Homework 4

Xinyi Lin

4/19/2019

```r
library(lasso2) # only for data
library(rpart) # for cart model
library(rpart.plot)
library(randomForest)
library(ranger)
library(caret)
library(gbm) # for boosting model
library(ISLR)
```

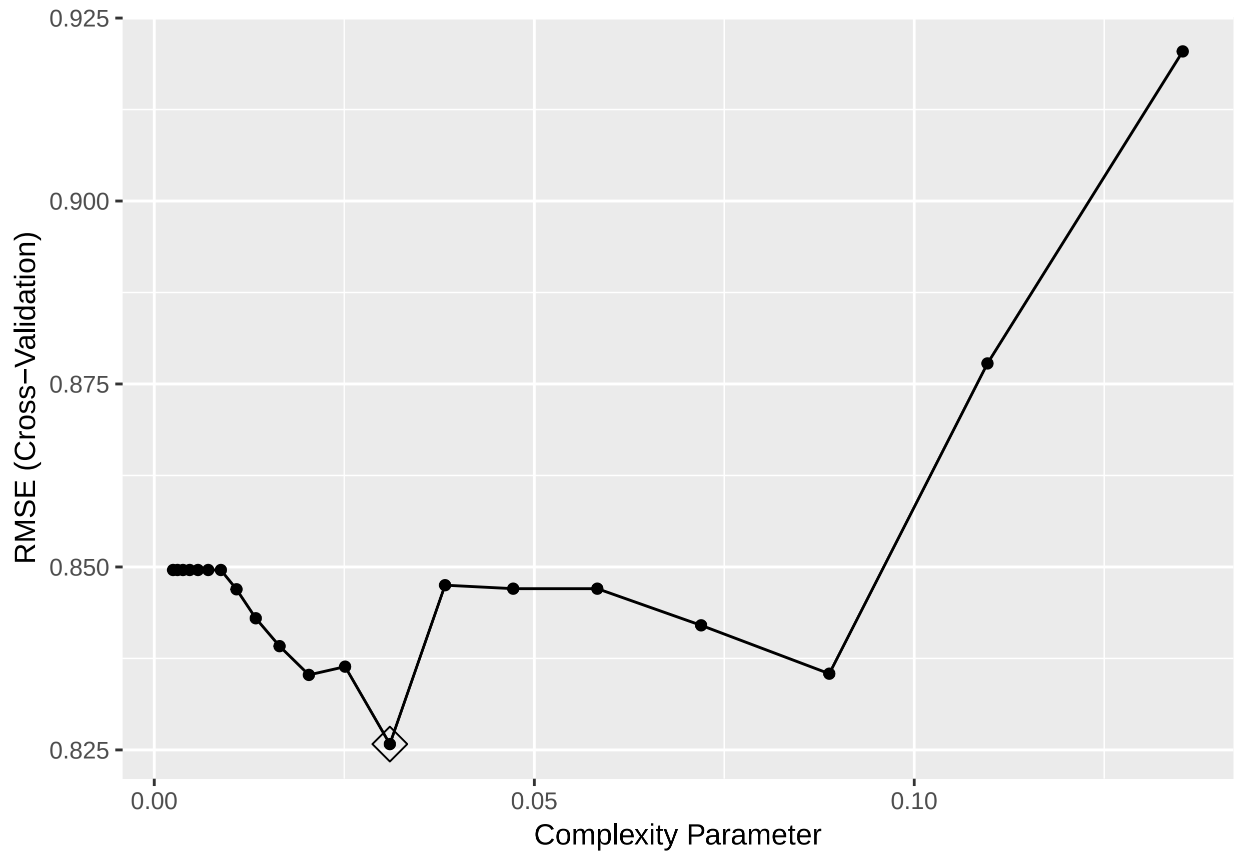
```r
data(Prostate)
```

Problem 1

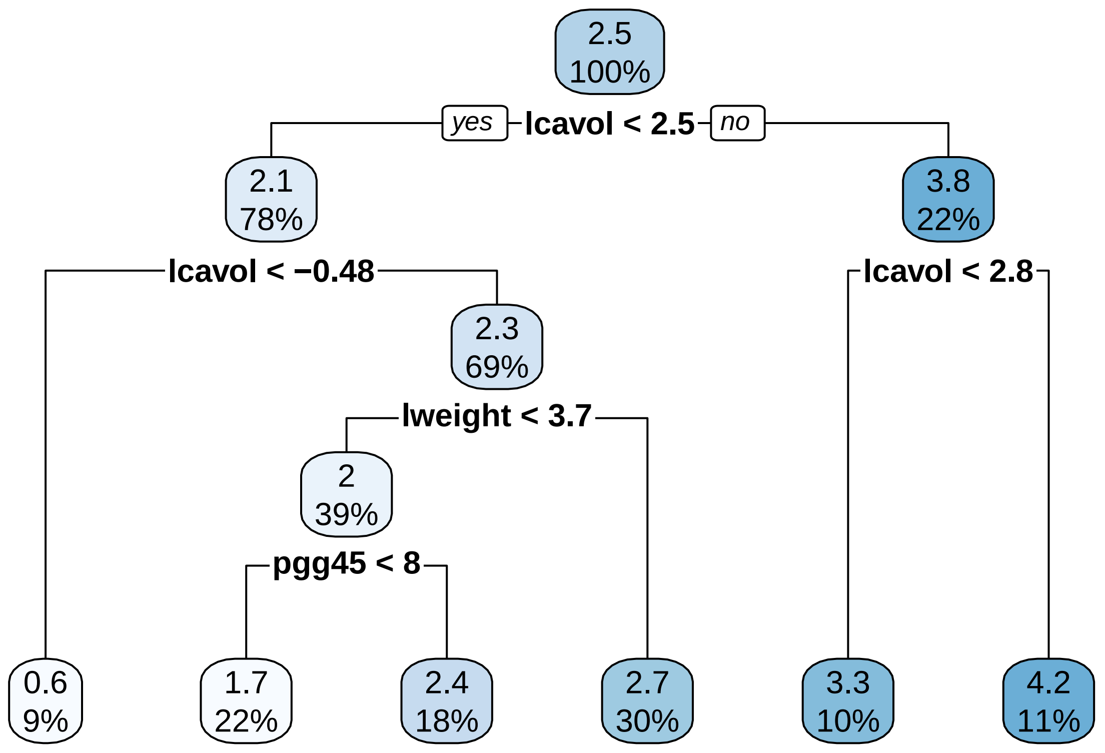
Question 1

Fit the regression tree. Use cross-validation to determine the optimal tree size. The following is the optimal tree.

