

DS2__HW4__REGRESSION__TREE

Siyan Chen

4/21/2019

```
data(Prostate)
head(Prostate)
```

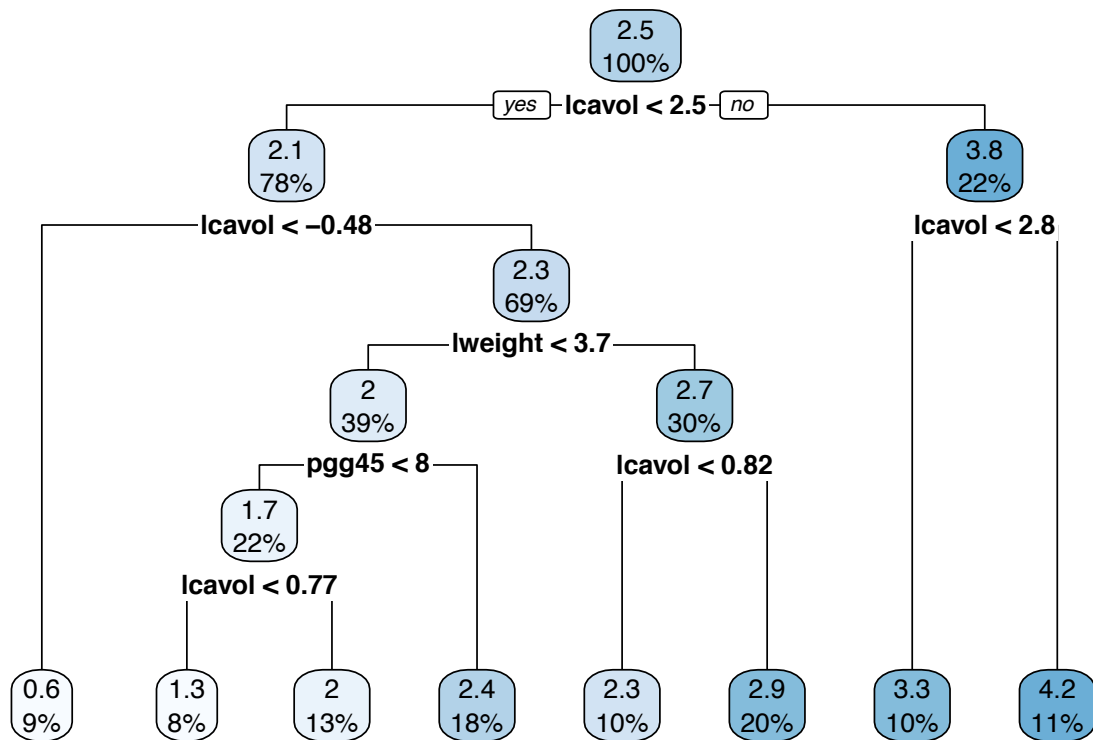
```
##      lcavol  lweight age      lbph svi      lcp gleason pgg45      lpsa
## 1 -0.5798185 2.769459  50 -1.386294  0 -1.386294      6      0 -0.4307829
## 2 -0.9942523 3.319626  58 -1.386294  0 -1.386294      6      0 -0.1625189
## 3 -0.5108256 2.691243  74 -1.386294  0 -1.386294      7     20 -0.1625189
## 4 -1.2039728 3.282789  58 -1.386294  0 -1.386294      6      0 -0.1625189
## 5  0.7514161 3.432373  62 -1.386294  0 -1.386294      6      0  0.3715636
## 6 -1.0498221 3.228826  50 -1.386294  0 -1.386294      6      0  0.7654678
```

```
set.seed(1)
rowtrain = createDataPartition(y = Prostate$lpsa,
                                p = 0.75,
                                list = FALSE)
ctrl1 = trainControl(method = "cv")
```

1a) Fit a regression tree with `lpsa` as the response; Use cross-validation to determine the optimal tree size. Which tree size corresponds to the lowest cross-validation error? Is this the same as the tree size obtained using the 1 SE rule?

```
set.seed(1)
rpart.fit = train(lpsa~., Prostate,
                  method = "rpart",
                  tuneGrid = data.frame(cp = exp(seq(-6,-2,length=20))), trControl = ctrl1)
rpart.plot(rpart.fit$finalModel)

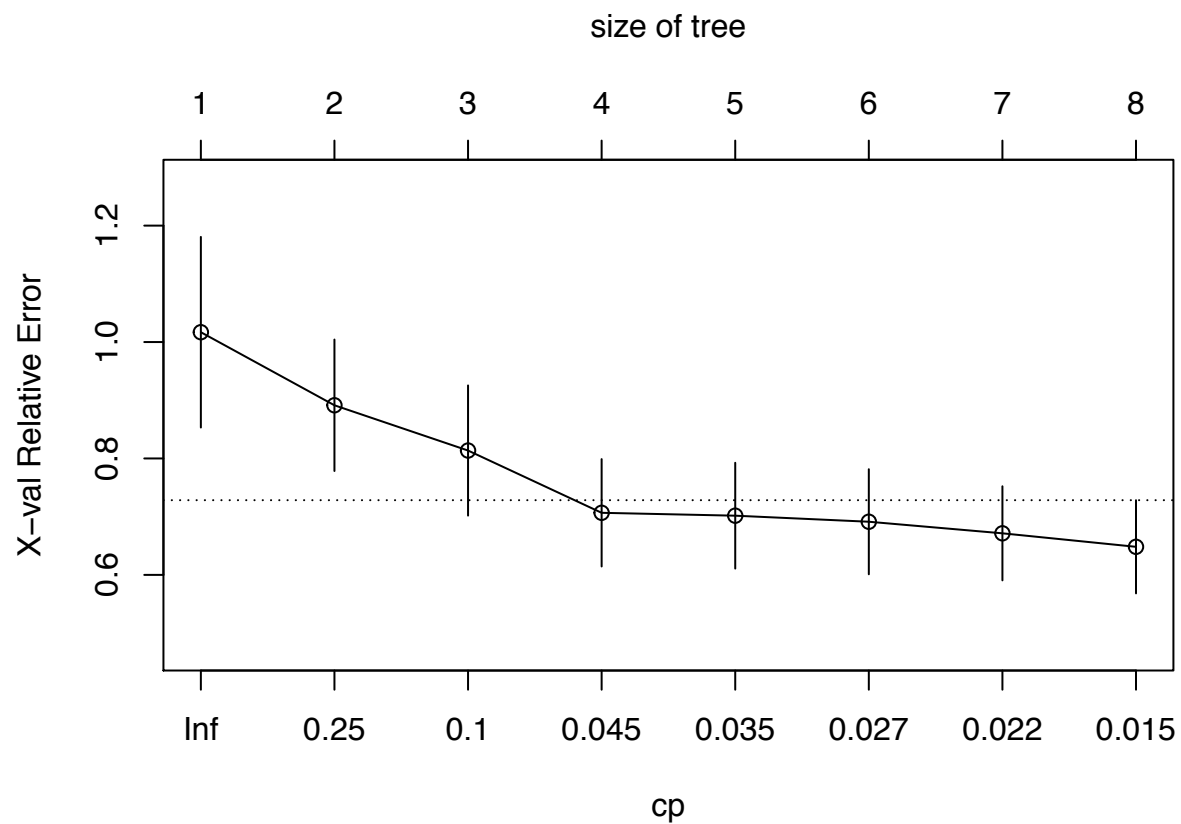
#####
set.seed(1)
tree1 = rpart(formula = lpsa~., data = Prostate)
rpart.plot(tree1)
```



```
cp_table = printcp(tree1)### cross validation built in rpart; large cp give smaller tree
```

```
##
## Regression tree:
## rpart(formula = lpsa ~ ., data = Prostate)
##
## Variables actually used in tree construction:
## [1] lcavol lweight pgg45
##
## Root node error: 127.92/97 = 1.3187
##
## n= 97
##
##      CP nsplit rel error  xerror   xstd
## 1 0.347108      0  1.00000 1.01687 0.163742
## 2 0.184647      1  0.65289 0.89137 0.112926
## 3 0.059316      2  0.46824 0.81363 0.111838
## 4 0.034756      3  0.40893 0.70667 0.092263
## 5 0.034609      4  0.37417 0.70171 0.090879
## 6 0.021564      5  0.33956 0.69128 0.090257
## 7 0.021470      6  0.31800 0.67139 0.080849
## 8 0.010000      7  0.29653 0.64826 0.080048
```

