P8106_hw4_yh3554

Data Science II Homework 4

Yi Huang

2023-04-21

Contents

```r
library(tidyverse)
library(dplyr)
library(knitr)
library(caret)
library(ISLR)
library(mlbench)
library(rpart)
library(rpart.plot)
library(party)
library(partykit)
library(pROC)
library(ranger)
library(gbm)
library(pdp)
library(ggplot2)
library(parallel)
library(doParallel)
```

Problem 1.

In this exercise, we will build tree-based models using the College data (see "College.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%).

The predictors are:

- Apps: Number of applications received
- Accept: Number of applications accepted
- Enroll: Number of new students enrolled
- Top10perc: Pct. new students from top 10% of H.S. class
- Top25perc: Pct. new students from top 25% of H.S. class
- F.Undergrad: Number of fulltime undergraduates
- P.Undergrad: Number of parttime undergraduates
- Room.Board: Room and board costs
- Books: Estimated book costs
- Personal: Estimated personal spending
- PhD: Pct. of faculty with Ph.D.'s
- Terminal: Pct. of faculty with terminal degree
- S.F.Ratio: Student/faculty ratio

- perc.alumni: Pct. alumni who donate
- Expend: Instructional expenditure per student
- Grad.Rate: Graduation rate

Data cleaning

```
# load data
dat <- read.csv("data/College.csv")[,-1]
dat <- na.omit(dat)
head(dat)
```

```
##   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
## 4  417    349    137        60        89         510          63    12960
## 5  193    146     55        16        44         249         869     7560
## 6  587    479    158        38        62         678          41    13500
##   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
## 4       5450   450      875  92       97       7.7          37  19016        59
## 5       4120   800     1500  76       72      11.9           2  10922        15
## 6       3335   500      675  67       73       9.4          11   9727        55
```

```
summary(dat)
```

```
##       Apps           Accept          Enroll         Top10perc
##  Min.   :   81   Min.   :   72   Min.   :  35.0   Min.   : 1.00
##  1st Qu.:  619   1st Qu.:  501   1st Qu.: 206.0   1st Qu.:17.00
##  Median : 1133   Median :  859   Median : 328.0   Median :25.00
##  Mean   : 1978   Mean   : 1306   Mean   : 456.9   Mean   :29.33
##  3rd Qu.: 2186   3rd Qu.: 1580   3rd Qu.: 520.0   3rd Qu.:36.00
##  Max.   :20192   Max.   :13007   Max.   :4615.0   Max.   :96.00
##    Top25perc       F.Undergrad     P.Undergrad       Outstate
##  Min.   :  9.00   Min.   :  139   Min.   :    1   Min.   : 2340
##  1st Qu.: 42.00   1st Qu.:  840   1st Qu.:   63   1st Qu.: 9100
##  Median : 55.00   Median : 1274   Median :  207   Median :11200
##  Mean   : 56.96   Mean   : 1872   Mean   :  434   Mean   :11802
##  3rd Qu.: 70.00   3rd Qu.: 2018   3rd Qu.:  541   3rd Qu.:13970
##  Max.   :100.00   Max.   :27378   Max.   :10221   Max.   :21700
##    Room.Board       Books           Personal        PhD
##  Min.   :2370   Min.   : 250.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3736   1st Qu.: 450.0   1st Qu.: 800   1st Qu.: 60.00
##  Median :4400   Median : 500.0   Median :1100   Median : 73.00
##  Mean   :4586   Mean   : 547.5   Mean   :1214   Mean   : 71.09
##  3rd Qu.:5400   3rd Qu.: 600.0   3rd Qu.:1500   3rd Qu.: 85.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :100.00
##     Terminal        S.F.Ratio      perc.alumni        Expend        Grad.Rate
##  Min.   : 24.00   Min.   : 2.50   Min.   : 2.00   Min.   : 3186   Min.   : 15
```

```
##  1st Qu.: 68.00   1st Qu.:11.10   1st Qu.:16.00   1st Qu.: 7477   1st Qu.: 58
##  Median : 81.00   Median :12.70   Median :25.00   Median : 8954   Median : 69
##  Mean   : 78.53   Mean   :12.95   Mean   :25.89   Mean   :10486   Mean   : 69
##  3rd Qu.: 92.00   3rd Qu.:14.50   3rd Qu.:34.00   3rd Qu.:11625   3rd Qu.: 81
##  Max.   :100.00   Max.   :39.80   Max.   :64.00   Max.   :56233   Max.   :118
```

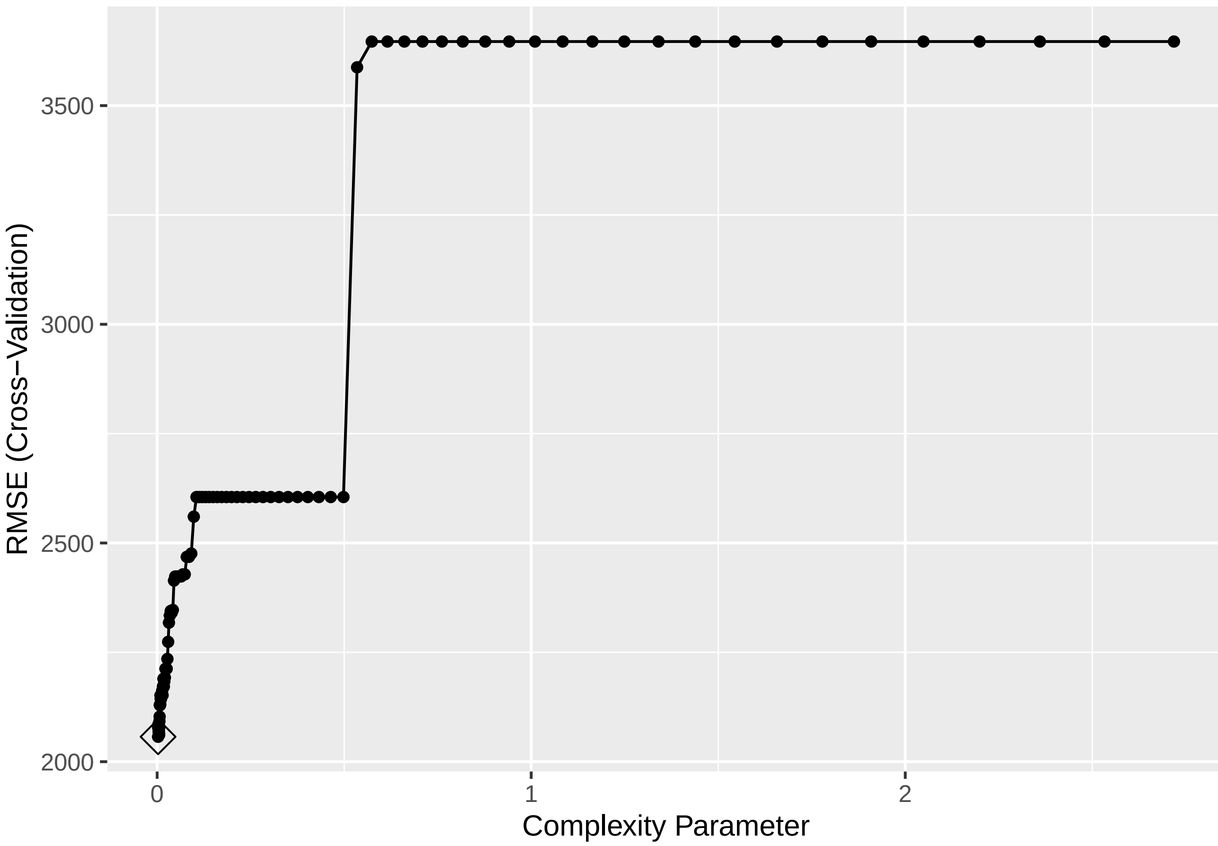
```
set.seed(123)
train_rows <- createDataPartition(y = dat$Outstate,
                                  p = 0.8,
                                  list = FALSE)

# training data
dat_train <- dat[train_rows, ]
x <- dat_train %>% select(-Outstate)
y <- dat_train$Outstate
# test data
dat_test <- dat[-train_rows, ]
x2 <- dat_test %>% select(-Outstate)
y2 <- dat_test$Outstate

set.seed(123)
# resampling method
ctrl <- trainControl(method = "cv")
```