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

set.seed(123)
rpart.fit1 <- train(lpsa~., Prostate,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 20))),
                    trControl = ctrl)
ggplot(rpart.fit1, highlight = TRUE)
```



```
rpart.plot(rpart.fit1$finalModel)
```



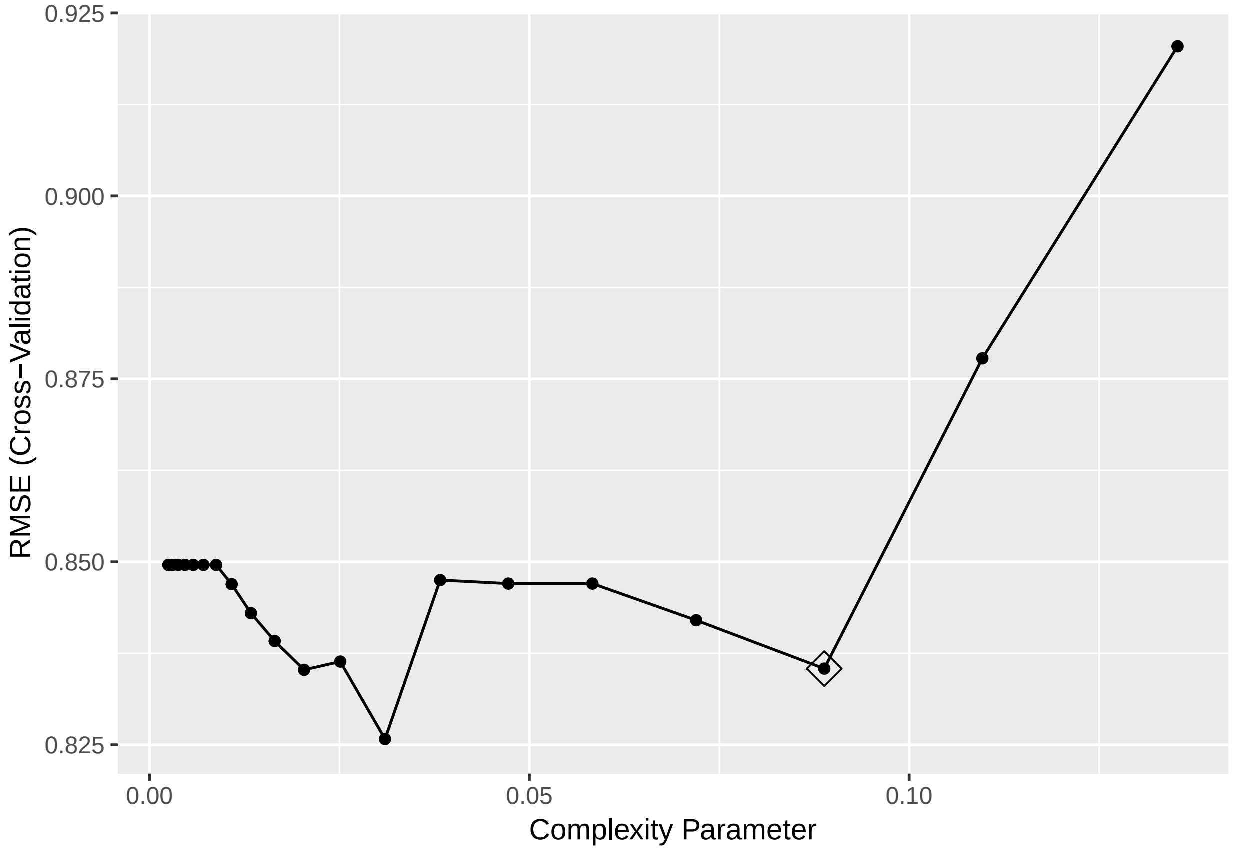
```r
rpart.fit1$finalModel$cptable
```

```
##           CP nsplit rel error
## 1 0.34710828      0 1.0000000
## 2 0.18464743      1 0.6528917
## 3 0.05931585      2 0.4682443
## 4 0.03475635      3 0.4089284
## 5 0.03460901      4 0.3741721
## 6 0.03100260      5 0.3395631
```

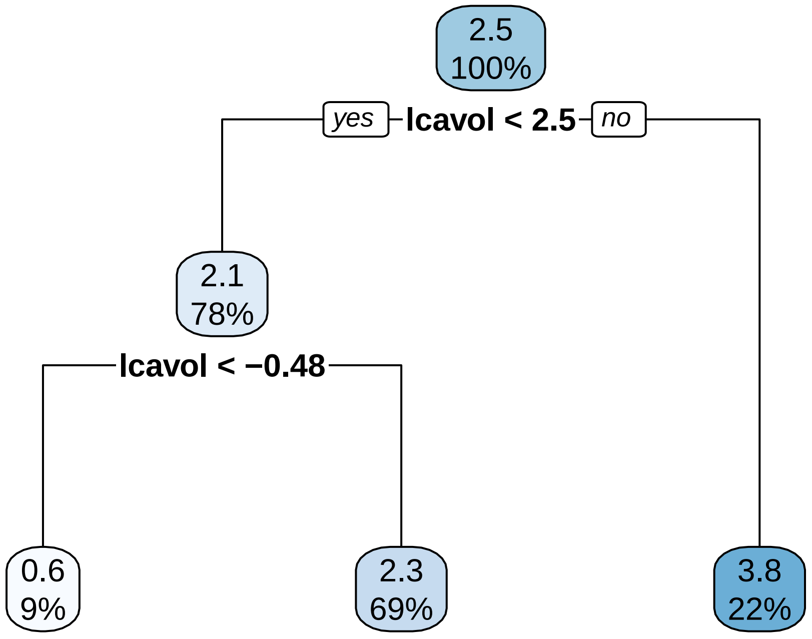
According to the plot and results given by `cptable`, we can find that when the number of splits equals to 6, the tree have lowest cross-validation error.

Using the 1 SE rule to obtain optimal tree size.

```r
# 1SE rule
set.seed(123)
rpart.fit2 <- train(lpsa~., Prostate,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 20))),
                    trControl = trainControl(method = "cv",
                                             number = 10,
                                             selectionFunction = "oneSE"))
ggplot(rpart.fit2, highlight = TRUE)
```



```r
rpart.plot(rpart.fit2$finalModel)
```



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

```
##           CP nsplit rel error
## 1 0.34710828      0 1.0000000
## 2 0.18464743      1 0.6528917
## 3 0.08882807      2 0.4682443
```

According to the plot and results given by `cptable`, we can find that when the number of splits equals to 3, the tree have lowest cross-validation error based on 1 SE rule. By comparing two tree, we can find that optimal tree sizes given by cross-validation and 1 SE rule are different.

Question 2

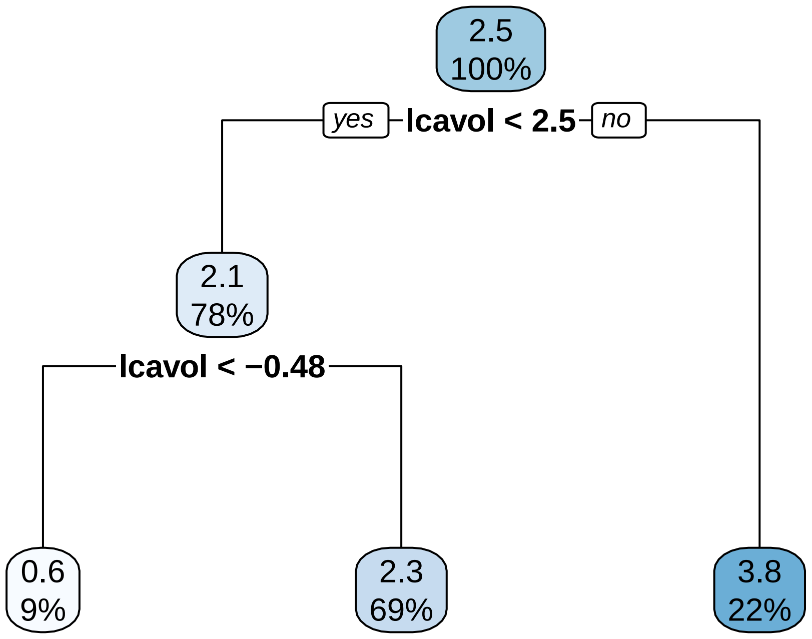
First, we compare cross-validation errors of two models.