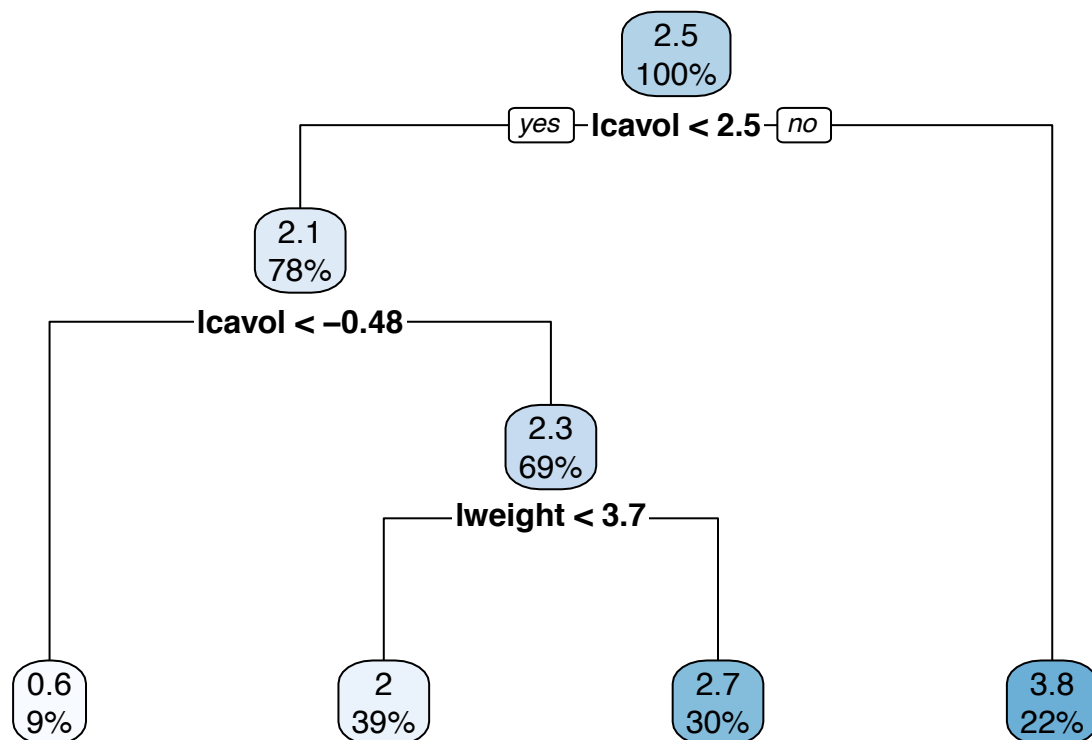
```
plotcp(tree1)
```



```

### tree by minimum cross validation error
minerror = which.min(cp_table[,4])
tree2 = prune(tree1, cp = cp_table[minerror,1])
### 1SE rule- simplest model with error smaller than the line
tree3 = prune(tree1, cp = cp_table[cp_table[,4]<cp_table[minerror,4]+cp_table[minerror,5],1][1])
# split 3
rpart.plot(tree3)

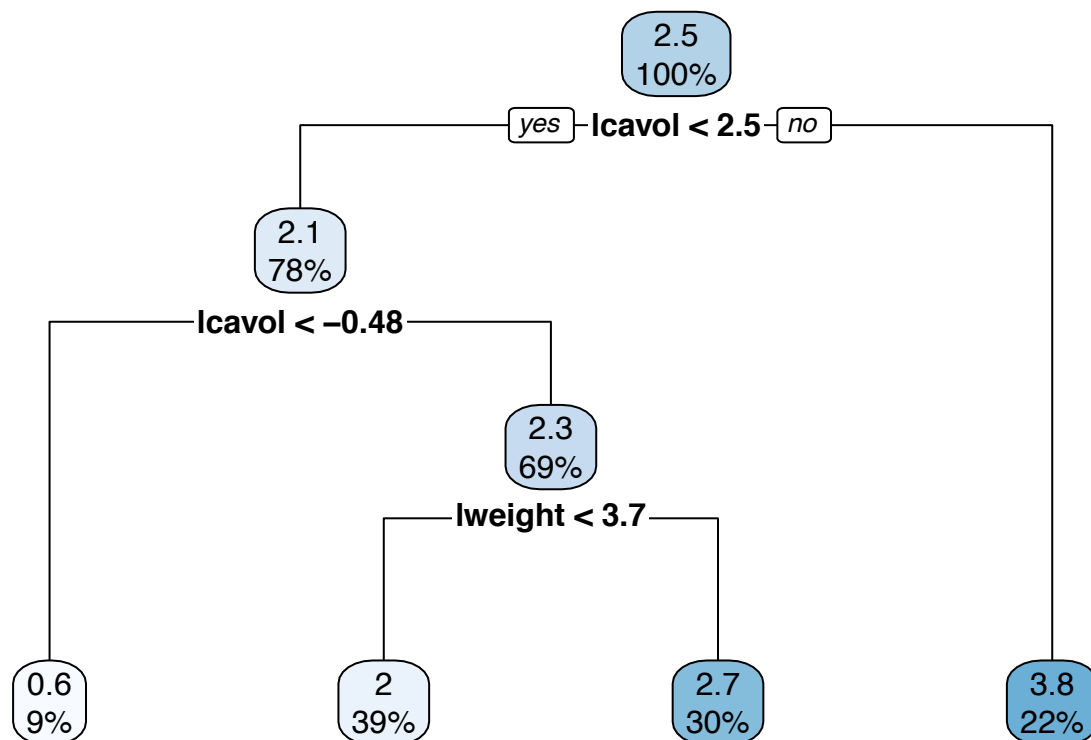
```



According to the plot, the optimal tree size is $7+1 = 8$ by cross validation. According to 1SE rule, the optimal split is 3. Therefore, the tree size are different.