(a)

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

(i) Build a regression tree on train data

```
set.seed(123)
rpart.fit <- train(Outstate ~ . ,
                   dat_train,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,1, length = 100))),
                   trControl = ctrl)

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

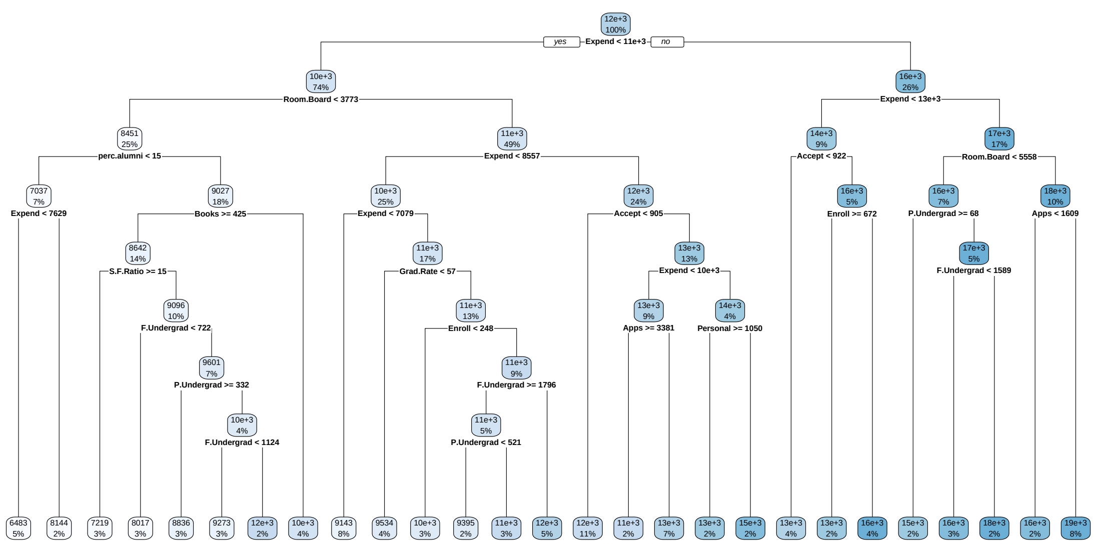



```r
rpart.fit$finalModel$tuneValue[[1]]
```

```
## [1] 0.002478752
```

(ii) create a plot of the tree

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

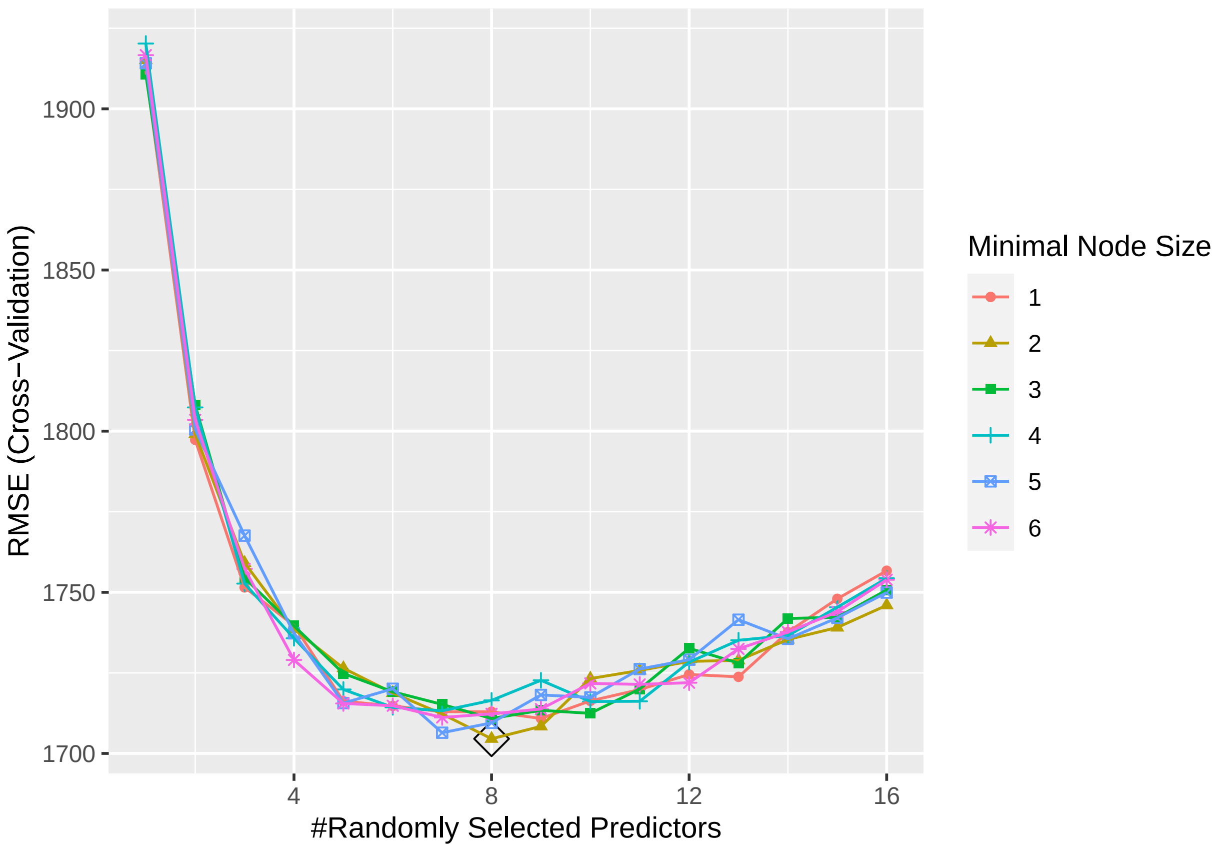


(b)

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

(i) Perform Random forest on train data