```r
resamp1 <- resamples(list(rpart.fit1 = rpart.fit1, rpart.fit2 = rpart.fit2))
summary(resamp1)
```

```
##
## Call:
## summary.resamples(object = resamp1)
##
## Models: rpart.fit1, rpart.fit2
## Number of resamples: 10
##
## MAE
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## rpart.fit1 0.5429763 0.6663771 0.6905617 0.6913353 0.7162924 0.8224935
## rpart.fit2 0.5557089 0.6612255 0.6788911 0.7025058 0.7189591 0.9178099
##            NA's
## rpart.fit1    0
## rpart.fit2    0
##
## RMSE
##                  Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## rpart.fit1 0.6366077 0.7982196 0.8377900 0.8257973 0.8568897 1.001955    0
## rpart.fit2 0.6280811 0.7813385 0.8307383 0.8354147 0.8836407 1.052604    0
##
## Rsquared
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## rpart.fit1 0.002567189 0.3883509 0.5074496 0.5002326 0.6074653 0.8862399
## rpart.fit2 0.036571902 0.3635001 0.5492505 0.4932753 0.6440849 0.8682403
##            NA's
## rpart.fit1    0
## rpart.fit2    0
```

As their cross-validation errors are very close, we choose optimal tree obtained by 1 SE rule because it is simpler. Following are the final tree.

```r
rpart.plot(rpart.fit2$finalModel)
```



Interpretation of the node 3.8:

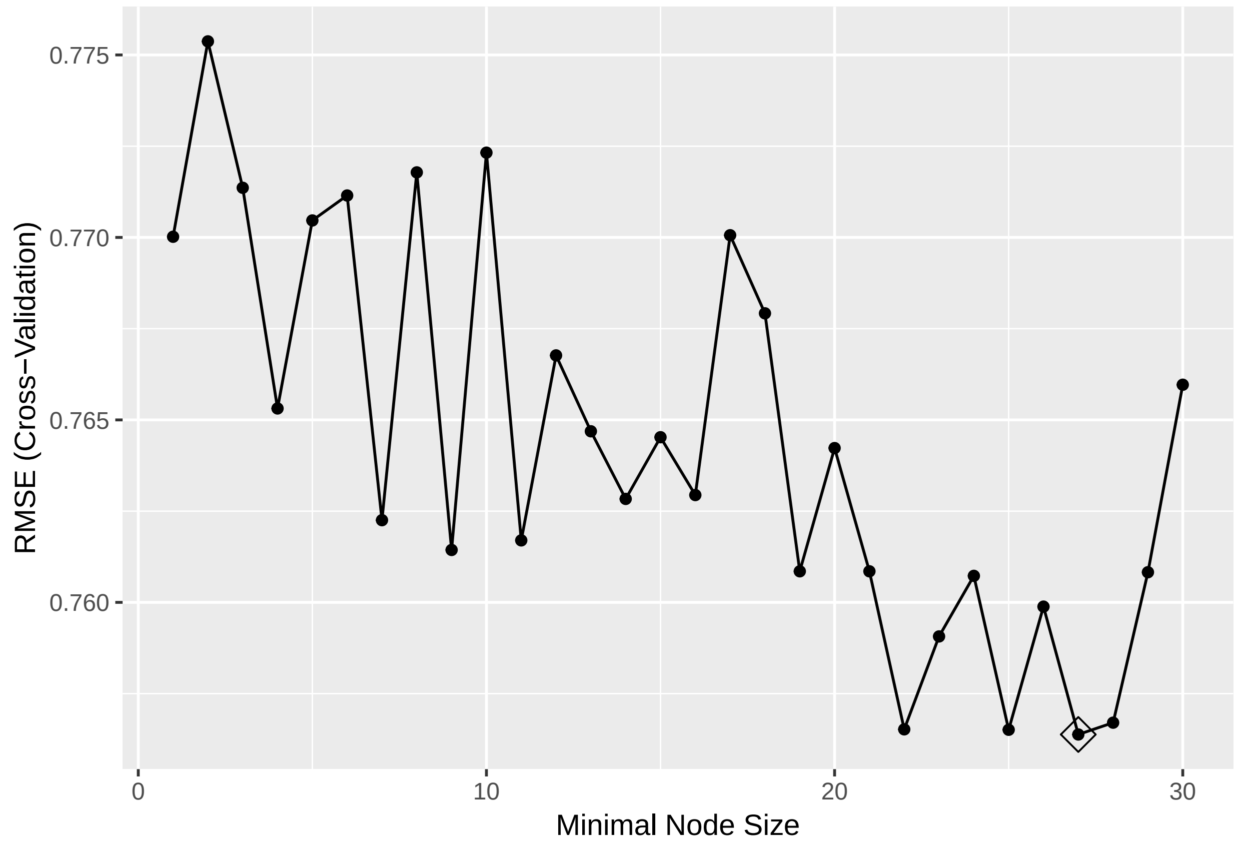
If the log of cancer volume is equals or larger than 2.5, than the mean of the log of prostate specific antigen is 3.8.This node contain 22% of training responses.

Question 3

Using `caret` package to find out the best minimal node size.

```r
rf.grid1 <- expand.grid(mtry = 8,
                        splitrule = "variance",
                        min.node.size = 1:30)
set.seed(123)
rf.fit1 <- train(lpsa~., Prostate,
                 method = "ranger",
                 tuneGrid = rf.grid1,
                 trControl = ctrl)
```

```r
ggplot(rf.fit1, highlight = TRUE)
```



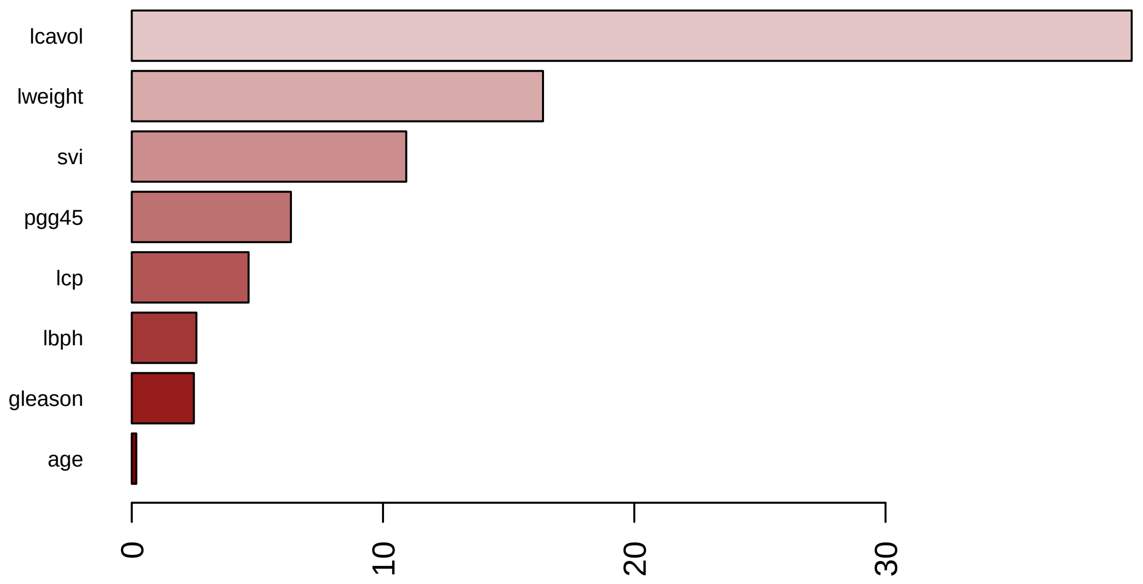
```r
rf.fit1$finalModel$min.node.size
```

```
## [1] 27
```

According to the result, the best minimal node size is 27.

Fit the bagging model and get the variable importance.