2 Create a plot of the final tree you choose. Pick one of the terminal nodes, and interpret the information displayed.

```
rpart.plot(tree3)
```

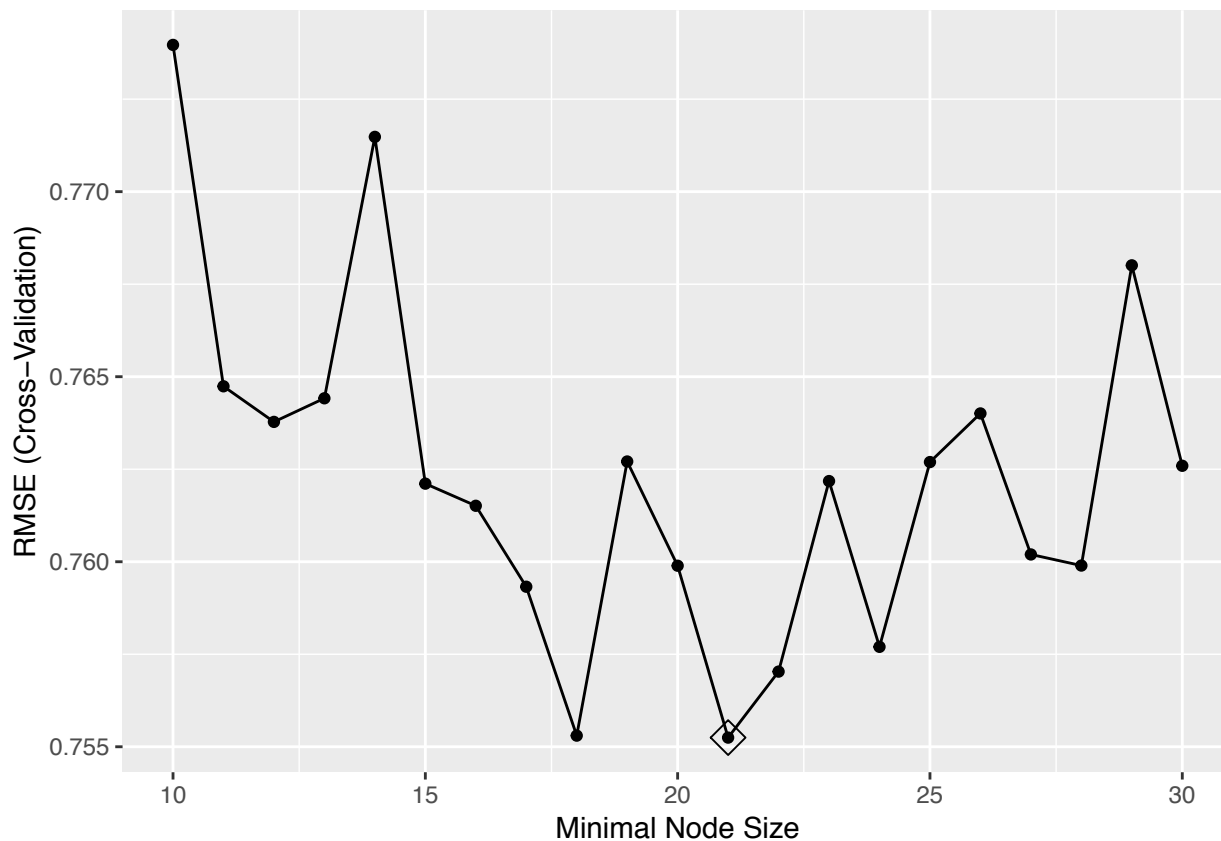


Since both models have similar cross validation error. We should choose the simpler model which is 1 SE model. Node 3.8 interpretation: when the log cancer volume is greater 2.5. the log mean of log prostate specific antigen is 3.8. The nodes include 22% response of training dataset.

(c) Perform bagging and report the variable importance.

```

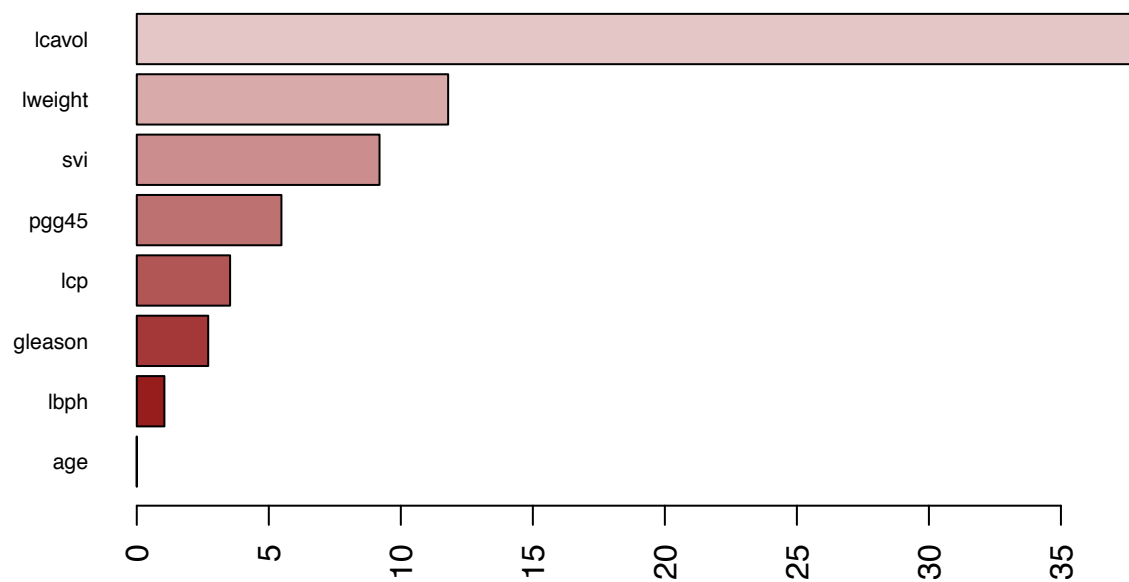
set.seed(2)
rf.grid = expand.grid(mtry = 8,
                      splitrule = "variance",
                      min.node.size = 10:30)
bagging.fit = train(lpsa~., Prostate,
                    method = "ranger",
                    tuneGrid = rf.grid,
                    trControl = ctrl)
ggplot(bagging.fit, highlight = TRUE)
  
```



```

bagging.imp = ranger(lpsa~., Prostate,
  mtry = 8,
  splitrule = "variance",
  min.node.size = 11,
  importance = "permutation",
  scale.permutation.importance = TRUE)
barplot(sort(ranger::importance(bagging.imp), decreasing = FALSE),
  las = 2, horiz = TRUE, cex.names = 0.7, col = colorRampPalette(colors = c("darkred", "white",