```
rf.grid <- expand.grid(mtry = 1:16,
                       splitrule = "variance",
                       min.node.size = 1:6)
set.seed(123)
no_cores <- detectCores() - 1
cl <- makePSOCKcluster(no_cores)
registerDoParallel(cl)
rf.fit <- train(Outstate ~ . ,
                dat_train,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl)
stopCluster(cl)
registerDoSEQ()
ggplot(rf.fit, highlight = TRUE)
```



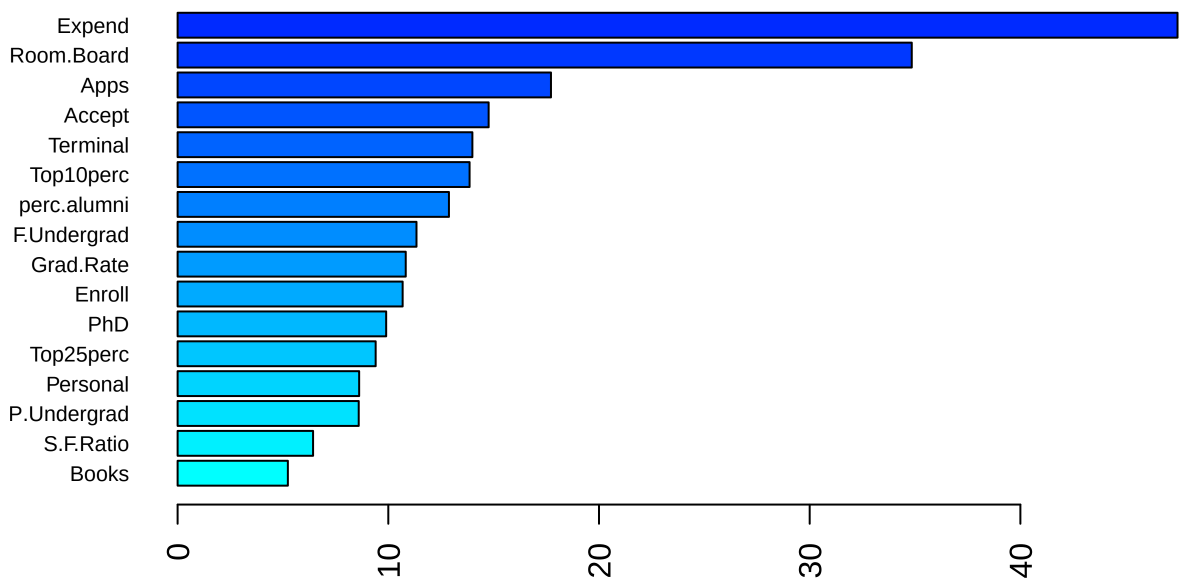

```
rf.fit$bestTune
```

```
##    mtry splitrule min.node.size
## 44    8  variance             2
```

The best tuning parameters are `mtry = 8` with `min.node.size = 2`.

(ii) Report the variable importance and the test error.

```
set.seed(123)
# variable importance
rf.final.per <- ranger(Outstate ~ . ,
                        dat_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(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 = dat_test)
rf.RMSE <- RMSE(rf.predict, y2)
rf.RMSE
```

```
## [1] 1850.927
```

The top 6 most important variables are `Expend`, `Room.Board`, `Apps`, `Accept`, `Terminal`, and `Top10perc`. The RMSE of test set is 1850.93.

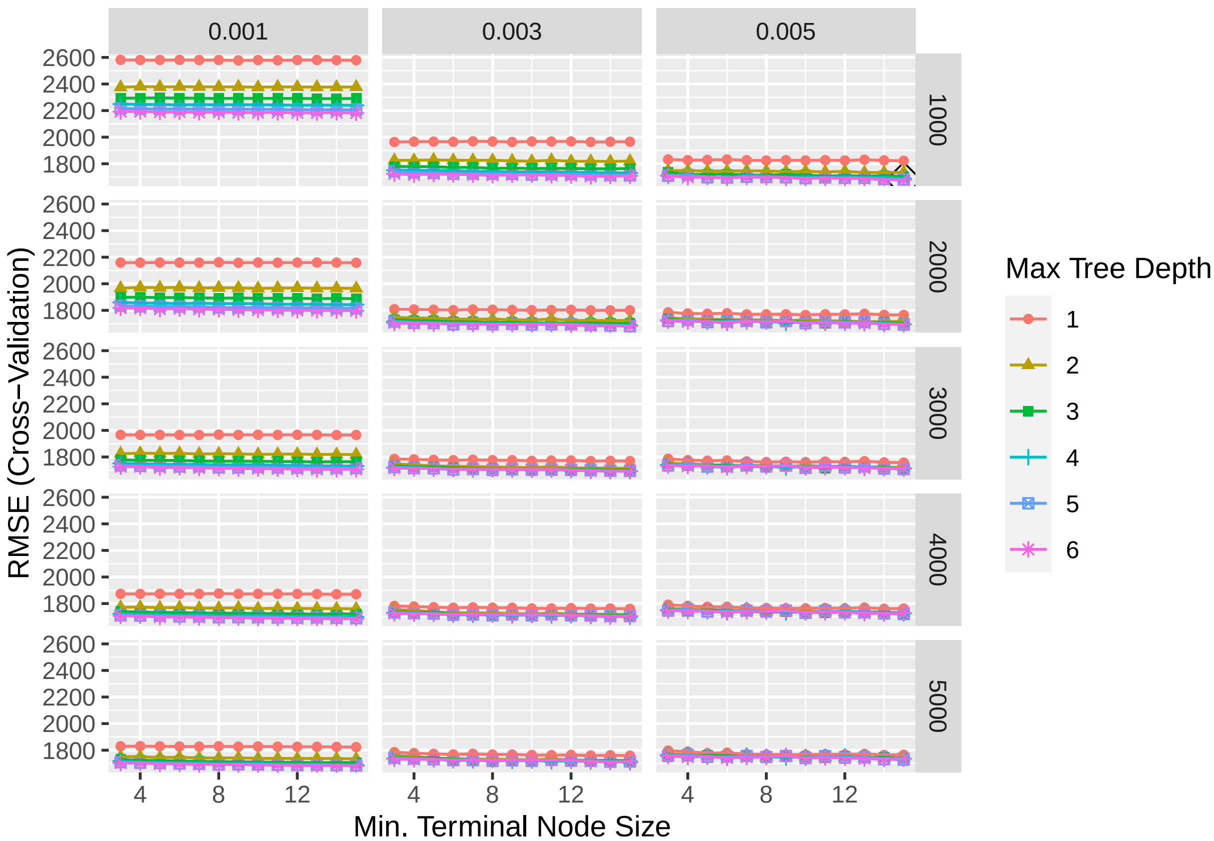
(c)

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

(i) Perform Boosting on the training data

