P8106_HW4_yz4184

Yunlin Zhou

Contents

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

Problem 1

```r
# Import and clean the data
college_df = read.csv("./College.csv")%>%
  janitor::clean_names()%>%
  drop_na()%>%
  relocate("outstate")%>%
  select(-college)

# Partition data into training/test sets
set.seed(1)
college_train = createDataPartition(y = college_df$outstate,
                                   p =0.8,
                                   list = FALSE)
train_df = college_df[college_train,]
test_df = college_df[-college_train,]
```

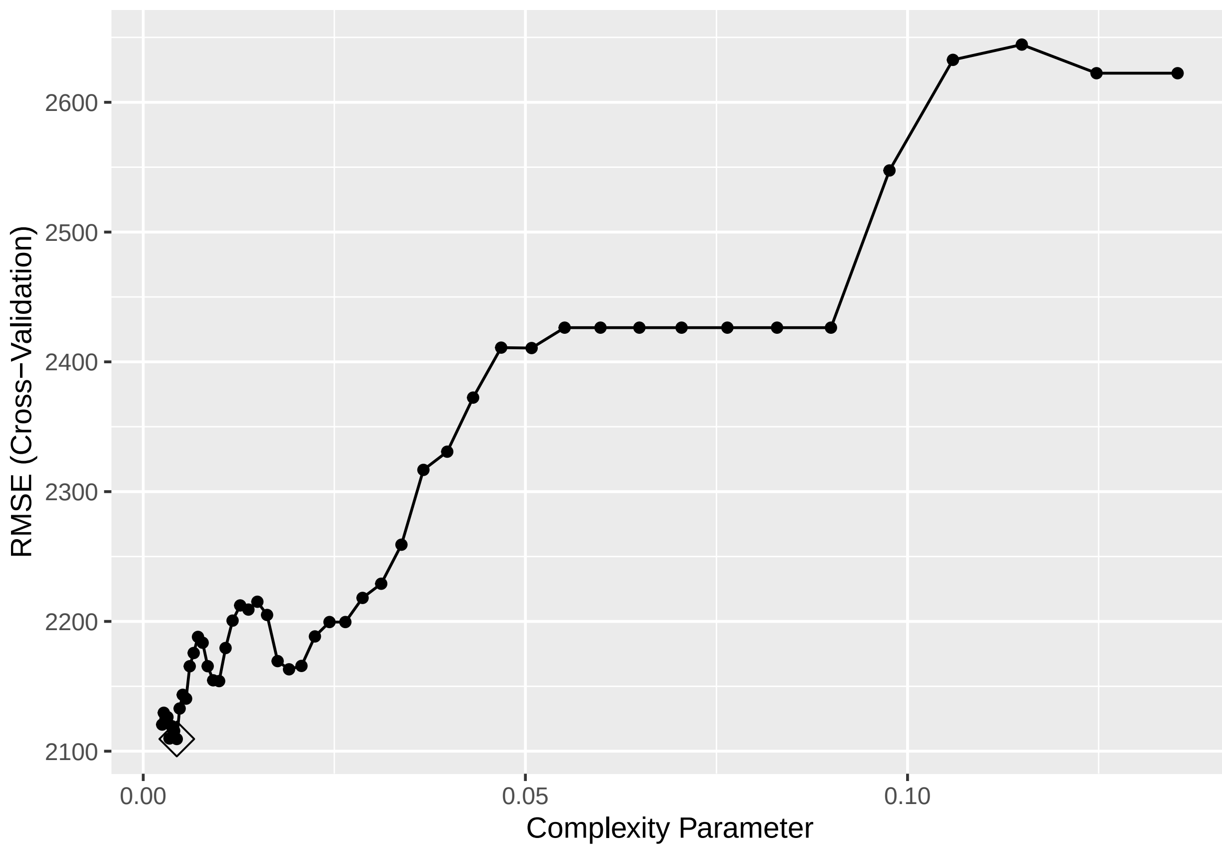
Part a

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

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

set.seed(1)
tree1 <- train(outstate ~ .,
                  train_df,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 50))),
                   trControl = ctrl)
tree1$bestTune
```