```



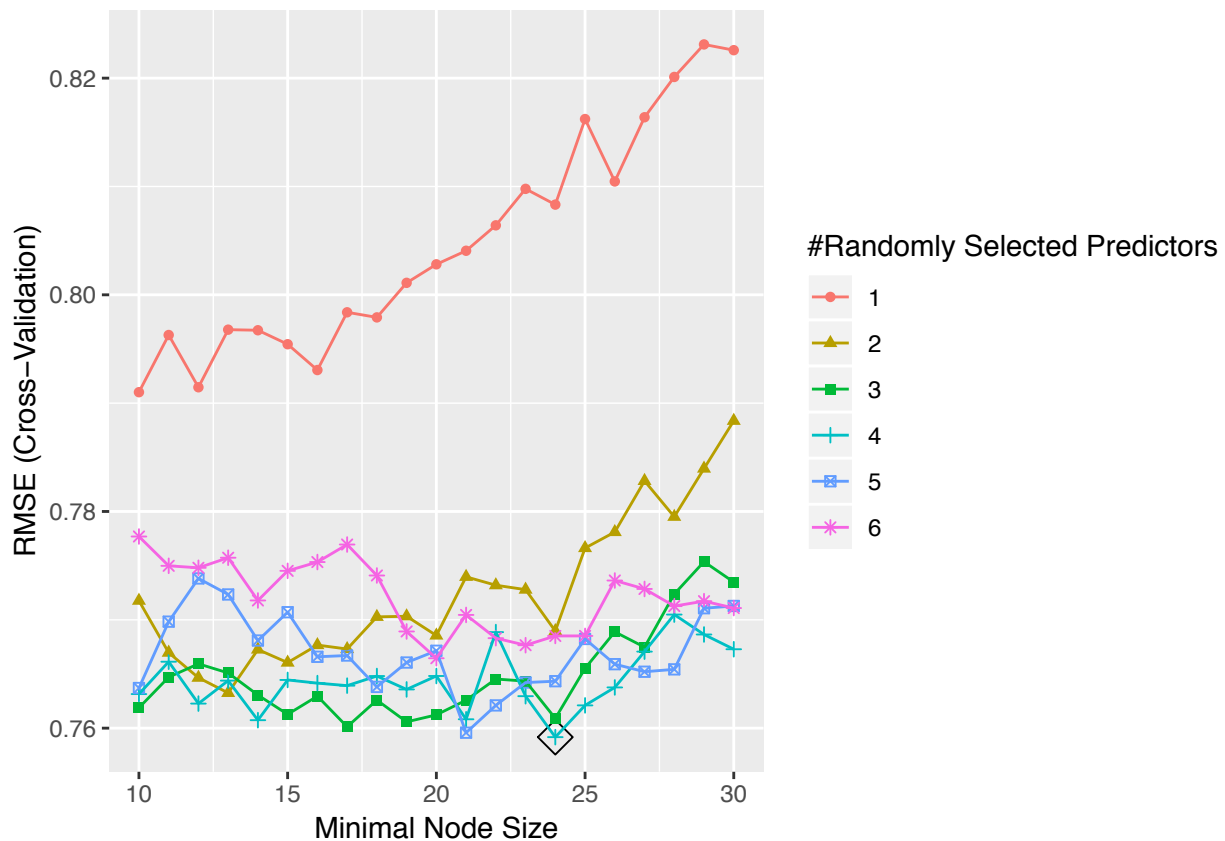
Based on the plot above, the variable importance is $lcavol > lweight > svi > pgg45 > lcp > gleason > lbph > age$.

(d) Perform random forests and report the variable importance.

```
### tuning parameter
rf.grid = expand.grid(mtry = 1:6,
                     splitrule = "variance",
                     min.node.size = 10:30)### ?

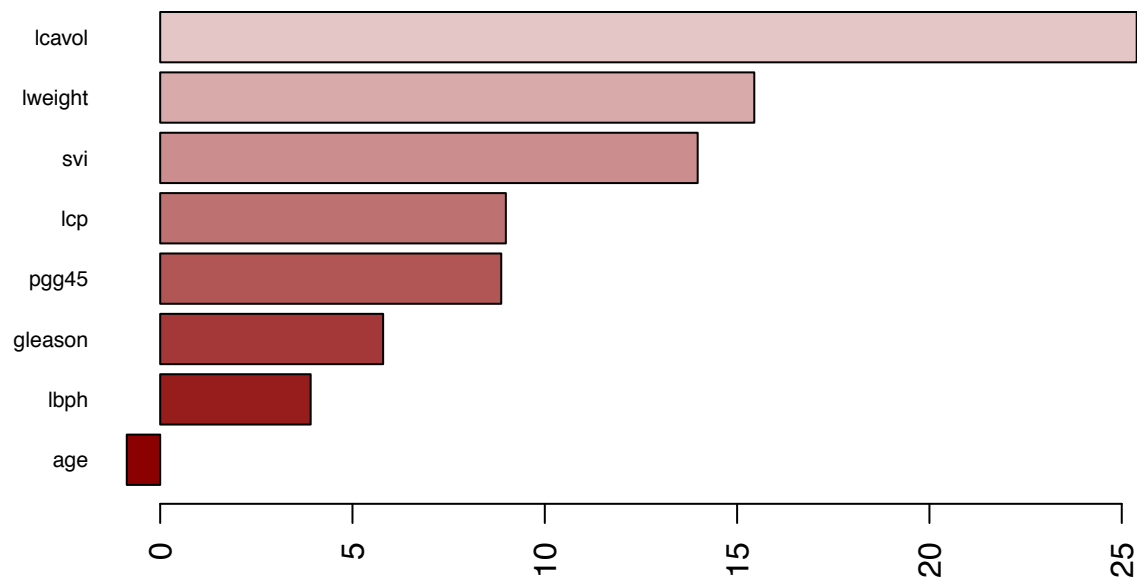
### fit model
set.seed(3)
rf.fit = train(lpsa~., Prostate,
               method = "ranger",
               tuneGrid = rf.grid,
               trControl = ctr1)

### tuning parameter choosen
ggplot(rf.fit, highlight = TRUE)
```



```
### variable importance.
rf.final.imp = ranger(lpsa~., Prostate,
                     mtry = 4, splitrule = "variance",
                     min.node.size = 24,
                     importance = "permutation",
                     scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf.final.imp), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7, col = colorRampPalette(colors = c("darkred", "white",
```