```r
bagging.final.per <- ranger(lpsa~., Prostate,
                            mtry = 8, splitrule = "variance",
                            min.node.size = 27,
                            importance = "permutation",
                            scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(bagging.final.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(19))
```

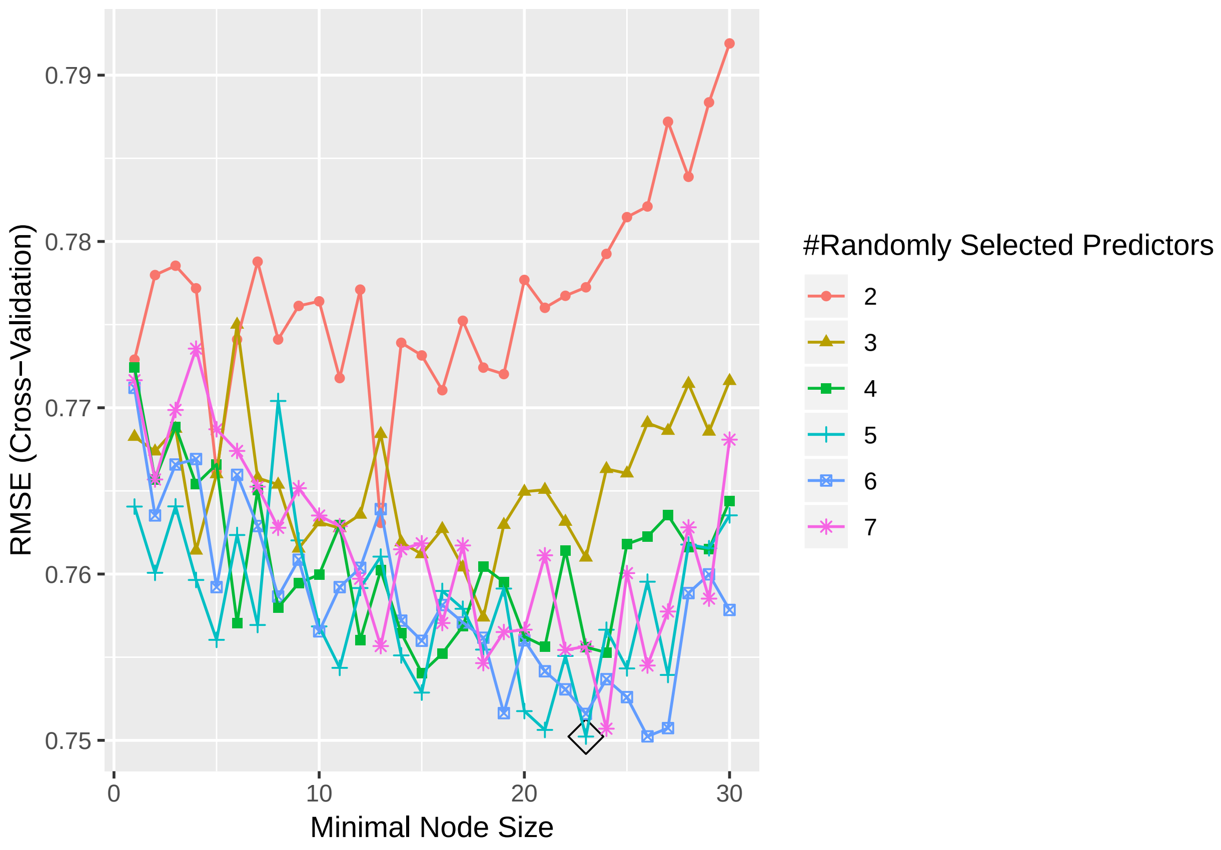


According to the plot above, the importance of each variable are `lcavol` > `lweight` > `svi` > `pgg45` > `lcp` > `lbph` > `gleason` > `age`.

Question 4

Using `caret` package to find out the best mtry and minimal node size.

```
rf.grid2 <- expand.grid(mtry = 2:7,
                        splitrule = "variance",
                        min.node.size = 1:30)
set.seed(123)
ctrl <- trainControl(method = "cv")
rf.fit2 <- train(lpsa~., Prostate,
                method = "ranger",
                tuneGrid = rf.grid2,
                trControl = ctrl)

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



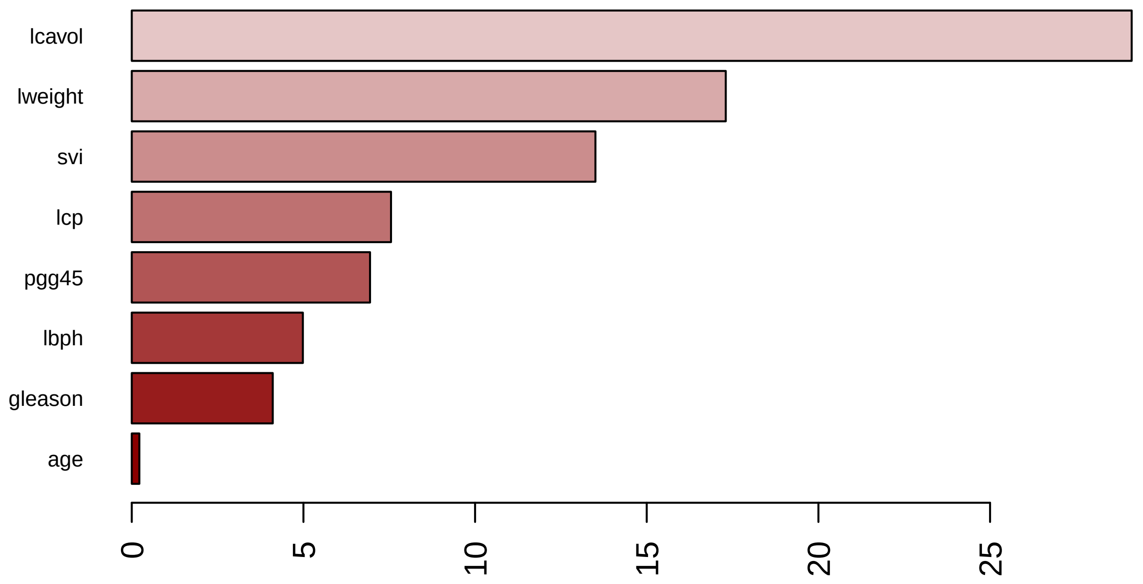
```
rf.fit1$finalModel$min.node.size
```

```
## [1] 27
```

According to the result, the best mtry is 5 and best minimal node size is 27.

Fit the random forests model and get the variable importance.

```
set.seed(123)
bagging.final.per <- ranger(lpsa~., Prostate,
                            mtry = 5, splitrule = "variance",
                            min.node.size = 27,
                            importance = "permutation",
                            scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(bagging.final.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(19))
```