```
##            cp
## 8 0.004389362
```

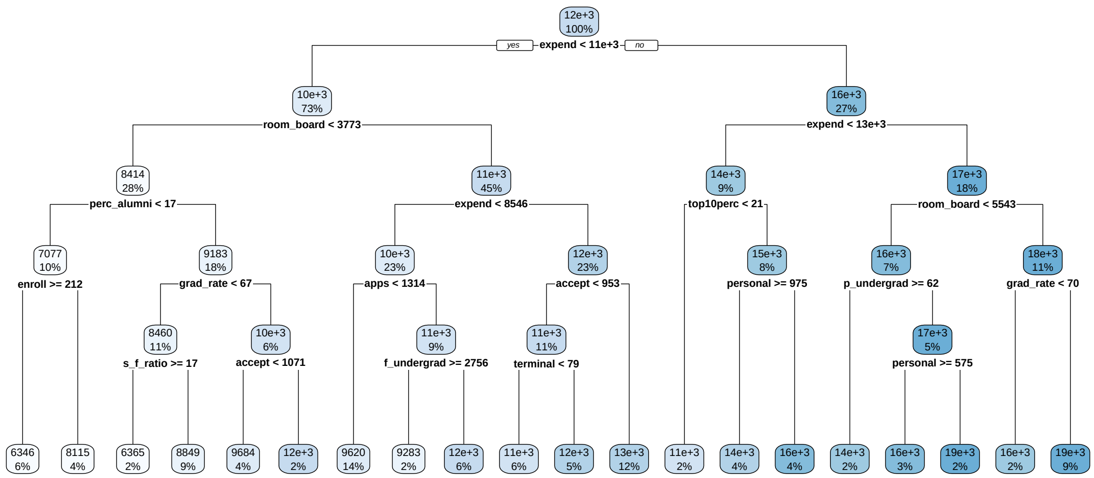
```
ggplot(tree1, highlight = TRUE)
```



In the pruned tree regression model, the tune parameter cp is 0.00438936184277844.

Create a plot of the tree.

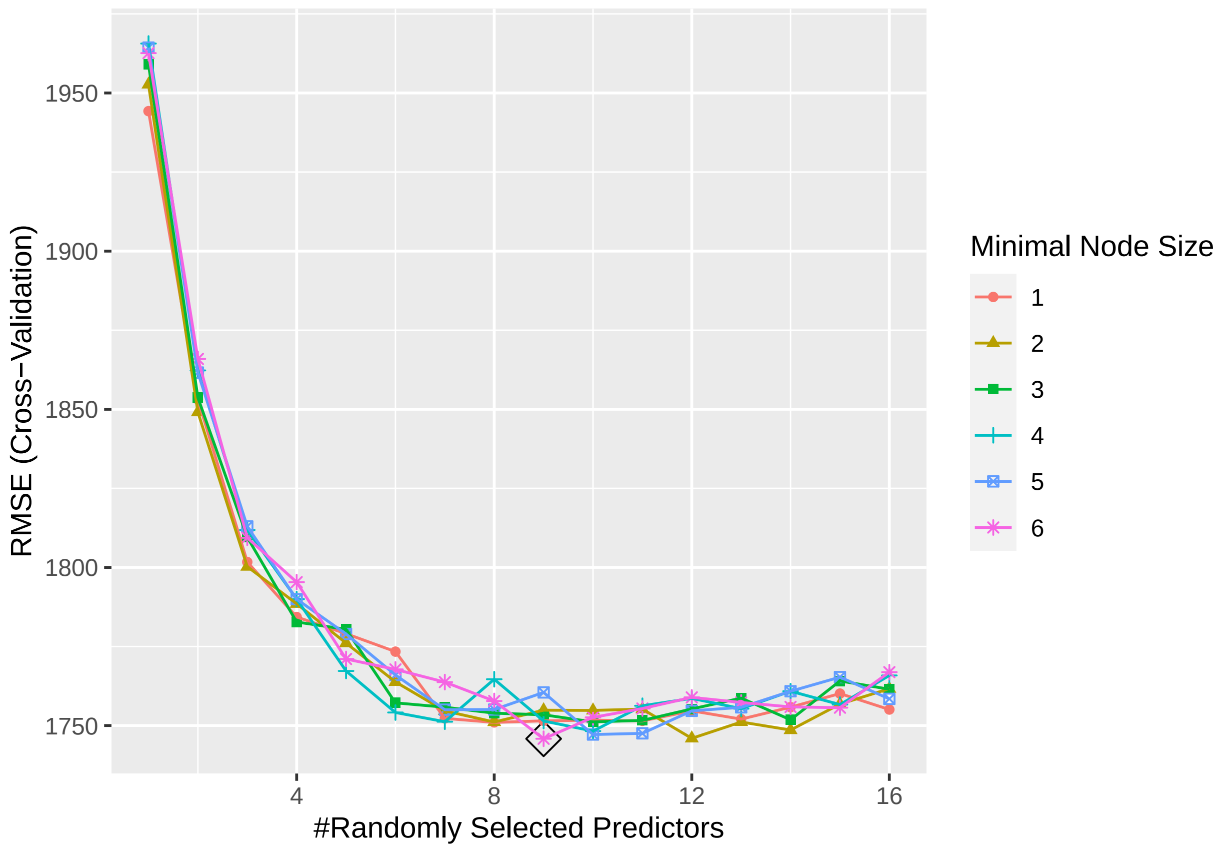
```
rpart.plot(tree1$finalModel)
```


12e+3
100%
yes
expend < 11e+3
no
10e+3
73%
room_board < 3773
16e+3
27%
expend < 13e+3
8414
28%
perc_alumni < 17
11e+3
45%
expend < 8546
14e+3
9%
top10perc < 21
17e+3
18%
room_board < 5543
7077
10%
enroll >= 212
9183
18%
grad_rate < 67
10e+3
23%
apps < 1314
12e+3
23%
accept < 953
15e+3
8%
personal >= 975
16e+3
7%
p_undergrad >= 62
18e+3
11%
grad_rate < 70
8460
11%
s_f_ratio >= 17
10e+3
6%
accept < 1071
11e+3
9%
f_undergrad >= 2756
11e+3
11%
terminal < 79
17e+3
5%
personal >= 575
6346
6%
8115
4%
6365
2%
8849
9%
9684
4%
12e+3
2%
9620
14%
9283
2%
12e+3
6%
11e+3
6%
12e+3
5%
13e+3
12%
11e+3
2%
14e+3
4%
16e+3
4%
14e+3
2%
16e+3
3%
19e+3
2%
16e+3
2%
19e+3
9%

Part b

Perform random forest on the training data.

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

rf1 <- train(outstate ~ .,
             train_df,
             method = "ranger",
             tuneGrid = rf.grid,
             trControl = ctrl)

ggplot(rf1, highlight = TRUE)
```



