-validation error? Is this the same as the tree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

I use `caret` to fit the classification tree, and plot the final tree. Mentioning that I use `metric = "Accuracy"` to make the model from `caret` comparable to the model from `rpart`. And I plot the final tree.

```
ctrl1 <- trainControl(method = "cv",
                      classProbs = TRUE)
set.seed(2022)
rpart.fit.OJ <- train(Purchase ~ . ,
                    OJ,
                    subset = rowTrain1,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-6,-3, len = 50))),
                    trControl = ctrl1,
                    metric = "Accuracy")
ggplot(rpart.fit.OJ, highlight = TRUE)
```




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



Using the lowest cross-validation error, we can see the tree size = 4.

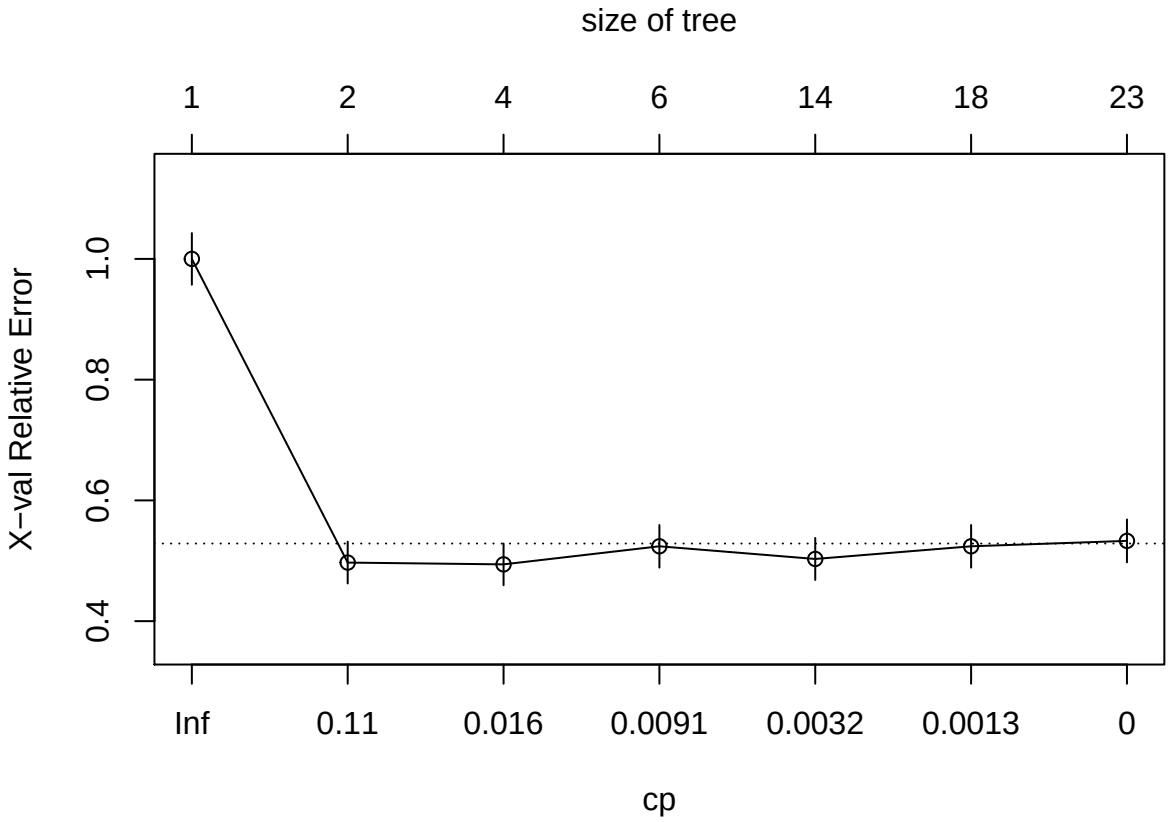
For the implementation of 1SE rule, I use `rpart` to fit the model. And plot the pruned tree under 1SE rule.

```
set.seed(2022)
tree1 <- rpart(formula = Purchase ~ . ,
               data = OJ,
               subset = rowTrain1,
               control = rpart.control(cp = 0))

cpTable <- printcp(tree1)
```

```
##
## Classification tree:
## rpart(formula = Purchase ~ ., data = OJ, subset = rowTrain1,
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
##  [1] ListPriceDiff  LoyalCH        PctDiscCH      PriceDiff      PriceMM
##  [6] SalePriceCH    SalePriceMM    SpecialCH      STORE          StoreID
## [11] WeekofPurchase
##
## Root node error: 334/857 = 0.38973
##
## n= 857
##
```