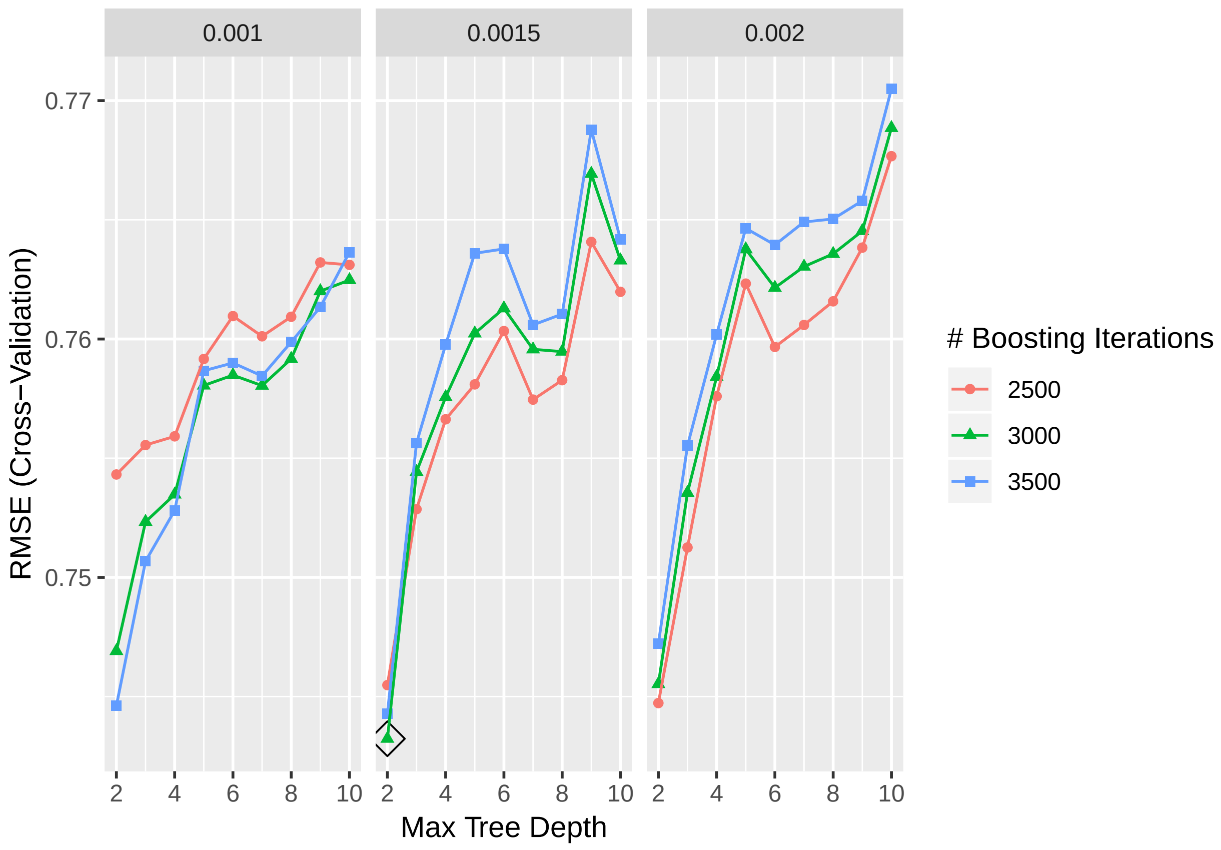


According to the plot above, the importance of each variable are `lcavol` > `lweight` > `svi` > `lcp` > `pgg45` > `lbph` > `gleason` > `age`.

Question 5

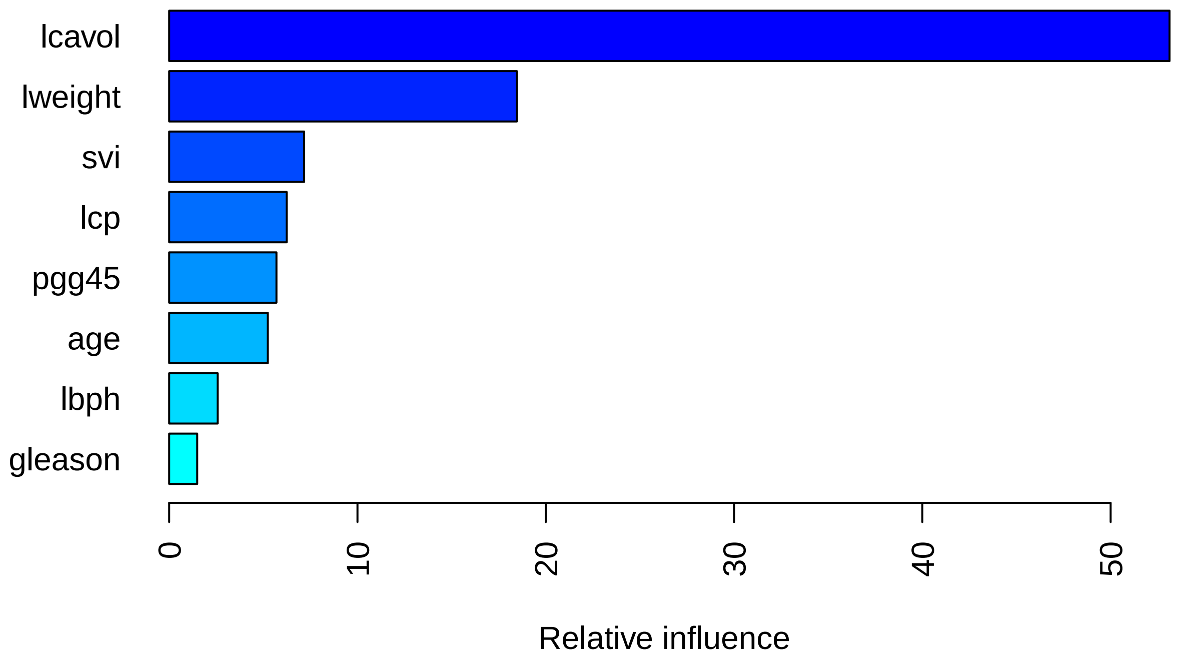
First, tune gbm model.

```
gbm.grid <- expand.grid(n.trees = c(2500,3000,3500),
                        interaction.depth = 2:10,
                        shrinkage = c(0.001,0.0015,0.002),
                        n.minobsinnode = 1)
set.seed(1)
gbm.fit <- train(lpsa~., Prostate,
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 verbose = FALSE)

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



Get the variable importance.

```r
summary(gbm.fit$finalModel, las = 2, cBars = 19, cex.names = 1)
```



```
##             var   rel.inf
## lcavol   lcavol 53.128137
## lweight lweight 18.465504
## svi          svi  7.169507
## lcp          lcp  6.240204
## pgg45      pgg45  5.697012
## age          age  5.236108
## lbph        lbph  2.575013
## gleason gleason  1.488513
```

According to the plot above, the importance of each variable are `lcavol` > `lweight` > `svi` > `lcp` > `pgg45` > `age` > `lbph` > `gleason`.

Question 6

```
resamp2 <- resamples(list(rpart.fit1 = rpart.fit1, rpart.fit2 = rpart.fit2, rf.fit1 = rf.fit1, rf.fit2 =
summary(resamp2)
```