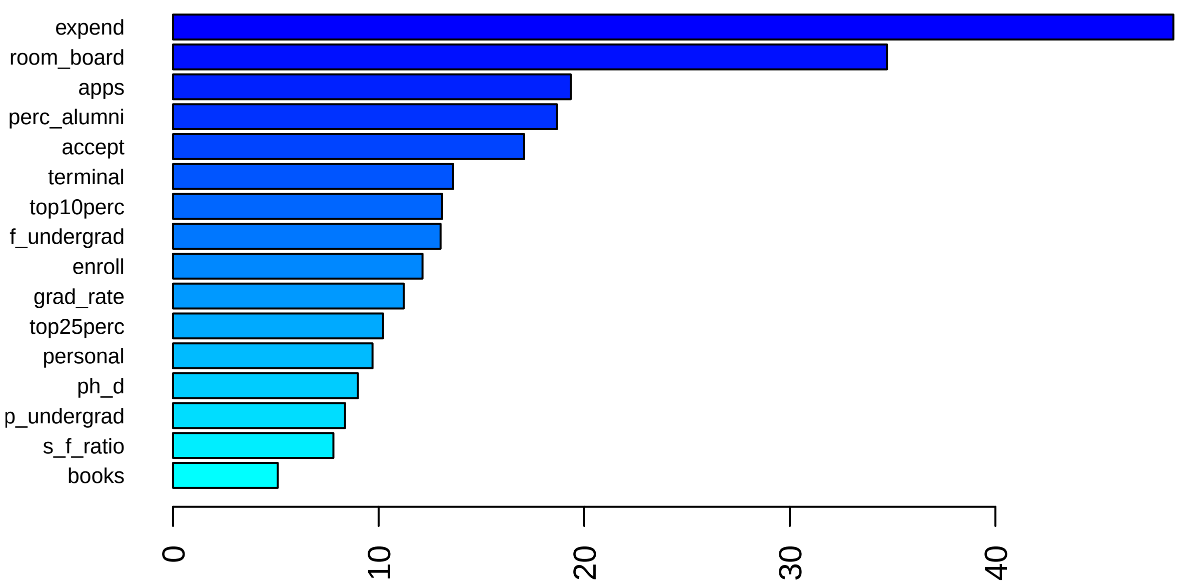

```
rf1$bestTune
```

```
##    mtry splitrule min.node.size
## 54    9  variance             6
```

In this random forest model, the best model is with minimum node size 6 and 9 selected predictors.

Report the variable importance.

```
set.seed(1)
rf1.final.per <- ranger(outstate ~ . ,
                        train_df,
                        mtry = rf1$bestTune[[1]],
                        splitrule = "variance",
                        min.node.size = rf1$bestTune[[3]],
                        importance = "permutation",
                        scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf1.final.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(16))
```



Using the permutation method, the most important predictors are expend and room_board.

Report the test error.

```
pred.rf <- predict(rf1, newdata = test_df)
te_rf = RMSE(pred.rf, test_df$outstate)
te_rf
```

```
## [1] 1651.307
```

The test error is 1651.3069135.