```
gbm.grid <- expand.grid(n.trees = c(1000, 2000, 3000, 4000, 5000),
                        interaction.depth = 1:6,
                        shrinkage = seq(0.001, 0.005, by = 0.002),
                        n.minobsinnode = c(3:15))

set.seed(123)
```

```r
no_cores <- detectCores() - 1
cl <- makePSOCKcluster(no_cores)
registerDoParallel(cl)
gbm.fit <- train(Outstate ~ . ,
                 dat_train,
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 verbose = FALSE)
stopCluster(cl)
registerDoSEQ()
ggplot(gbm.fit, highlight = TRUE)
```




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

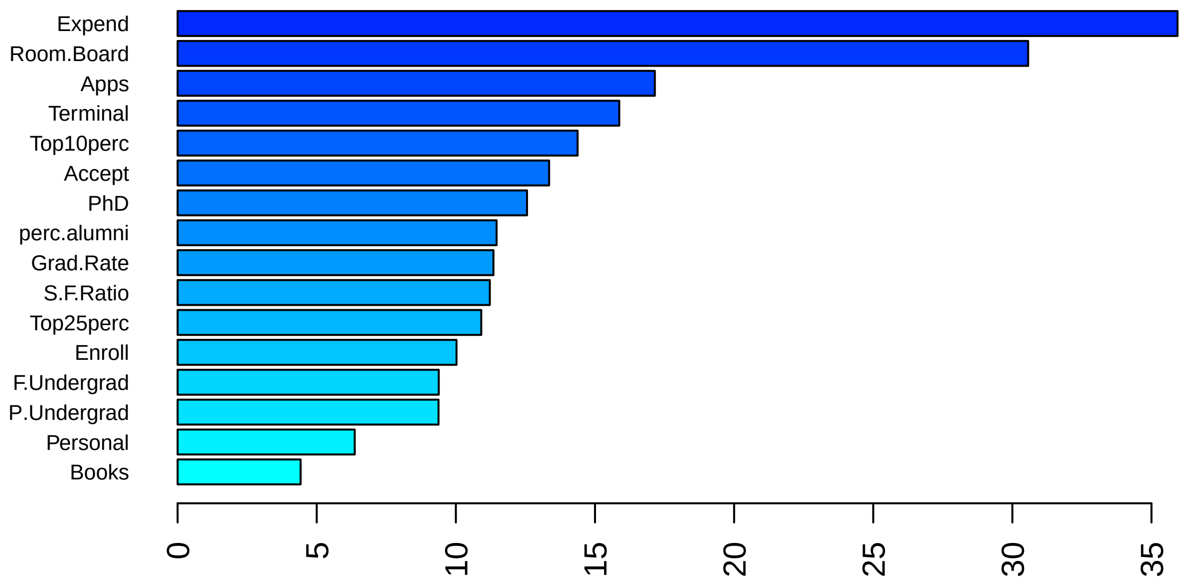
```
##      n.trees interaction.depth shrinkage n.minobsinnode
## 1166    1000                 6     0.005             15
```

The best tuning parameters are `n.trees = 1000`, `interaction.depth = 6`, `shrinkage = 0.005` and `nminobsinode = 15`.

(ii) Report the variable importance and the test error.

```r
set.seed(123)
gbm.final.per <- ranger(Outstate ~ . ,
                        dat_train,
                        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 = dat_test)
gbm.RMSE <- RMSE(gbm.predict, y2)
gbm.RMSE
```

```
## [1] 1733.132
```

The top 6 most important variables are `Expend`, `Room.Board`, `Apps`, `Terminal`, `Top10perc`, and `Accept`. The RMSE of test set is 1733.13.

Problem 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.

Data cleaning

```
data(OJ)
OJ <- na.omit(OJ)
set.seed(123)