```
##
## Call:
## summary.resamples(object = resamp2)
##
## Models: rpart.fit1, rpart.fit2, rf.fit1, rf.fit2, gbm.fit
## Number of resamples: 10
##
## MAE
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## rpart.fit1 0.5429763 0.6663771 0.6905617 0.6913353 0.7162924 0.8224935
## rpart.fit2 0.5557089 0.6612255 0.6788911 0.7025058 0.7189591 0.9178099
## rf.fit1    0.4126400 0.5939534 0.6200048 0.6240501 0.6471810 0.9036674
## rf.fit2    0.4217262 0.5708569 0.6135913 0.6165715 0.6554724 0.8437466
## gbm.fit    0.4181877 0.5326862 0.5686643 0.6109521 0.6529704 0.8898397
##            NA's
## rpart.fit1    0
## rpart.fit2    0
## rf.fit1       0
## rf.fit2       0
## gbm.fit       0
##
## RMSE
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## rpart.fit1 0.6366077 0.7982196 0.8377900 0.8257973 0.8568897 1.0019554
## rpart.fit2 0.6280811 0.7813385 0.8307383 0.8354147 0.8836407 1.0526044
## rf.fit1    0.4799010 0.7297126 0.7553279 0.7563801 0.7961595 0.9874921
## rf.fit2    0.4944191 0.7176596 0.7675318 0.7502277 0.8058060 0.9208883
## gbm.fit    0.5184725 0.6507144 0.7421628 0.7432312 0.7995495 0.9686217
##            NA's
## rpart.fit1    0
## rpart.fit2    0
## rf.fit1       0
## rf.fit2       0
## gbm.fit       0
##
## Rsquared
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## rpart.fit1 0.002567189 0.3883509 0.5074496 0.5002326 0.6074653 0.8862399
## rpart.fit2 0.036571902 0.3635001 0.5492505 0.4932753 0.6440849 0.8682403
## rf.fit1    0.031045040 0.4712345 0.5929200 0.5888112 0.7705369 0.9272054
## rf.fit2    0.112080510 0.4842046 0.6035318 0.6024839 0.7749519 0.9230549
## gbm.fit    0.374540607 0.5844846 0.6615999 0.6374203 0.7210225 0.7802402
##            NA's
## rpart.fit1    0
## rpart.fit2    0
## rf.fit1       0
## rf.fit2       0
## gbm.fit       0
```

According to results above, we can find that the cross-validation error of boosting model is the smallest, so the boosting model is the best model.

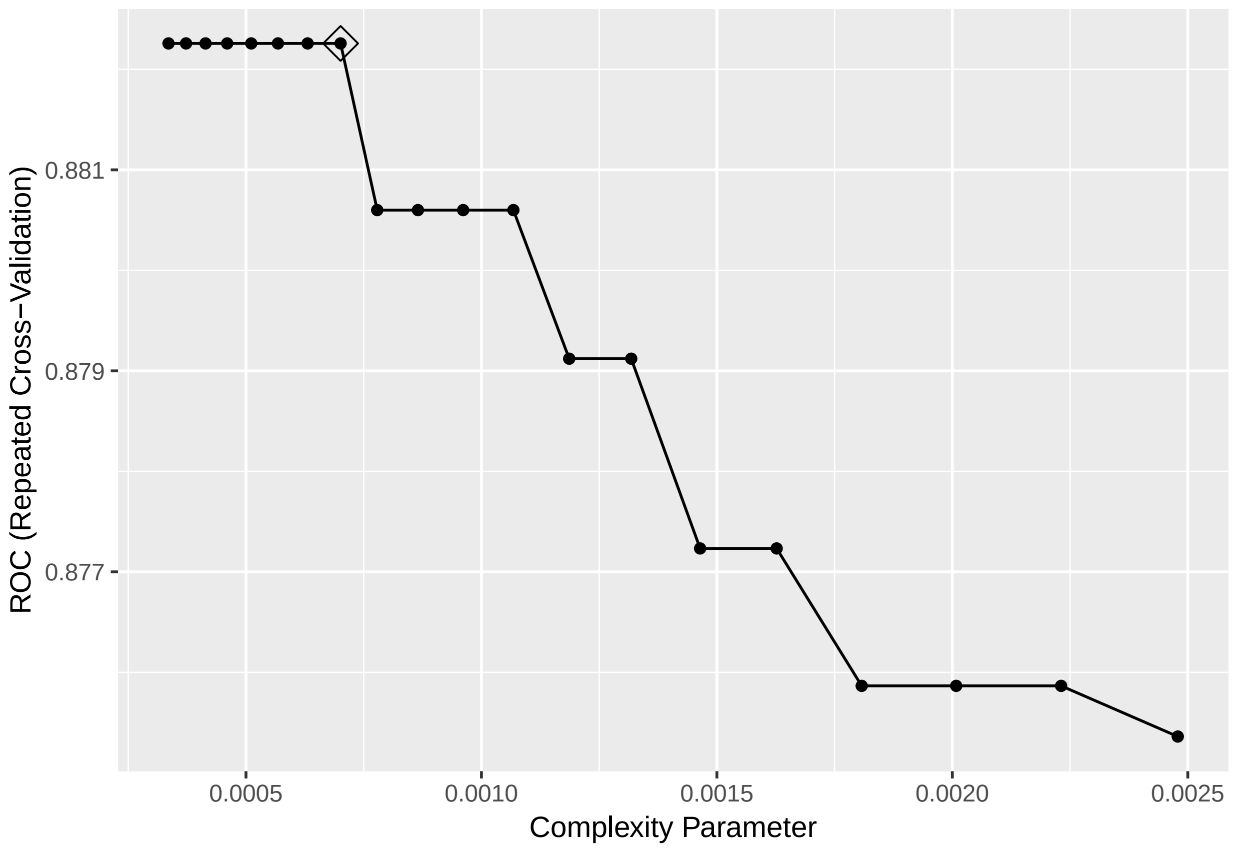
Problem 2

```r
OJ$Purchase = as.factor(OJ$Purchase)
set.seed(123)
train_ind <- sample(seq_len(nrow(OJ)), size = 800)

train <- OJ[train_ind, ]
test <- OJ[-train_ind, ]
```

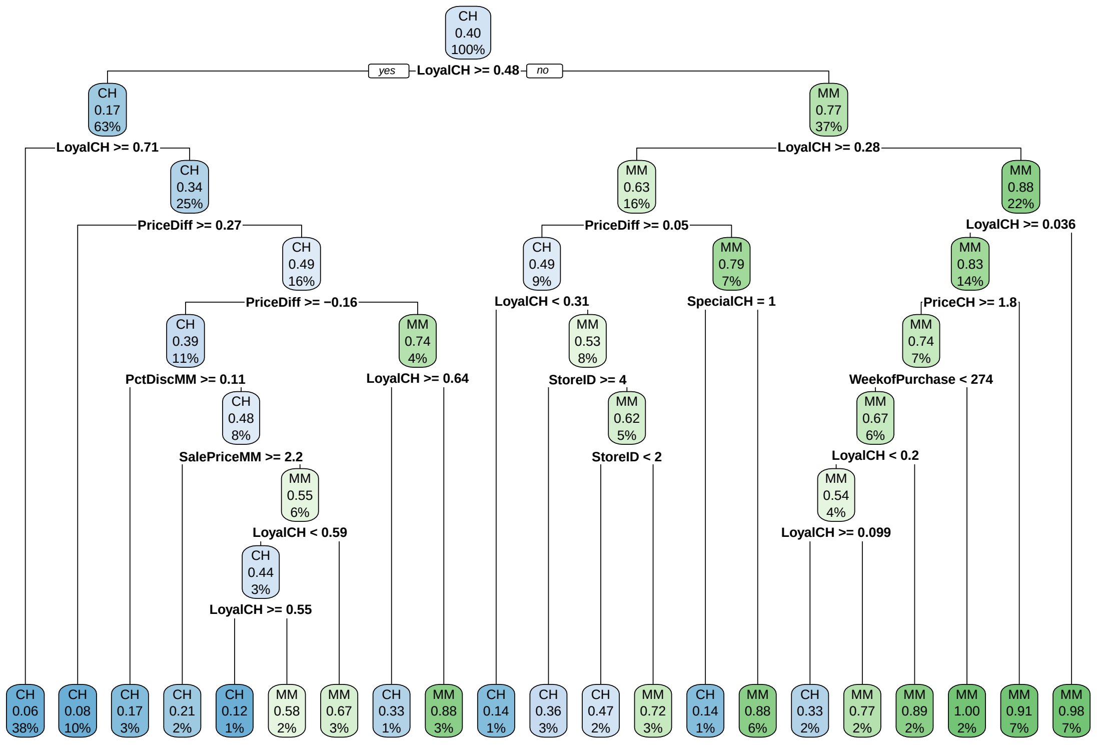
Question 1

Fit the classification tree and use cross-validation to determine the tree size. The plot for optimal tree are shown below.

```r
ctrl <- trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

set.seed(1)
rpart.fit <- train(Purchase~., train,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-8,-6, len = 20))),
                   trControl = ctrl,
                   metric = "ROC")
ggplot(rpart.fit, highlight = TRUE)
```



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



Predict the response on the test data and calculate the test classification error rate.