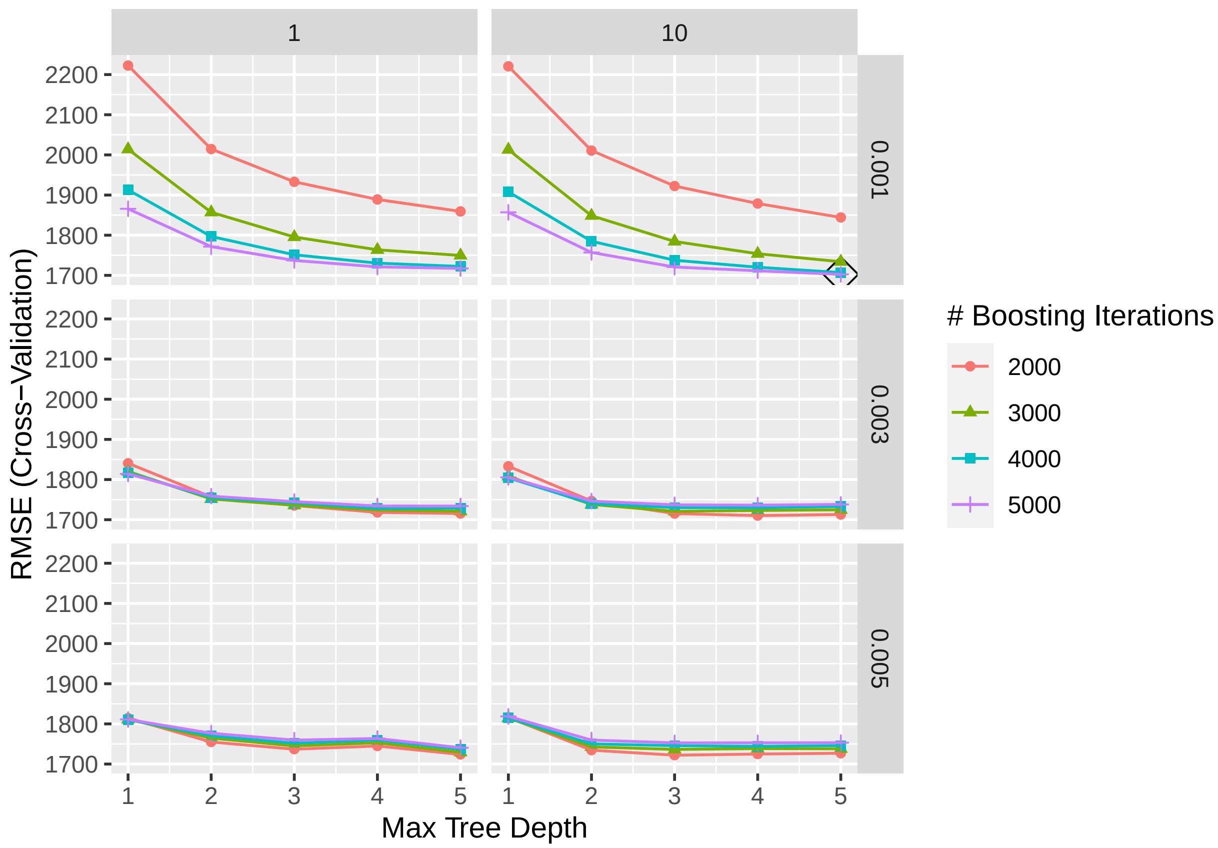
Part c

Perform boosting on the training data.

```
gbm.grid <- expand.grid(n.trees = c(2000,3000,4000,5000),
                        interaction.depth = 1:5,
                        shrinkage = c(0.001,0.003,0.005),
                        n.minobsinnode = c(1,10))
set.seed(1)
gbm1 <- train(outstate ~ . ,
              train_df,
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 verbose = FALSE)

gbm1$bestTune
```

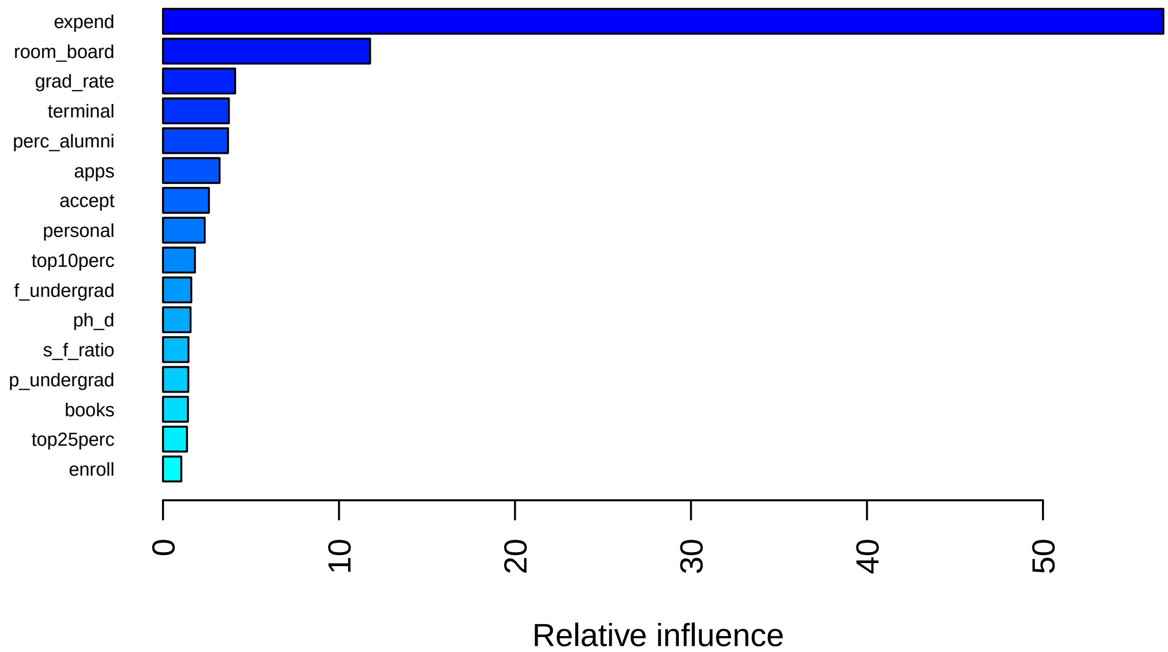
```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 40    5000                 5     0.001             10
```

```
ggplot(gbm1, highlight = TRUE)
```



Report the variable importance.

```
summary(gbm1$finalModel, las = 2, cBars = 16, cex.names = 0.6)
```




```
##                     var    rel.inf
## expend           expend 56.839618
## room_board   room_board 11.763945
## grad_rate     grad_rate  4.097538
## terminal       terminal  3.740792
## perc_alumni perc_alumni  3.692448
## apps               apps  3.212812
## accept           accept  2.606760
## personal       personal  2.368107
## top10perc     top10perc  1.812568
## f_undergrad f_undergrad  1.605638
## ph_d               ph_d  1.562622
## s_f_ratio     s_f_ratio  1.445236
## p_undergrad p_undergrad  1.436907
## books             books  1.415833
## top25perc     top25perc  1.361722
## enroll           enroll  1.037453
```

The most important predictors are expend and room_board.

Report the test error.

```
pred.gbm <- predict(gbm1, newdata = test_df)
te_gbm = RMSE(pred.gbm, test_df$outstate)
te_gbm
```

```
## [1] 1620.551
```

The test error is 1620.5511732.