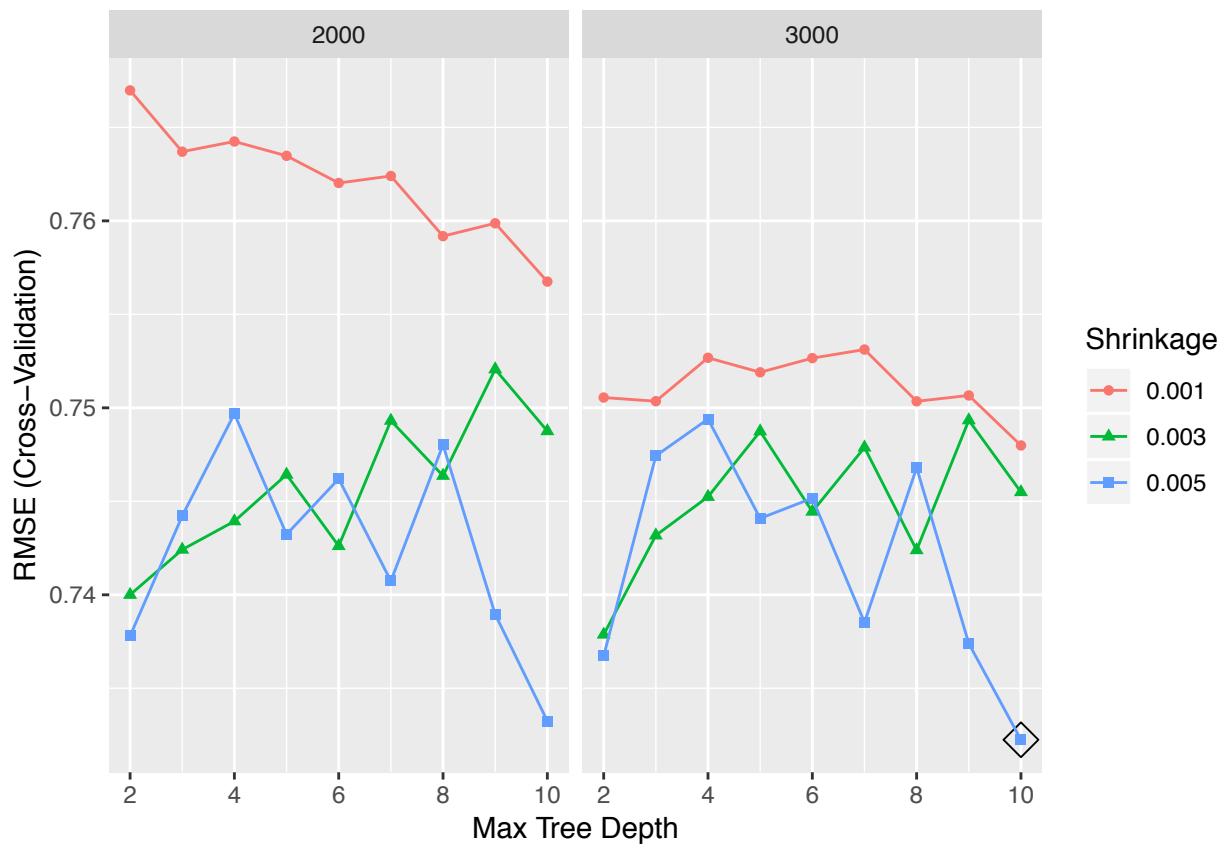


Based on the plot above, the variable importance is $lcavol > lweight > svi > lcp > pgg45 > gleason > lbph > age$.

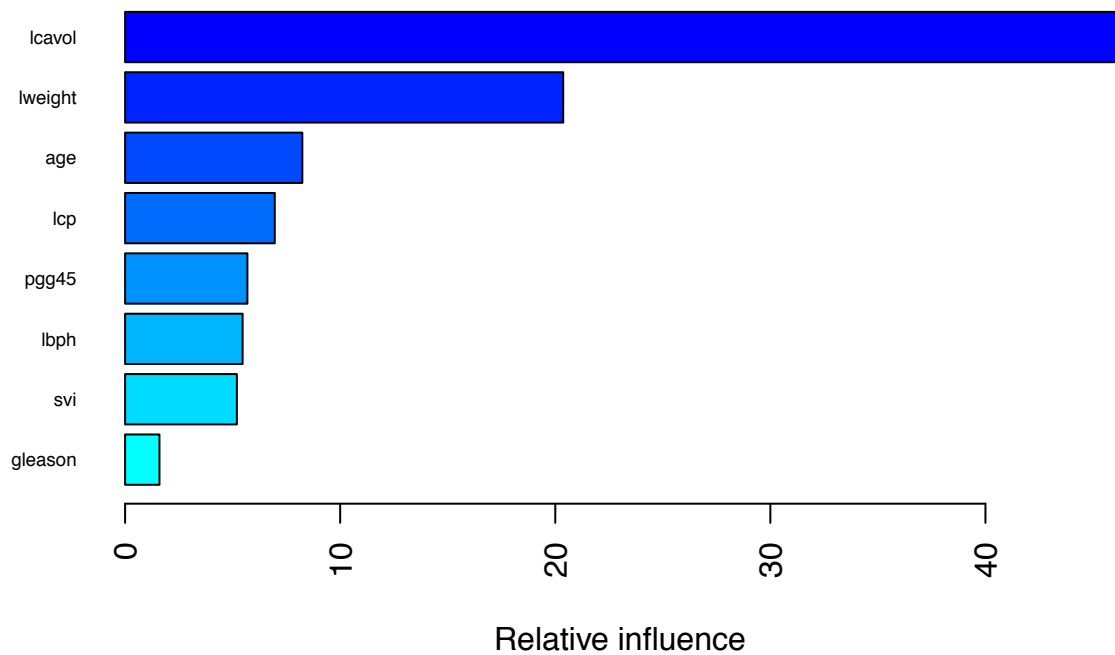
(e) Perform boosting and report the variable importance.

```
### tuning parameter
gbm.grid = expand.grid(n.trees = c(2000,3000),
                      interaction.depth = 2:10,
                      shrinkage = c(0.001, 0.003, 0.005),
                      n.minobsinnode = 1)

### fit model
gbm.fit = train(lpsa~., Prostate,
               method = "gbm",
               tuneGrid = gbm.grid,
               trControl = ctrl1,
               verbose = FALSE)
ggplot(gbm.fit, highlight = TRUE)
```

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



```
##      var  rel.inf
## lcavol  lcavol 46.491222
## lweight lweight 20.369375
## age     age    8.239670
```

```
## lcp          lcp  6.957807
## pgg45       pgg45  5.685430
## lbph        lbph  5.463286
## svi         svi   5.195660
## gleason     gleason 1.597550
```

Based on the plot above, the variable importance is lcavol > lweight > lcp > age > svi > pgg45 > lbph > gleason

(f) Which of the above models will you select to predict PSA level? Explain

```
resamp = resamples(list(rpart = rpart.fit, bagging = bagging.fit, rf = rf.fit, gbm = gbm.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: rpart, bagging, rf, gbm
## Number of resamples: 10
##
## MAE
##      Min.    1st Qu.    Median      Mean   3rd Qu.    Max. NA's
## rpart  0.6029257 0.6833061 0.7132861 0.7481058 0.7878006 1.0108929    0
## bagging 0.4132455 0.5733499 0.6171783 0.6242907 0.6717655 0.8203720    0
## rf      0.3207104 0.4336260 0.6678633 0.6367605 0.7578791 1.0413658    0
## gbm     0.4073739 0.5278648 0.5467414 0.5822022 0.6558448 0.8031722    0
##
## RMSE
##      Min.    1st Qu.    Median      Mean   3rd Qu.    Max. NA's
## rpart  0.6940977 0.7916380 0.8659767 0.8688293 0.9202064 1.1161271    0
## bagging 0.5690718 0.6507318 0.7658300 0.7552480 0.8558485 0.9142573    0
## rf      0.3852574 0.5493899 0.8002497 0.7591698 0.9212254 1.1998445    0
## gbm     0.4994624 0.6739685 0.7284938 0.7322453 0.7914498 0.9575104    0
##
## Rsquared
##      Min.    1st Qu.    Median      Mean   3rd Qu.    Max. NA's
## rpart  0.10742511 0.4435906 0.5507225 0.4869429 0.5676906 0.6732530    0
## bagging 0.30582778 0.5199532 0.6269879 0.6297696 0.7633807 0.9030913    0
## rf      0.05163731 0.4903155 0.6836592 0.5723541 0.7337990 0.8431251    0
## gbm     0.18153923 0.4188219 0.6308190 0.5685605 0.7310841 0.7685486    0
```

Boosting model is selected because the cross validation error of boosting model is smallest among these models.