train_rows2 <- createDataPartition(y = OJ$Purchase,
                                   p = 0.8,
```

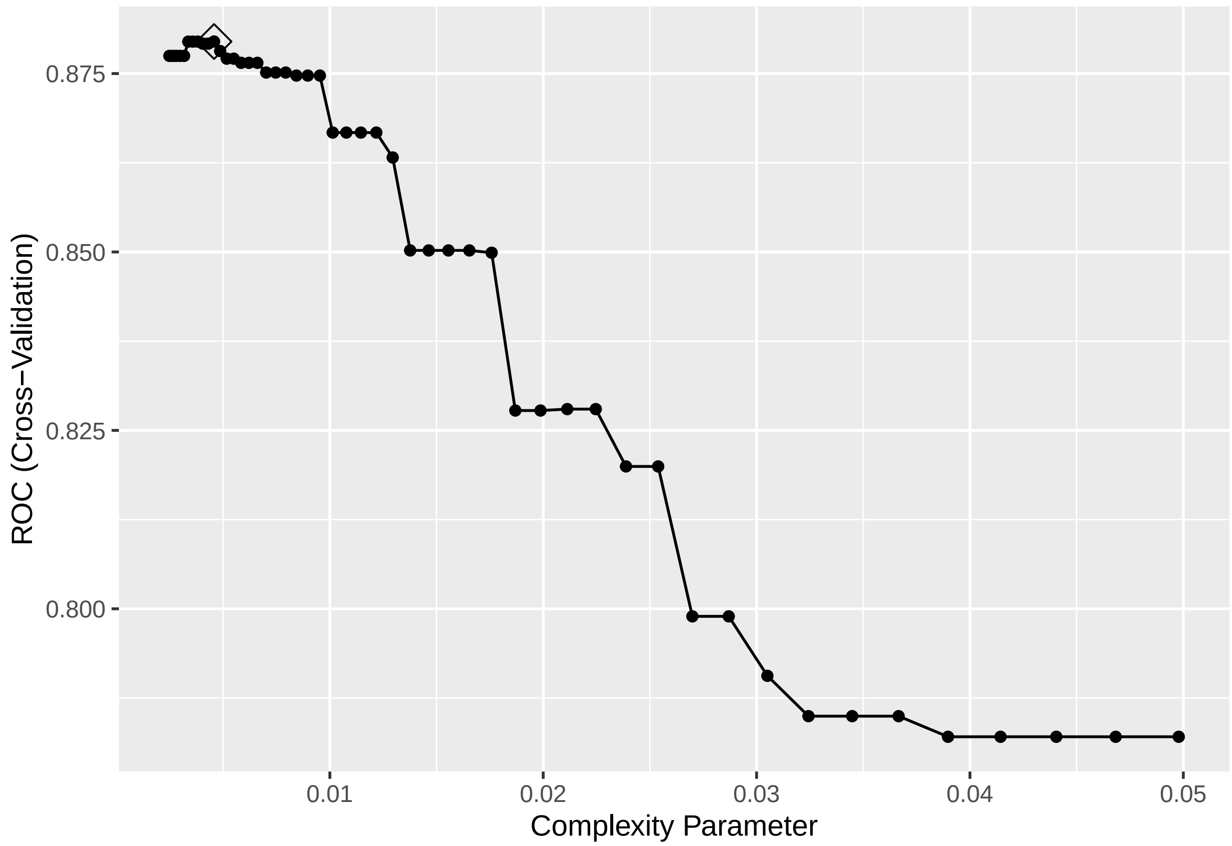
```
                                     list = FALSE)

# training data
OJ_train <- OJ[train_rows2, ]
# test data
OJ_test <- OJ[-train_rows2, ]

# resampling method
ctrl1 <- trainControl(method = "cv",
                      classProbs = TRUE)

set.seed(123)
ctrl2 <- trainControl(method = "cv",
                     classProbs = TRUE,
                     summaryFunction = twoClassSummary)
```

(a)

(i)

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?

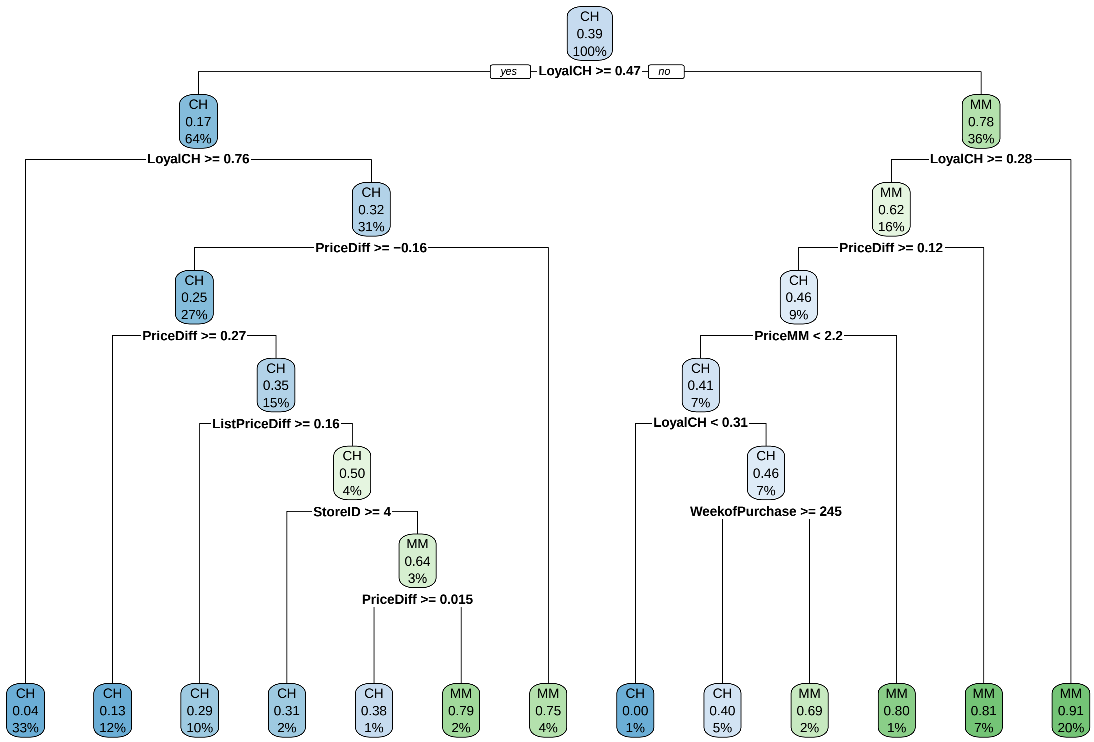
```
set.seed(123)
rpart.fit.OJ <- train(Purchase ~ . ,
                      OJ_train,
                      method = "rpart",
                      tuneGrid = data.frame(cp = exp(seq(-6,-3, len = 50))),
                      trControl = ctrl2,
                      metric = "ROC")
ggplot(rpart.fit.OJ, highlight = TRUE)
```