Queation 2

```
# Import and clean the data
data(OJ)
oj_df=
  OJ %>%
  na.omit() %>%
  janitor::clean_names()

# Partition data into training/test sets
set.seed(2)
oj_train = createDataPartition(y = oj_df$purchase,
                               p = 700/1070,
                                   list = FALSE)
train_oj = oj_df[oj_train,]
test_oj = oj_df[-oj_train,]
```

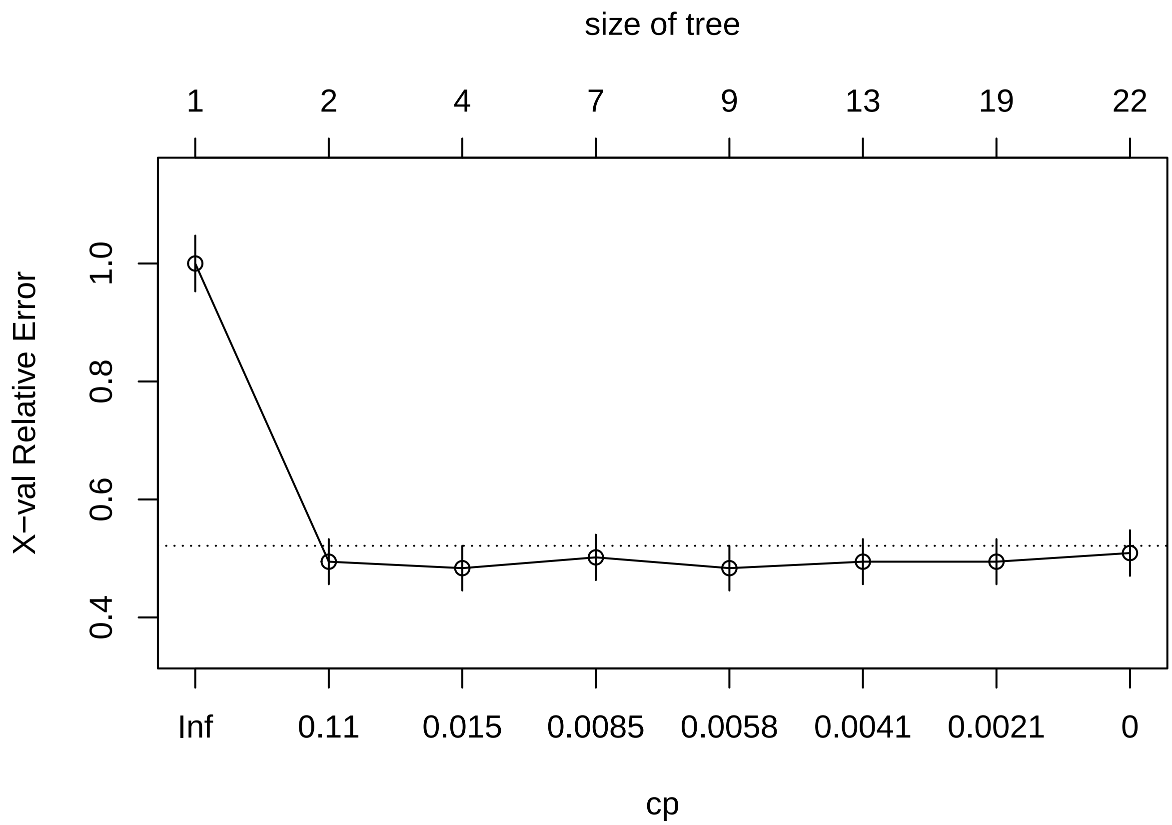
Part a

Build a classification tree using the training data

```
set.seed(2)
tree2 <- rpart(formula = purchase ~ . ,
               data =  train_oj,
               control = rpart.control(cp = 0))

cpTable <- printcp(tree2)
```