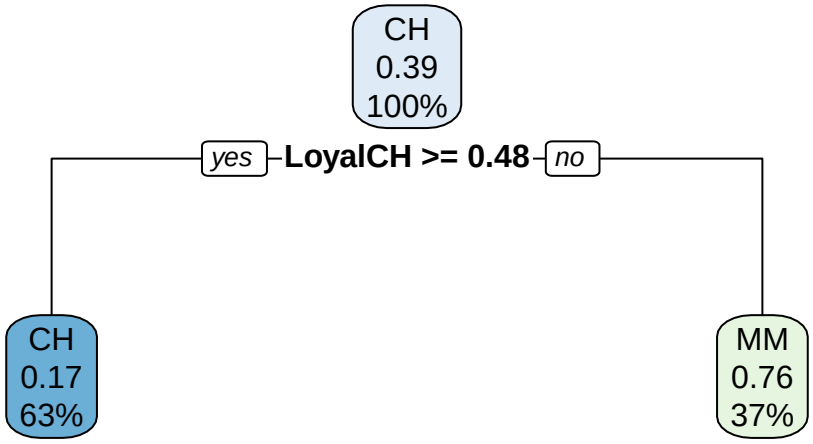
```
##          CP nsplit rel error  xerror     xstd
## 1 0.5029940      0   1.00000 1.00000 0.042745
## 2 0.0224551      1   0.49701 0.49701 0.034638
## 3 0.0119760      3   0.45210 0.49401 0.034559
## 4 0.0069860      5   0.42814 0.52395 0.035332
## 5 0.0014970     13   0.35629 0.50299 0.034796
## 6 0.0011976     17   0.35030 0.52395 0.035332
## 7 0.0000000     22   0.34431 0.53293 0.035556
```

```
plotcp(tree1)
```

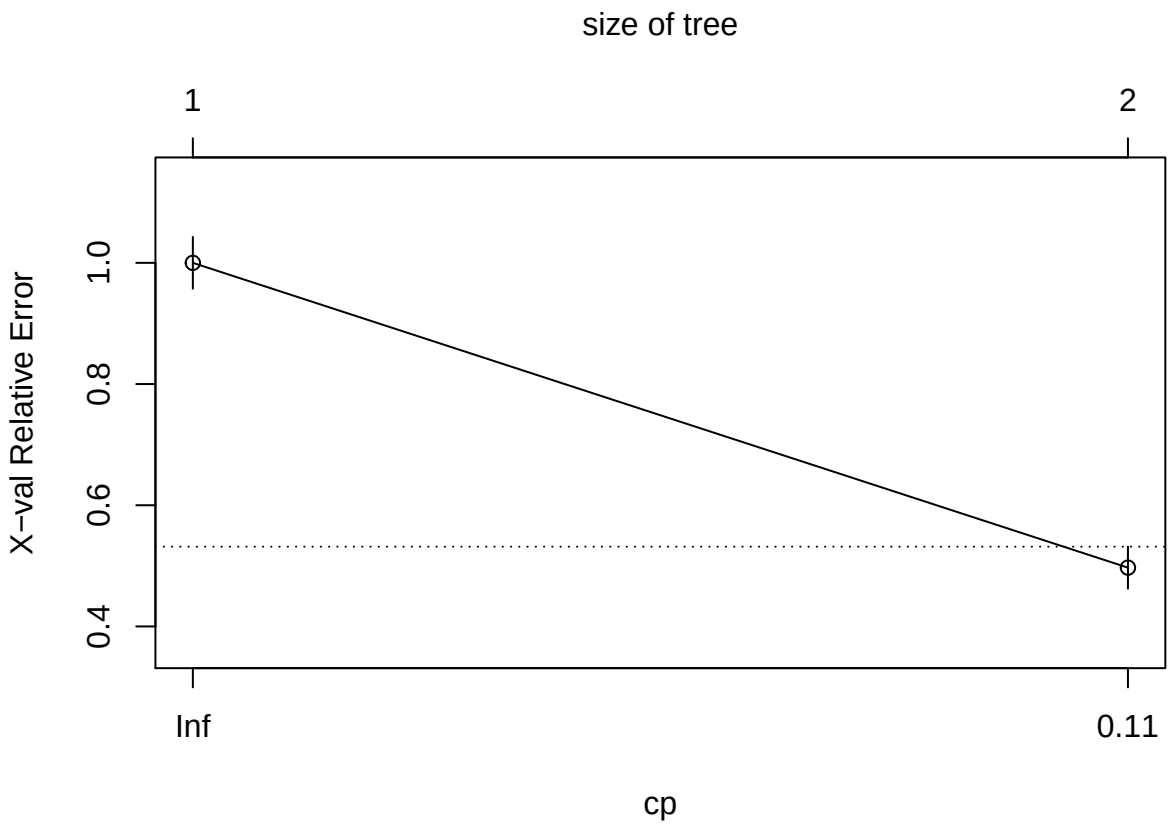



```
#rpart.plot(tree1)
```

```
# 1SE rule
minErr <- which.min(cpTable[,4])
tree2 <- prune(tree1,cp = cpTable[cpTable[,4]<cpTable[minErr,4]+cpTable[minErr,5],1][1])
rpart.plot(tree2)
```




```
plotcp(tree2)
```