Problem 2

```
data(OJ)
levels(OJ$Purchase)
```

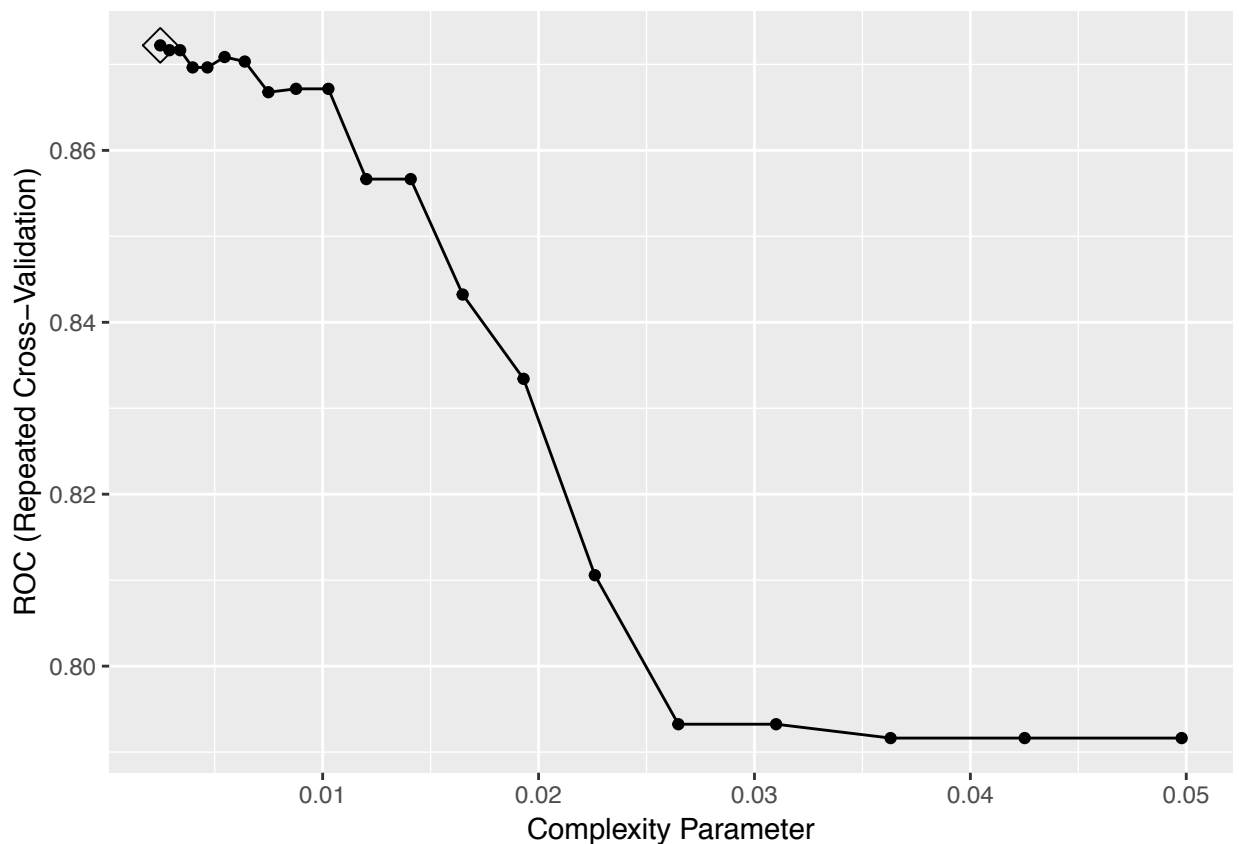
```
## [1] "CH" "MM"
```

```
rowtrain2 = createDataPartition(y = OJ$Purchase,
                                p = 800/1070,
                                list = FALSE)
```

(a) Fit a classification tree to the training set, 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. Predict the response on the test data. What is the test classification error rate?

```
ctrl1 = trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

set.seed(1)
rpart.fit2 = train(Purchase ~ ., OJ,
                   subset = rowtrain2,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6, -3, len=20))),
                   trControl = ctrl1,
                   metric = "ROC")
ggplot(rpart.fit2, highlight = TRUE)
```

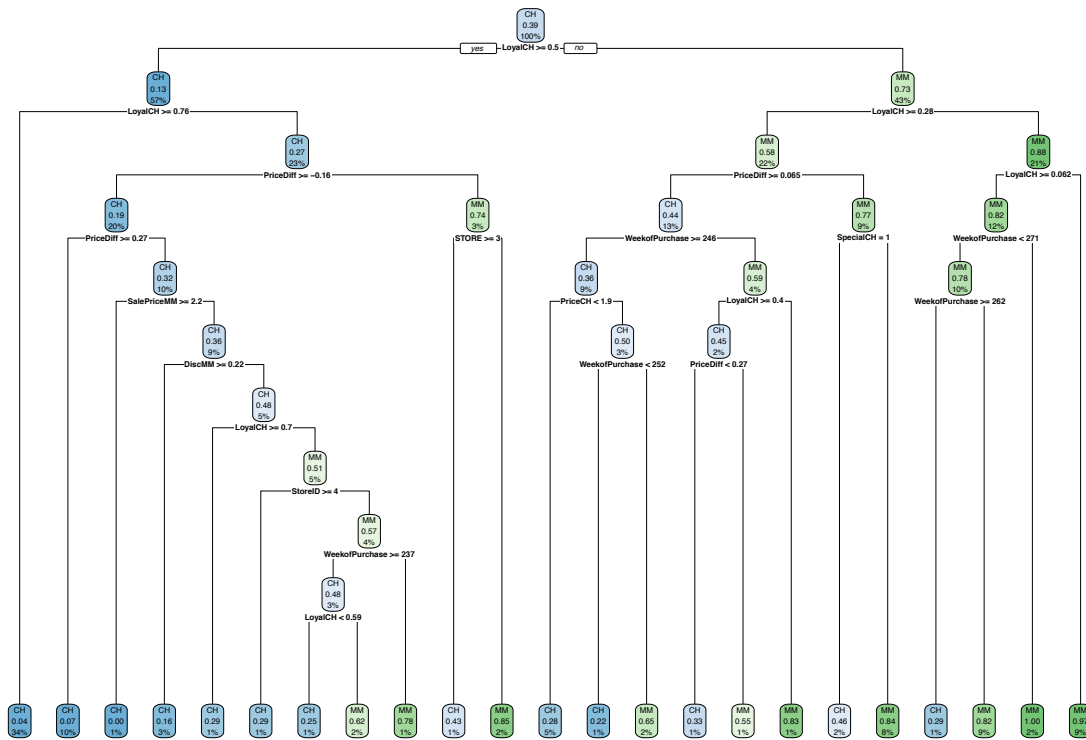


```
rpart.fit2$finalModel$cptable
```

```
##          CP nsplit rel error
## 1 0.503205128    0 1.0000000
## 2 0.020833333    1 0.4967949
## 3 0.019230769    5 0.4134615
```

```
## 4 0.008012821      6 0.3942308
## 5 0.006410256      8 0.3782051
## 6 0.003205128      9 0.3717949
## 7 0.002564103     15 0.3525641
## 8 0.002478752     22 0.3269231
```

```
### plot of final tree
rpart.plot(rpart.fit2$finalModel)
```



```
### predict
pred = predict(rpart.fit2, newdata = OJ[-rowtrain2,])
### test classification error rate
mean(pred != OJ[-rowtrain2,]$Purchase)
```

```
## [1] 0.1821561
```

According to cross validation, the number of split is 10 and the optimal tree size is 11. The test classification error rate is 0.1858736.