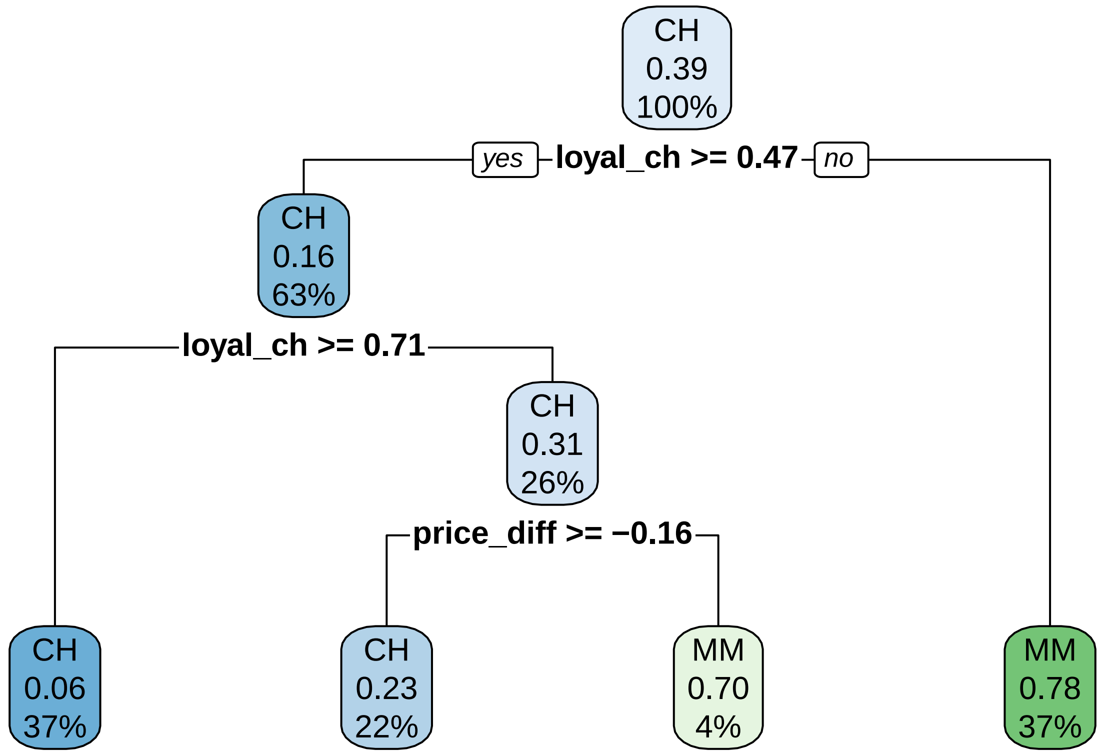
```
##
## Classification tree:
## rpart(formula = purchase ~ ., data = train_oj, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] list_price_diff loyal_ch        price_diff      sale_price_ch
## [5] sale_price_mm   store           store_id        weekof_purchase
##
## Root node error: 273/701 = 0.38944
##
## n= 701
##
##          CP nsplit rel error  xerror     xstd
## 1 0.5201465      0   1.00000 1.00000 0.047291
## 2 0.0219780      1   0.47985 0.49451 0.038243
## 3 0.0097680      3   0.43590 0.48352 0.037916
## 4 0.0073260      6   0.40659 0.50183 0.038457
## 5 0.0045788      8   0.39194 0.48352 0.037916
## 6 0.0036630     12   0.37363 0.49451 0.038243
## 7 0.0012210     18   0.34799 0.49451 0.038243
## 8 0.0000000     21   0.34432 0.50916 0.038668
```

```
plotcp(tree2)
```



Use cross-validation to determine the tree size and create a plot of the final tree.

```
# minimum cross-validation error
minErr <- which.min(cpTable[,4])
tree3 <- prune(tree2, cp = cpTable[minErr,1])
rpart.plot(tree3)
```




```
summary(tree3)
```

```
## Call:
## rpart(formula = purchase ~ ., data = train_oj, control = rpart.control(cp = 0))
##   n= 701
##
##           CP nsplit rel error    xerror       xstd
## 1 0.52014652      0 1.0000000 1.0000000 0.04729133
## 2 0.02197802      1 0.4798535 0.4945055 0.03824313
## 3 0.00976801      3 0.4358974 0.4835165 0.03791591
##
## Variable importance
##        loyal_ch      price_diff   sale_price_mm     pct_disc_mm         disc_mm
##              77               7               4               3               3
## weekof_purchase        price_ch      special_mm        price_mm        store_id
##               2               1               1               1               1
##
## Node number 1: 701 observations,    complexity param=0.5201465
##   predicted class=CH  expected loss=0.3894437  P(node) =1
##     class counts:   428   273
##    probabilities: 0.611 0.389
##   left son=2 (443 obs) right son=3 (258 obs)
##   Primary splits:
##       loyal_ch   < 0.469289 to the right, improve=121.50000, (0 missing)
##       store_id   < 3.5      to the right, improve= 32.47240, (0 missing)
```