From the cp plot and cp table, using the 1 SE rule, we can see the tree size = 2.

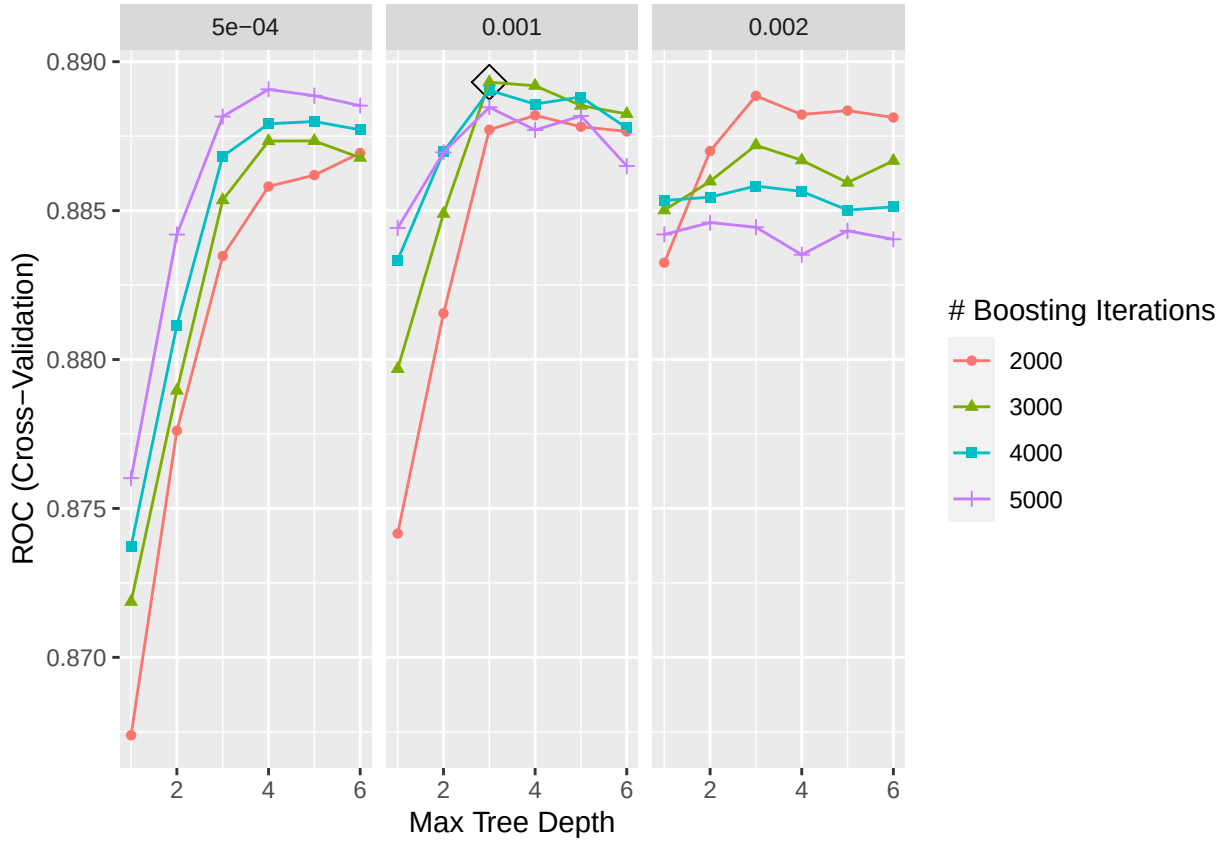
The tree size obtained by using cross validation is different from the tree size obtained by using 1 SE rule.