```
rpart.fit.OJ$bestTune
```

```
##              cp
## 11 0.004572226
```

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



The tree size of 13 has lowest cross-validation error with cp =0.004572.
Note: tree size = number of split + 1

(ii)

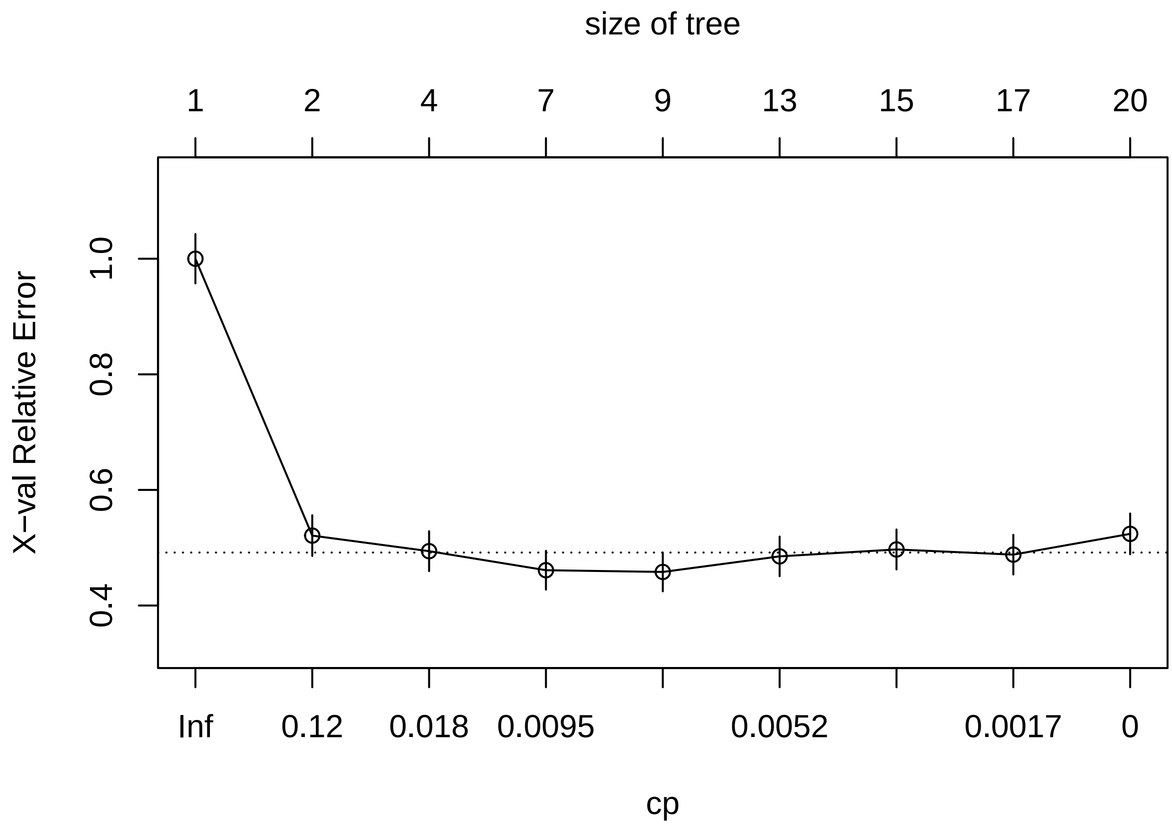
Is this the same as the tree size obtained using the 1 SE rule?

```
set.seed(123)
tree1 <- rpart(formula = Purchase ~ . ,
               OJ_train,
               control = rpart.control(cp = 0))
cpTable <- printcp(tree1)
```

```
##
## Classification tree:
## rpart(formula = Purchase ~ ., data = OJ_train, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] ListPriceDiff  LoyalCH        PriceCH        PriceDiff      PriceMM
## [6] SpecialCH      StoreID        WeekofPurchase
##
## Root node error: 334/857 = 0.38973
##
## n= 857
##
##          CP nsplit rel error  xerror     xstd
## 1 0.517964      0   1.00000 1.00000 0.042745
## 2 0.026946      1   0.48204 0.52096 0.035257
## 3 0.011976      3   0.42814 0.49401 0.034559
## 4 0.007485      6   0.39222 0.46108 0.033651
## 5 0.005988      8   0.37725 0.45808 0.033566
```

```
## 6 0.004491     12   0.35329 0.48503 0.034317
## 7 0.002994     14   0.34431 0.49701 0.034638
## 8 0.000998     16   0.33832 0.48802 0.034398
## 9 0.000000     19   0.33533 0.52395 0.035332
```

```
plotcp(tree1)
```

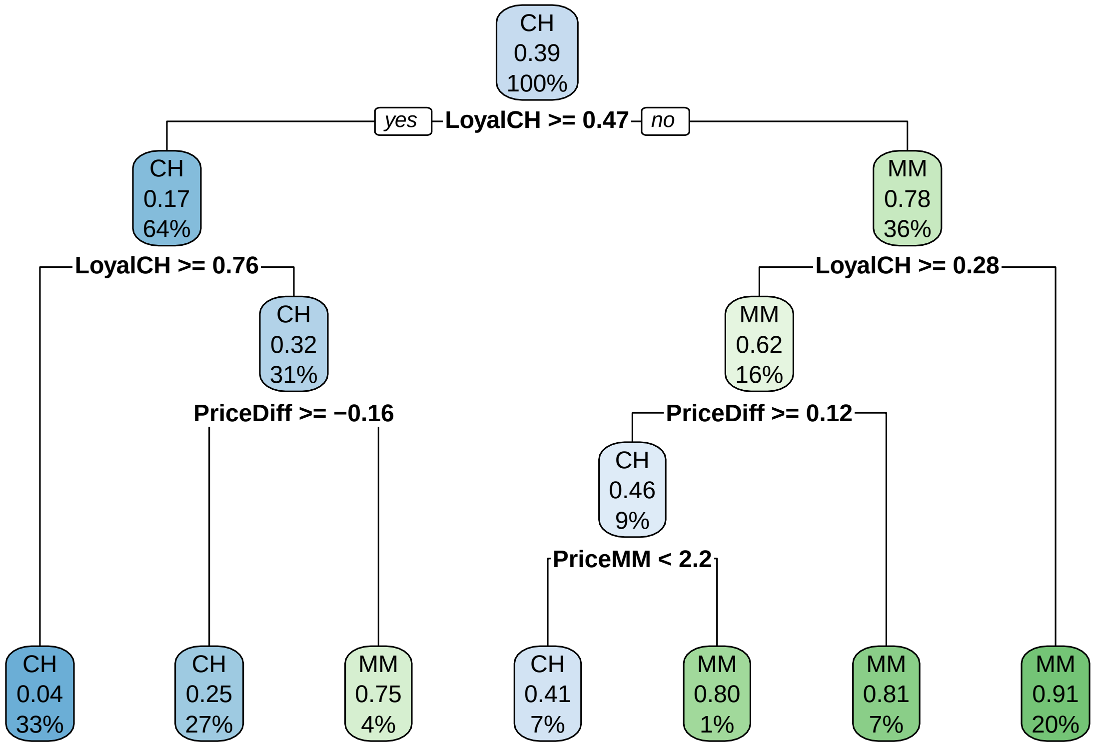



```
# rpart.plot(tree1)