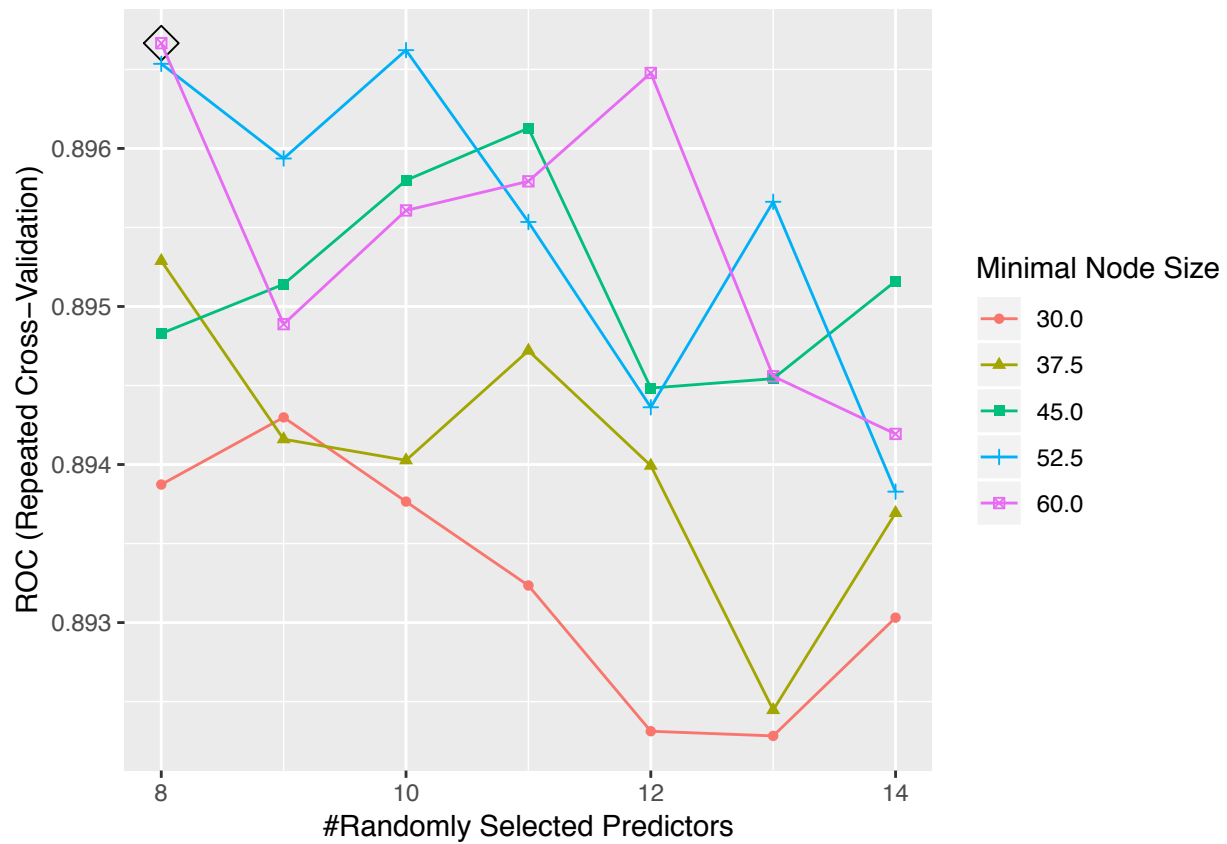
(b) Perform random forests on the training set and report variable importance. What is the test error rate?

```
rf.grid2 = expand.grid(mtry = 8:14,
                      splitrule = "gini",
                      min.node.size = seq(30,60,length = 5))

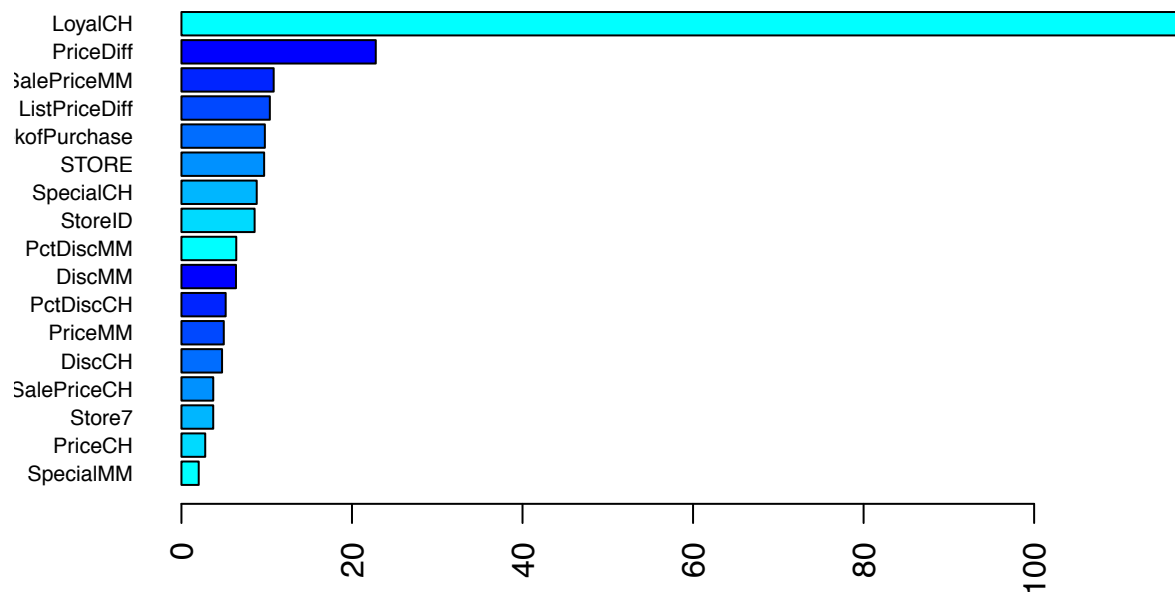
set.seed(1)
rf.fit2 = train(Purchase~., OJ,
                subset = rowtrain2,
                method = "ranger",
                tuneGrid = rf.grid2,
                metric = "ROC",
```

```
trControl = ctr1)

ggplot(rf.fit2, highlight = TRUE)
```



```
### VARIABLE IMPORTANCE
rf2.final.imp = ranger(Purchase~., OJ[rowtrain2,],
  mtry = 9 ,
  min.node.size = 45,
  splitrule = "gini",
  importance = "permutation",
  scale.permutation.importance = TRUE)
barplot(sort(ranger::importance(rf2.final.imp), decreasing = FALSE),
  las = 2, hori = TRUE, cex.names = 0.7,
  col = colorRampPalette(colors = c("cyan", "blue"))(8))
```



```
### test error rate
```

```
pred_rf = predict(rf.fit2, newdata = OJ[-rowtrain2,])
mean(pred_rf != OJ[-rowtrain2,]$Purchase)
```