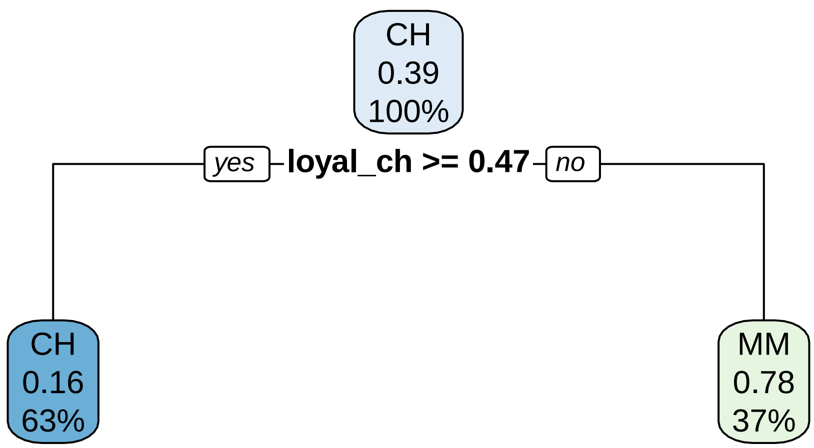
```
##       price_diff < 0.015    to the right, improve= 21.38773, (0 missing)
##       store7     splits as  RL,           improve= 19.81980, (0 missing)
##       store      < 0.5      to the left,  improve= 19.81980, (0 missing)
##   Surrogate splits:
##       disc_mm       < 0.57     to the left,  agree=0.641, adj=0.023, (0 split)
##       pct_disc_mm   < 0.264375 to the left,  agree=0.641, adj=0.023, (0 split)
##       sale_price_mm < 1.385    to the right, agree=0.638, adj=0.016, (0 split)
##       price_diff    < -0.575   to the right, agree=0.638, adj=0.016, (0 split)
##       sale_price_ch < 2.025    to the left,  agree=0.633, adj=0.004, (0 split)
##
## Node number 2: 443 observations,    complexity param=0.02197802
##   predicted class=CH  expected loss=0.1647856  P(node) =0.6319544
##     class counts:   370    73
##    probabilities: 0.835 0.165
##   left son=4 (259 obs) right son=5 (184 obs)
##   Primary splits:
##       loyal_ch        < 0.705699 to the right, improve=13.233360, (0 missing)
##       price_diff      < -0.39    to the right, improve=11.999940, (0 missing)
##       sale_price_mm   < 2.04     to the right, improve= 7.131761, (0 missing)
##       special_mm      < 0.5      to the left,  improve= 5.427963, (0 missing)
##       list_price_diff < 0.235    to the right, improve= 5.218477, (0 missing)
##   Surrogate splits:
##       price_ch        < 1.775    to the right, agree=0.639, adj=0.130, (0 split)
##       weekof_purchase < 237.5    to the right, agree=0.634, adj=0.120, (0 split)
##       price_mm        < 2.04     to the right, agree=0.630, adj=0.109, (0 split)
##       store_id        < 2.5      to the right, agree=0.614, adj=0.071, (0 split)
##       sale_price_mm   < 2.04     to the right, agree=0.605, adj=0.049, (0 split)
##
## Node number 3: 258 observations
##   predicted class=MM  expected loss=0.2248062  P(node) =0.3680456
##     class counts:    58   200
##    probabilities: 0.225 0.775
##
## Node number 4: 259 observations
##   predicted class=CH  expected loss=0.06177606  P(node) =0.3694722
##     class counts:   243    16
##    probabilities: 0.938 0.062
##
## Node number 5: 184 observations,    complexity param=0.02197802
##   predicted class=CH  expected loss=0.3097826  P(node) =0.2624822
##     class counts:   127    57
##    probabilities: 0.690 0.310
##   left son=10 (154 obs) right son=11 (30 obs)
##   Primary splits:
##       price_diff      < -0.165   to the right, improve=10.915950, (0 missing)
##       list_price_diff < 0.235    to the right, improve= 8.137578, (0 missing)
##       store_id        < 5.5      to the right, improve= 5.270274, (0 missing)
##       store7          splits as  RL,           improve= 5.270274, (0 missing)
##       store           < 0.5      to the left,  improve= 5.270274, (0 missing)
##   Surrogate splits:
##       sale_price_mm   < 1.585    to the right, agree=0.891, adj=0.333, (0 split)
##       pct_disc_mm     < 0.187437 to the left,  agree=0.886, adj=0.300, (0 split)
##       disc_mm         < 0.57     to the left,  agree=0.880, adj=0.267, (0 split)
##       weekof_purchase < 274.5    to the left,  agree=0.875, adj=0.233, (0 split)
```

```
##       special_mm      < 0.5      to the left,  agree=0.859, adj=0.133, (0 split)
##
## Node number 10: 154 observations
##   predicted class=CH  expected loss=0.2337662  P(node) =0.2196862
##     class counts:   118    36
##    probabilities: 0.766 0.234
##
## Node number 11: 30 observations
##   predicted class=MM  expected loss=0.3  P(node) =0.04279601
##     class counts:     9    21
##    probabilities: 0.300 0.700
```

When the tree size is 3 corresponds to the lowest cross-validation error.

Using the 1 SE rule

```r
tree4 <- prune(tree2, cp = cpTable[cpTable[,4]<cpTable[minErr,4]+cpTable[minErr,5],1][1])
rpart.plot(tree4)
```