set.seed(123)
# 1SE rule
minErr <- which.min(cpTable[,4])
tree2 <- prune(tree1,cp = cpTable[cpTable[,4]<cpTable[minErr,4]+cpTable[minErr,5],1][1])
cpTable <- printcp(tree2)
```

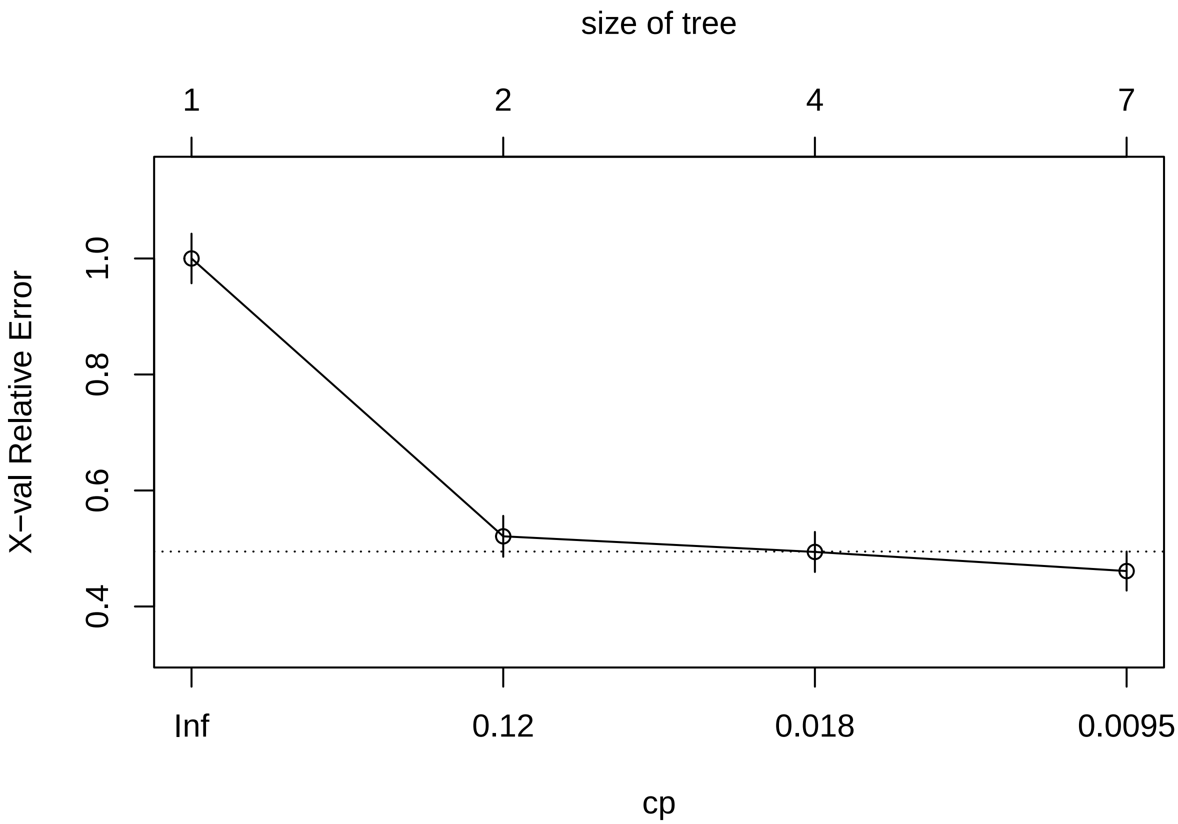
```
##
## Classification tree:
## rpart(formula = Purchase ~ ., data = OJ_train, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] LoyalCH   PriceDiff PriceMM
##
## Root node error: 334/857 = 0.38973
##
## n= 857
##
##         CP nsplit rel error  xerror     xstd
## 1 0.517964      0   1.00000 1.00000 0.042745
## 2 0.026946      1   0.48204 0.52096 0.035257
```

```
## 3 0.011976      3   0.42814 0.49401 0.034559
## 4 0.007485      6   0.39222 0.46108 0.033651
```

```
rpart.plot(tree2)
```




```
plotcp(tree2)
```



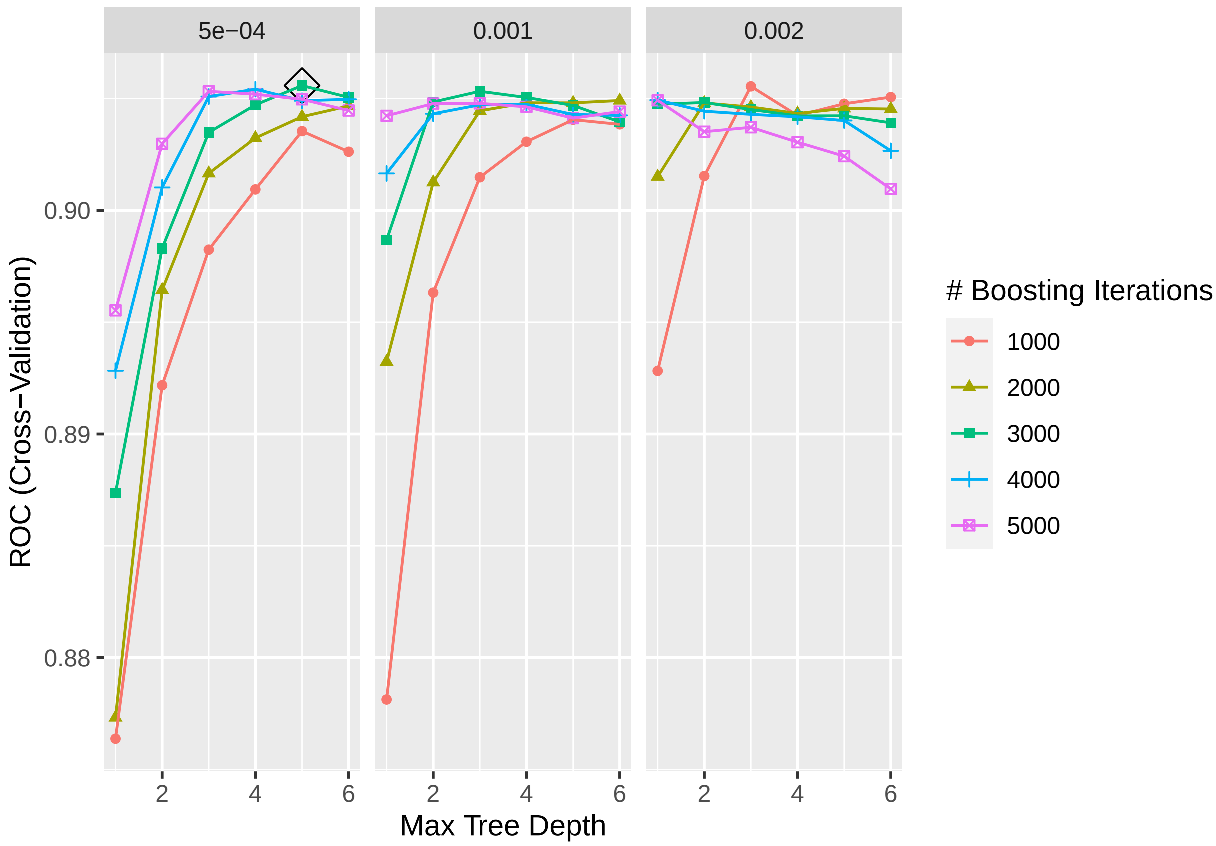
Under 1 SE rule, the tree size with lowest cross-validation error is 7. The tree size obtained by using cross validation is different from the tree size obtained by using 1 SE rule.

(b)

(i)

Perform boosting on the training data and report the variable importance.