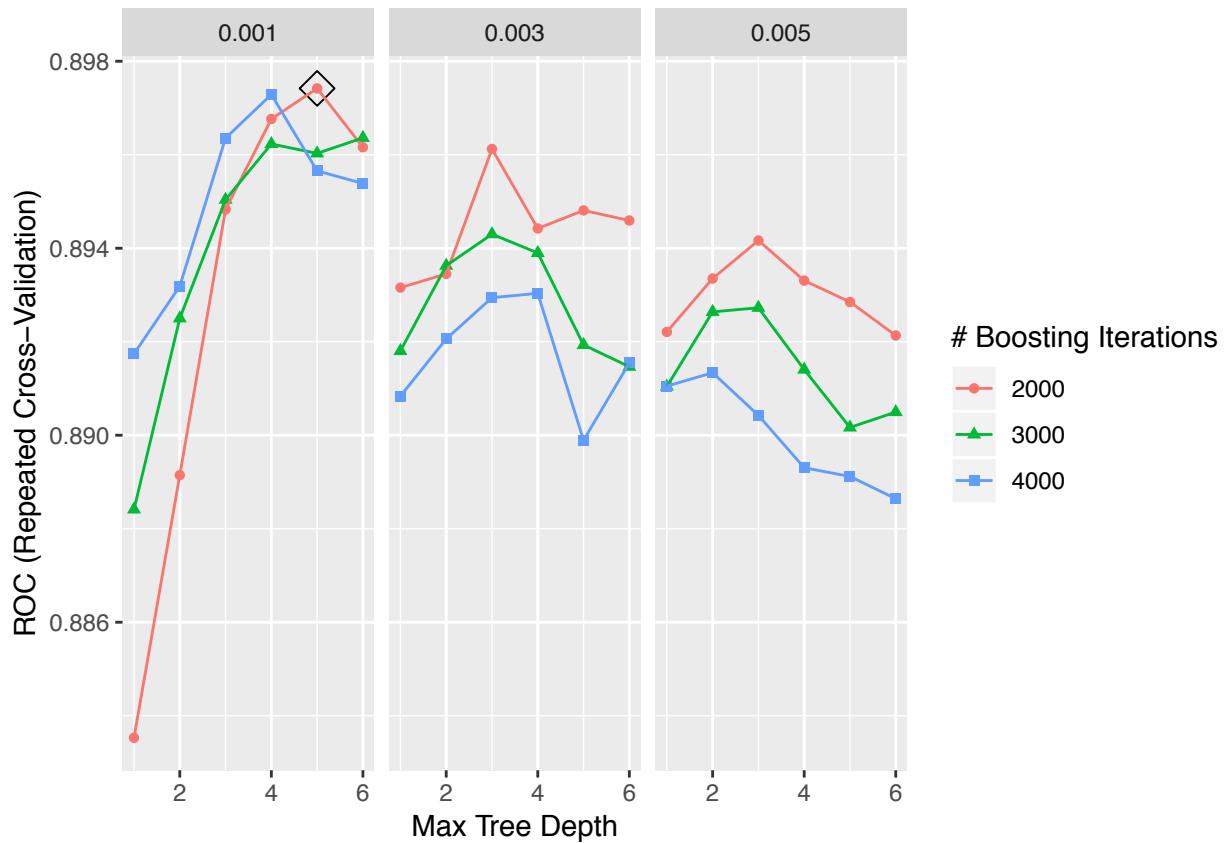
```
## [1] 0.1710037
```

The importance of variable was shown above.

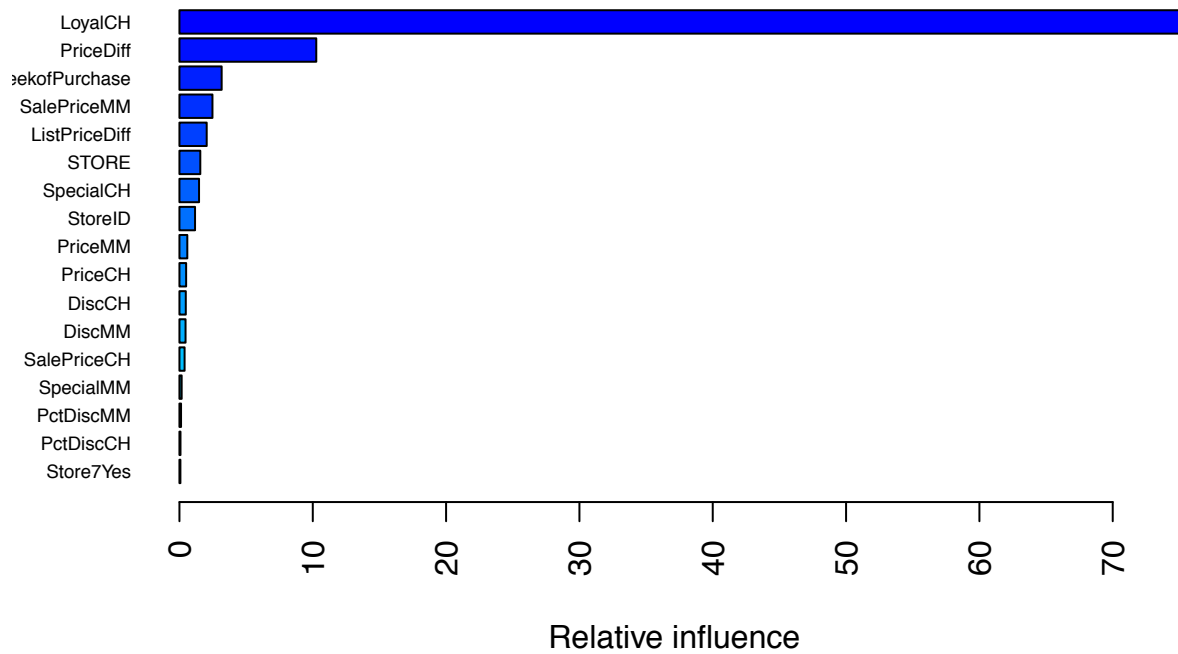
The test classification error rate is 0.1821561.

(c) Perform boosting on the training set and report variable importance. What is the test error rate?

```
gbm2.grid = expand.grid(n.trees = c(2000, 3000, 4000),
                        interaction.depth = 1:6,
                        shrinkage = c(0.001, 0.003, 0.005),
                        n.minobsinnode = 1)
gbm2.fit = train(Purchase ~ ., OJ[rowtrain2,],
                  tuneGrid = gbm2.grid,
                  trControl = ctrl1,
                  method = "gbm",
                  distribution = "bernoulli",
                  metric = "ROC",
                  verbose = FALSE)
ggplot(gbm2.fit, highlight = TRUE)
```



```
### important variable
summary(gbm2.fit$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```



```
##          var      rel.inf
## LoyalCH    LoyalCH 75.00017781
## PriceDiff  PriceDiff 10.26319526
```

```
## WeekofPurchase WeekofPurchase 3.16546239
## SalePriceMM      SalePriceMM 2.47592453
## ListPriceDiff    ListPriceDiff 2.04450990
## STORE            STORE 1.56394329
## SpecialCH        SpecialCH 1.46930112
## StoreID          StoreID 1.17043846
## PriceMM          PriceMM 0.59068255
## PriceCH          PriceCH 0.50269348
## DiscCH           DiscCH 0.47539396
## DiscMM           DiscMM 0.45799764
## SalePriceCH      SalePriceCH 0.38504642
## SpecialMM        SpecialMM 0.16546476
## PctDiscMM        PctDiscMM 0.12021707
## PctDiscCH        PctDiscCH 0.07733877
## Store7Yes        Store7Yes 0.07221257
```

```
### test error rate
```

```
pred_gbm2 = predict(gbm2.fit, newdata = OJ[-rowtrain2,])
mean(pred_gbm2!=OJ[-rowtrain2,]$Purchase)
```

```
## [1] 0.1710037
```

The importance of variable was shown above. The test classification error rate is 0.1747212.