```
rpart.pred <- predict(rpart.fit, newdata = test)
mean(test$Purchase != rpart.pred)
```

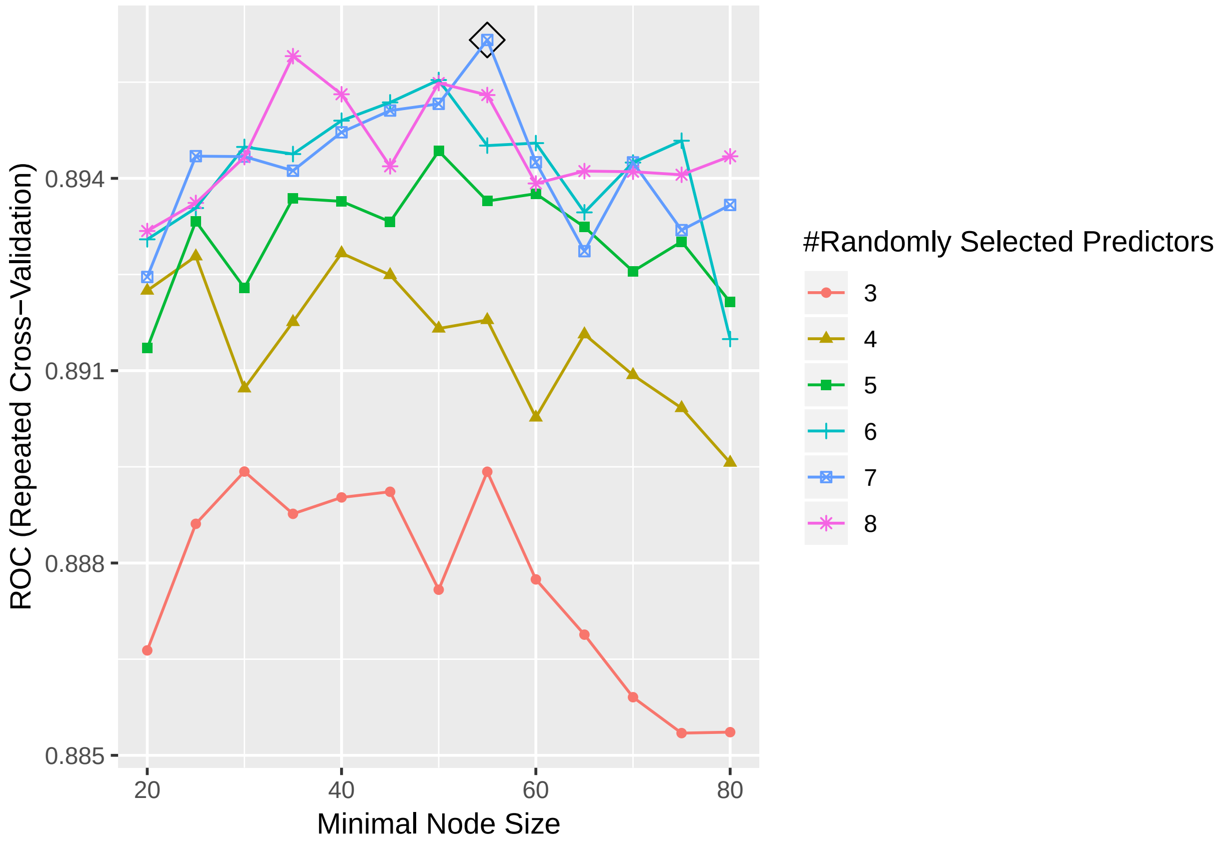
```
## [1] 0.2111111
```

According to the result shown above, the test classification error rate is 21.11%.

Question 2

Fit random forests using training data.

```
rf.grid <- expand.grid(mtry = 3:8,
                       splitrule = "gini",
                       min.node.size = seq(20,80,5))
set.seed(123)
rf.fit <- train(Purchase~., train,
                method = "ranger",
                tuneGrid = rf.grid,
                metric = "ROC",
                trControl = ctrl)

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

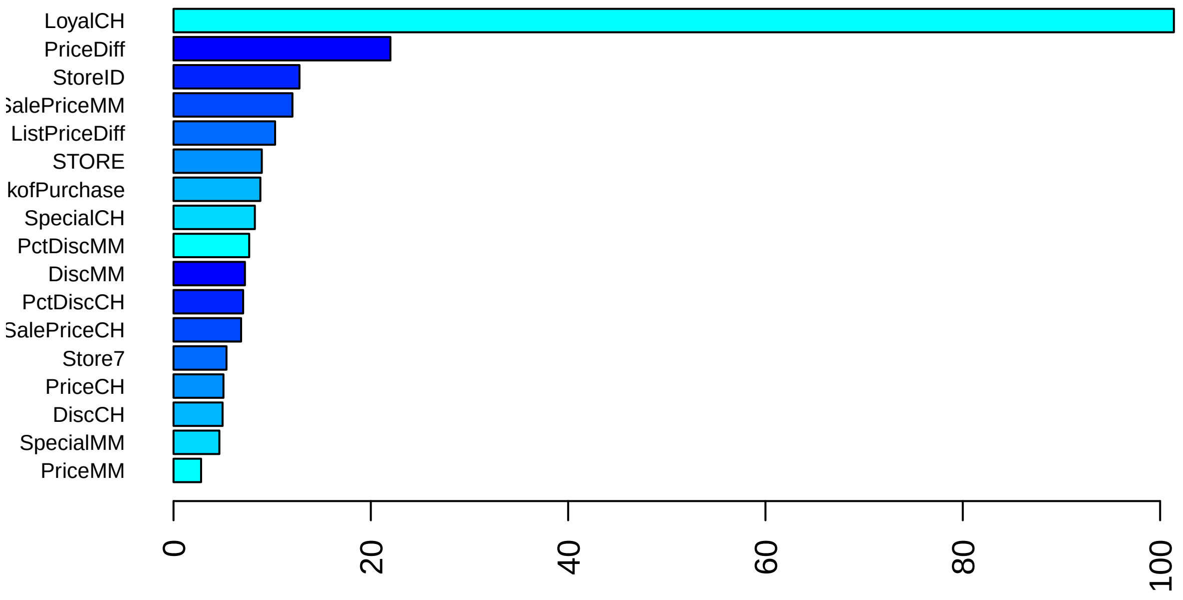


```
rf.fit$finalModel$min.node.size
```

```
## [1] 55
```

Get variable importance

```r
set.seed(123)
rf.final <- ranger(Purchase~., train,
                         mtry = 7,
                         min.node.size = 55,
                         splitrule = "gini",
                         importance = "permutation",
                         scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf.final), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(8))
```



According to the plot above, the importance of each variable are `LoyalCH` > `PriceDiff` > `alePriceMM` > `ListPriceDiff` > `STORE` > `WeekofPurchase` > `SpecialCH` > `PctDiscMM` > `DiscMM` > `PctDiscCH` > `SalePriceCH` > `Store7` > `PriceCH` > `DiscCH` > `SpecialMM` > `PriceMM`.