```
gbmA.grid <- expand.grid(n.trees = c(1000,2000,3000,4000,5000),
                         interaction.depth = 1:6,
                         shrinkage = c(0.0005,0.001,0.002),
                         n.minobsinnode = 1)

set.seed(123)
no_cores <- detectCores() - 1
cl <- makePSOCKcluster(no_cores)
registerDoParallel(cl)
gbmA.fit <- train(Purchase ~ . ,
                  OJ_train,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl2,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)
stopCluster(cl)
registerDoSEQ()
ggplot(gbmA.fit, highlight = TRUE)
```




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

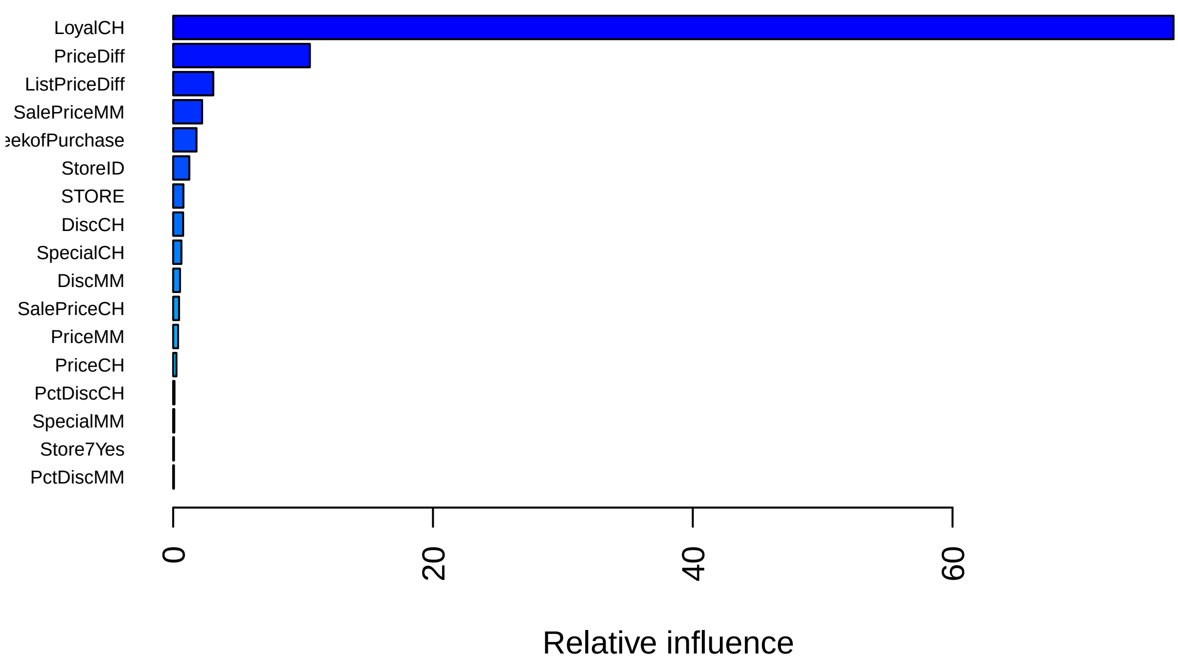


Table 1: Variable importance from boosting model

	var	rel.inf
LoyalCH	LoyalCH	77.013
PriceDiff	PriceDiff	10.524
ListPriceDiff	ListPriceDiff	3.089
SalePriceMM	SalePriceMM	2.229
WeekofPurchase	WeekofPurchase	1.794
StoreID	StoreID	1.242
STORE	STORE	0.789
DiscCH	DiscCH	0.769
SpecialCH	SpecialCH	0.634
DiscMM	DiscMM	0.525
SalePriceCH	SalePriceCH	0.452
PriceMM	PriceMM	0.378
PriceCH	PriceCH	0.253
PctDiscCH	PctDiscCH	0.101
SpecialMM	SpecialMM	0.086
Store7Yes	Store7Yes	0.061
PctDiscMM	PctDiscMM	0.059

In the boosting model, the top 2 most important variables are `LoyalCH` and `PriceDiff`.

(ii)

What is the test error rate?

```
gbmA.pred <- predict(gbmA.fit, newdata = OJ_test, type = "raw")
error.rate.gbmA <- mean(gbmA.pred != OJ$Purchase[-train_rows2])
error.rate.gbmA
```

```
## [1] 0.1971831
```

The test error rate is 0.197.

Additional analysis: comparing classfication tree and boostrap

Report cross-validation results on train data

```
set.seed(123)
resamp <- resamples(list( ctrees = rpart.fit.OJ,
                         gbmA = gbmA.fit))

summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: ctrees, gbmA
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## ctrees 0.7777149 0.8547942 0.8879662 0.8794884 0.9177400 0.9268648    0
```

```
## gbmA   0.8546380 0.8899477 0.9073427 0.9055784 0.9278846 0.9519726    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## ctrees 0.7884615 0.8537736 0.8762700 0.8642598 0.8846154 0.9038462    0
## gbmA   0.8269231 0.8750000 0.9143687 0.8966255 0.9230769 0.9245283    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## ctrees 0.6470588 0.6519608 0.7121212 0.7220143 0.7803030 0.8484848    0
## gbmA   0.6176471 0.6742424 0.7205882 0.7397504 0.8181818 0.8484848    0
```

Based on the cross-validation results on train data, bootstrap has a higher mean ROC value, implies bootstrap method performs better than classification tree.