Part B

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

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


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)
set.seed(2022)

no_cores <- detectCores() - 1
cl <- makePSOCKcluster(no_cores)
registerDoParallel(cl)

gbmA.fit <- train(Purchase ~ . ,
                  OJ,
                  subset = rowTrain1,
```

```
                  tuneGrid = gbmA.grid,
                  trControl = ctrl2,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)

stopCluster(cl)
registerDoSEQ()

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




```
gbmA.pred <- predict(gbmA.fit, newdata = OJ[-rowTrain1,], type = "raw")
error.rate.gbmA <- mean(gbmA.pred != OJ$Purchase[-rowTrain1])
```

```
summary(gbmA.fit$finalModel, las = 2, cBars = 19, cex.names = 0.6) %>% knitr::kable(digits = 3, caption
```

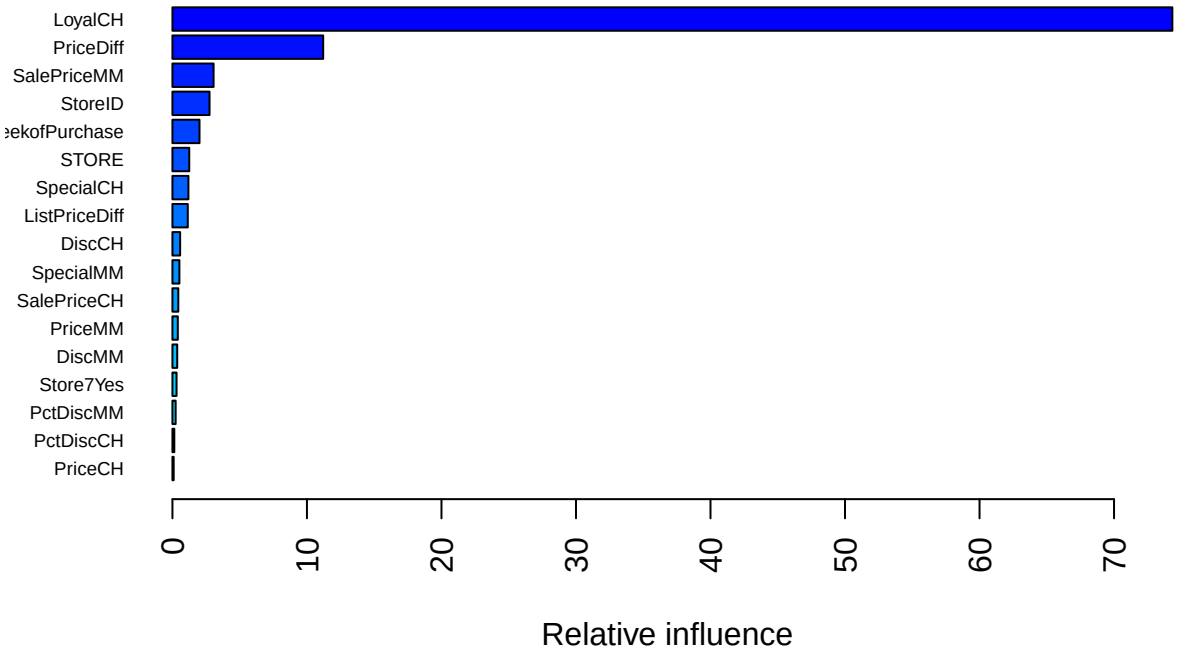


Table 1: Variable importance from boosting model

	var	rel.inf
LoyalCH	LoyalCH	74.336
PriceDiff	PriceDiff	11.207
SalePriceMM	SalePriceMM	3.059
StoreID	StoreID	2.757
WeekofPurchase	WeekofPurchase	2.013
STORE	STORE	1.249
SpecialCH	SpecialCH	1.188
ListPriceDiff	ListPriceDiff	1.139
DiscCH	DiscCH	0.573
SpecialMM	SpecialMM	0.517
SalePriceCH	SalePriceCH	0.435
PriceMM	PriceMM	0.398
DiscMM	DiscMM	0.355
Store7Yes	Store7Yes	0.306
PctDiscMM	PctDiscMM	0.238
PctDiscCH	PctDiscCH	0.138
PriceCH	PriceCH	0.092

From the variable importance plot (the relative influence plot), we can see that `LoyalCH` and `PriceDiff` are the top 2 most important variables.

The test error rate is 0.136.