Predict the response on the test data and calculate the test classification error rate.

```r
rf.pred <- predict(rf.fit, newdata = test)
mean(test$Purchase != rf.pred)
```

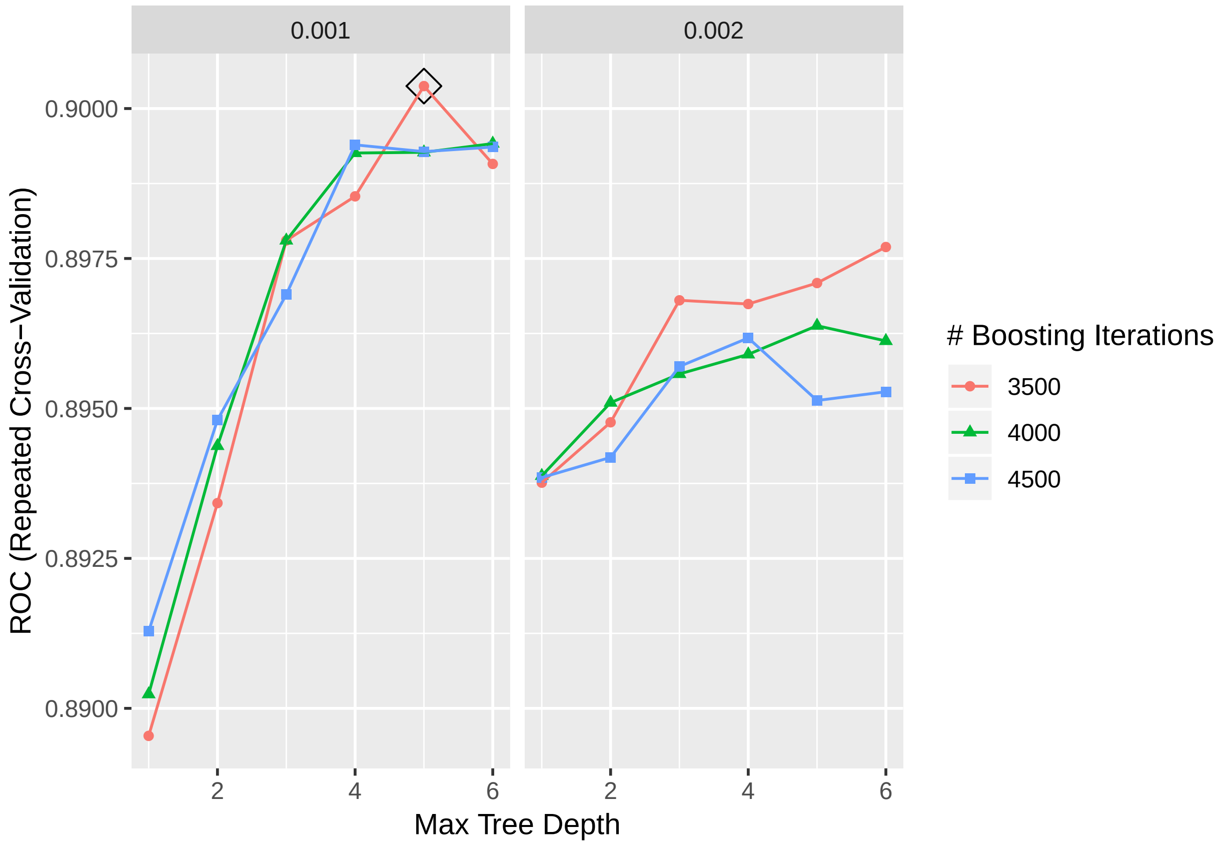
```
## [1] 0.162963
```

According to the result shown above, the test classification error rate is 16.30%.

Question 3

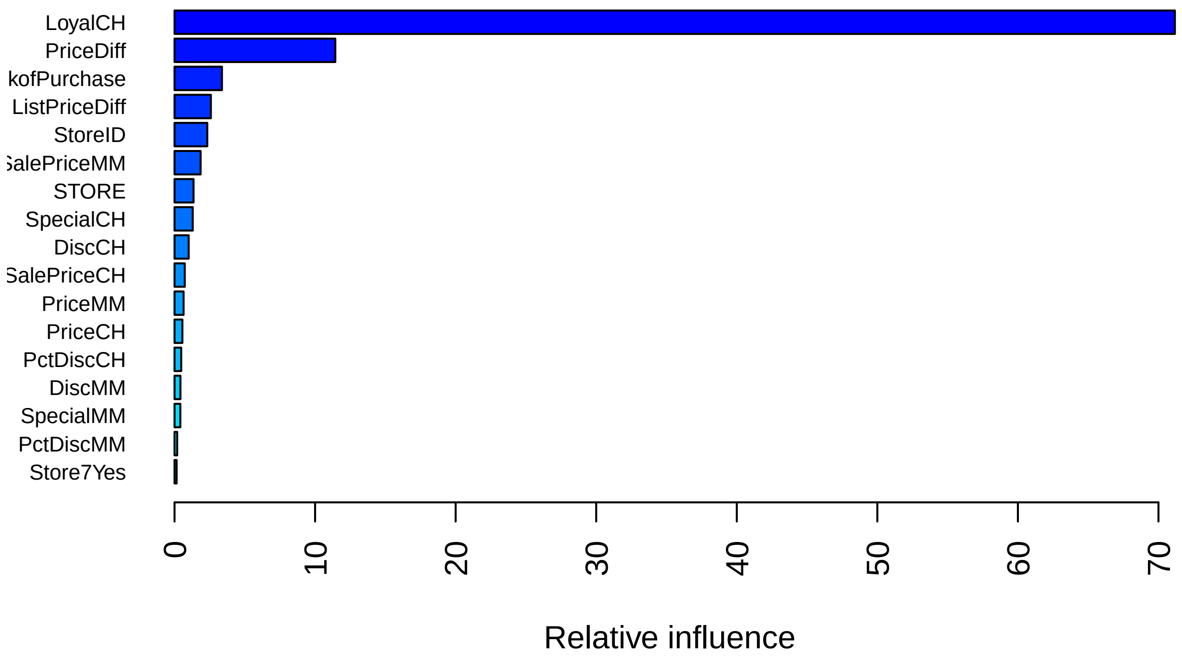
Fit the boosting model.

```r
gbm.grid <- expand.grid(n.trees = c(3500,4000,4500),
                        interaction.depth = 1:6,
                        shrinkage = c(0.001,0.002),
                        n.minobsinnode = 1)
set.seed(123)
# Binomial loss function
gbm.fit <- train(Purchase~., train,
                 tuneGrid = gbm.grid,
                 trControl = ctrl,
                 method = "gbm",
                 distribution = "bernoulli",
                 metric = "ROC",
                 verbose = FALSE)

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



Get variable importance.

```
summary(gbm.fit$finalModel, las = 2, cBars = 19, cex.names = 0.7)
```



```
##                            var    rel.inf
## LoyalCH                 LoyalCH 71.1689896
## PriceDiff             PriceDiff 11.4396835
## WeekofPurchase WeekofPurchase  3.3755505
## ListPriceDiff   ListPriceDiff  2.5863627
## StoreID                 StoreID  2.3307525
## SalePriceMM         SalePriceMM  1.8558136
## STORE                     STORE  1.3527768
## SpecialCH             SpecialCH  1.2954112
## DiscCH                   DiscCH  1.0074954
## SalePriceCH         SalePriceCH  0.7303539
## PriceMM                 PriceMM  0.6444564
## PriceCH                 PriceCH  0.5606354
## PctDiscCH             PctDiscCH  0.4759041
## DiscMM                   DiscMM  0.4212396
## SpecialMM             SpecialMM  0.4131723
## PctDiscMM             PctDiscMM  0.1901652
## Store7Yes             Store7Yes  0.1512371
```

According to the plot above, the importance of each variable are `LoyalCH` > `PriceDiff` > `WeekofPurchase` > `ListPriceDiff` > `StoreID` > `alePriceMM` > `STORE` > `SpecialCH` > `DiscCH` > `SalePriceCH` > `PriceMM` > `PriceCH` > `PctDiscCH` > `DiscMM` > `SpecialMM` > `PctDiscMM` > `Store7Yes`.

```
gbm.pred <- predict(gbm.fit, newdata = test)
mean(test$Purchase != gbm.pred)
```

```
## [1] 0.1481481
```

According to the result shown above, the test classification error rate is 14.81%.