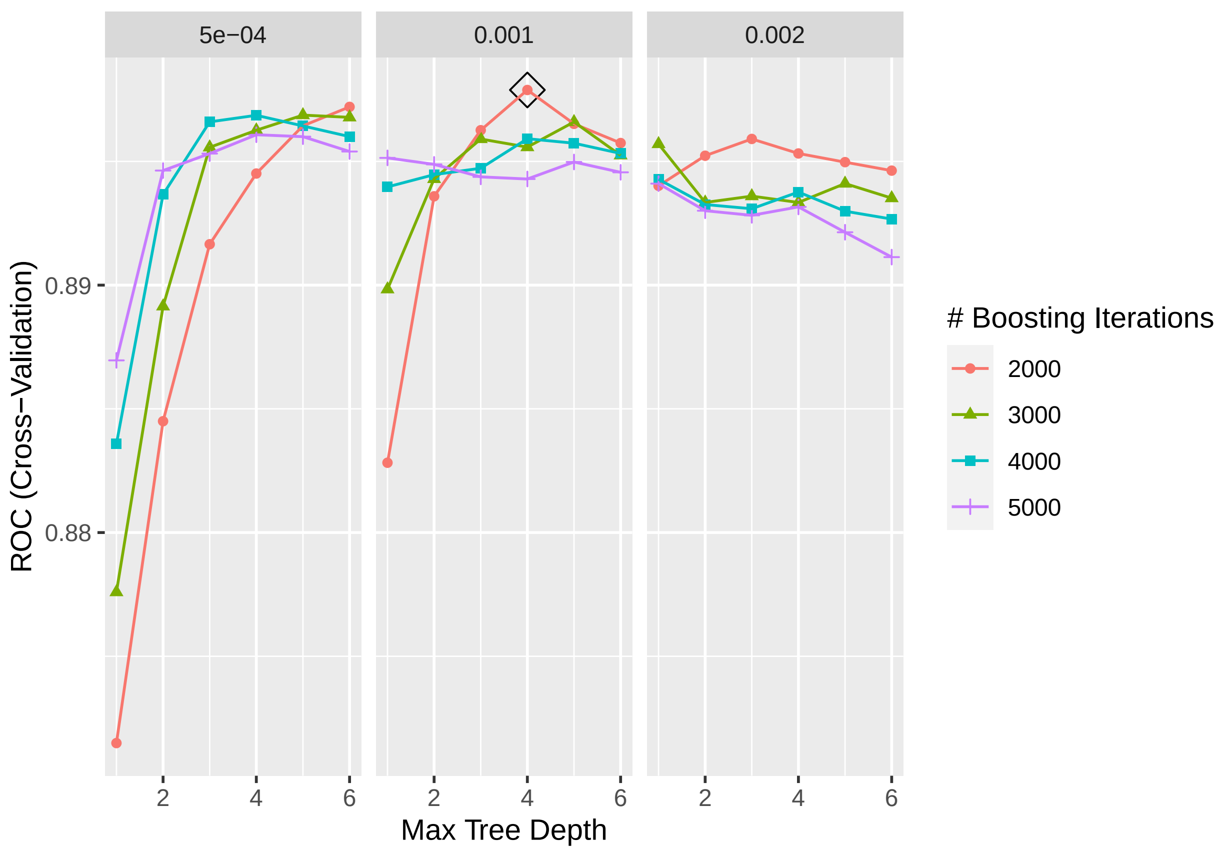


The tree is smaller when the tree size obtained using the 1 SE rule.

Part b

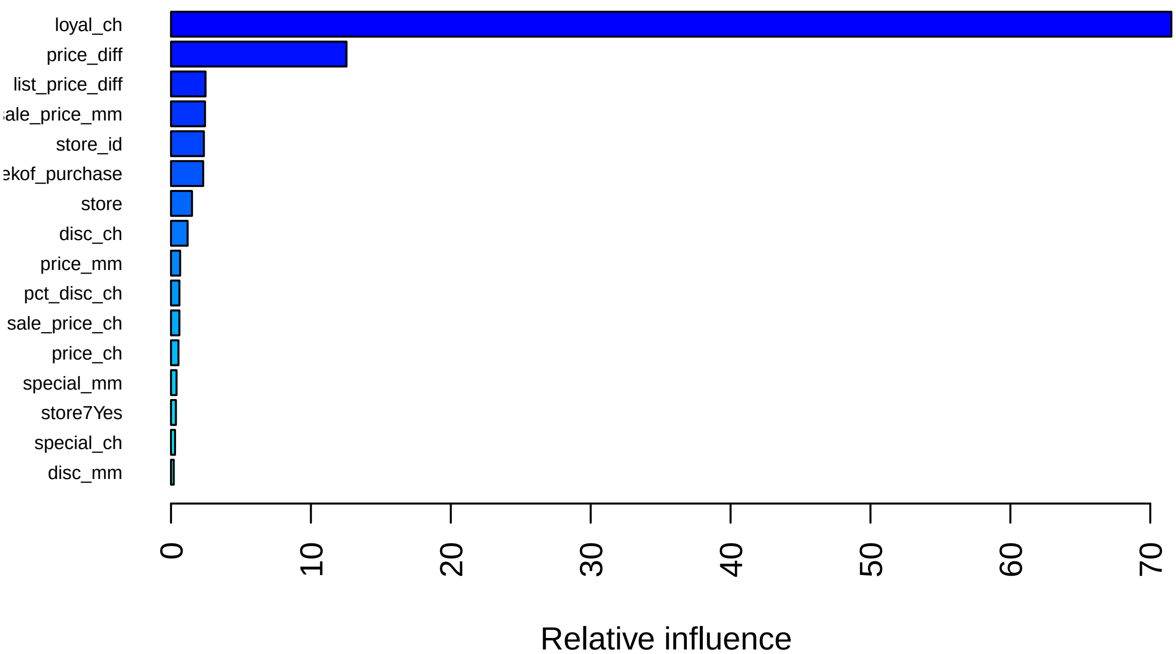
Perform boosting on the training data.

```
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(2)
gbmA.fit <- train(purchase ~ . ,
                  data = train_oj,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl2,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)

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



Report the variable importance.

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




```
##                             var    rel.inf
## loyal_ch               loyal_ch 71.5077135
## price_diff           price_diff 12.5439501
## list_price_diff list_price_diff  2.4675725
## sale_price_mm     sale_price_mm  2.4288086
## store_id               store_id  2.3498719
## weekof_purchase weekof_purchase  2.2994427
## store                     store  1.4990614
## disc_ch                 disc_ch  1.1872920
## price_mm               price_mm  0.6534050
## pct_disc_ch         pct_disc_ch  0.6014305
## sale_price_ch     sale_price_ch  0.5990112
## price_ch               price_ch  0.5322951
## special_mm           special_mm  0.3947971
## store7Yes             store7Yes  0.3494613
## special_ch           special_ch  0.2865049
## disc_mm                 disc_mm  0.1970333
## pct_disc_mm         pct_disc_mm  0.1023489
```

The most important variables are loyal_ch and price_diff.

Test error

```
gbmA.pred <- predict(gbmA.fit, newdata = test_oj)
test.error = mean(gbmA.pred != test_oj$purchase)
test_error_rate = test.error*100
test_error_rate
```

```
## [1] 16.53117
```

The test error rate is16.5311653%.