HW4

Amin Yakubu

4/19/2019

```
library(lasso2)
```

```
## R Package to solve regression problems while imposing
##   an L1 constraint on the parameters. Based on S-plus Release 2.1
## Copyright (C) 1998, 1999
## Justin Lokhorst   <jlokhors@stats.adelaide.edu.au>
## Berwin A. Turlach <bturlach@stats.adelaide.edu.au>
## Bill Venables     <wvenable@stats.adelaide.edu.au>
##
## Copyright (C) 2002
## Martin Maechler <maechler@stat.math.ethz.ch>
```

```
library(ISLR)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
library(partykit)
```

```
## Loading required package: libcoin

##
## Attaching package: 'partykit'

## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner,
##     node_surv, node_terminal, varimp
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(ranger)
```

```
##
## Attaching package: 'ranger'

## The following object is masked from 'package:randomForest':
##
##     importance
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```r
library(plotmo)
```

```
## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```r
library(pdp)
library(lime)
```

Question 1a

```r
seed = 1
data("Prostate")
ctrl <- trainControl(method = "cv")
```

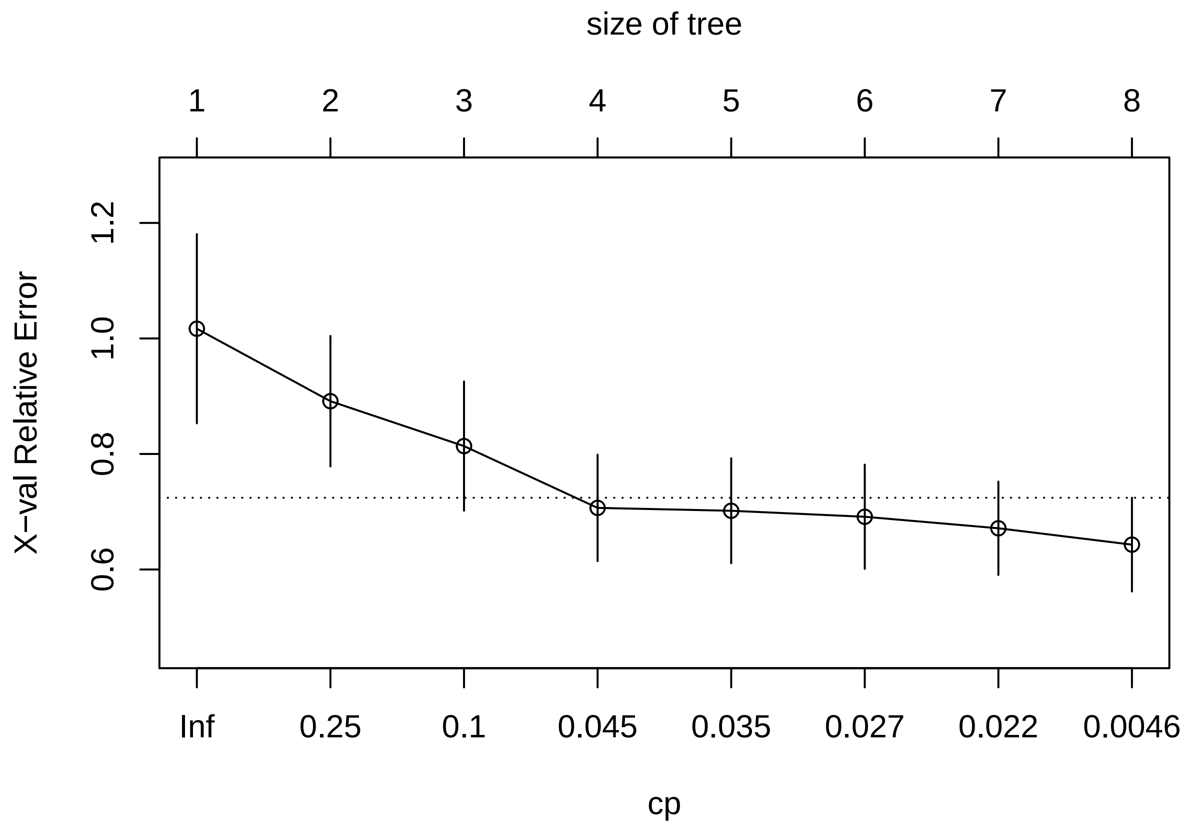
Here, I fit a regression tree with lpsa as the response and the other variables as predictors, and then use cross-validation to determine the optimal tree size. I'll tune over complexity parameter.

```r
set.seed(seed)

tree <- rpart(formula = lpsa ~., data = Prostate,
              control = rpart.control(cp = 0.001))

cpTable <- printcp(tree)
```

```
##
## Regression tree:
## rpart(formula = lpsa ~ ., data = Prostate, control = rpart.control(cp = 0.001))
##
## 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 0.001000      7   0.29653 0.64305 0.081145
```

```r
plotcp(tree)
```



```r
minErr <- which.min(cpTable[,4])
```

Tree size of 8 corresponds to the lowest cross validation error.

Checking tree size which corresponds to the 1SE rule

```r
# Tree size = nsplit + 1
cpTable[cpTable[,4] < cpTable[minErr,4] + cpTable[minErr,5], 2][1] + 1
```

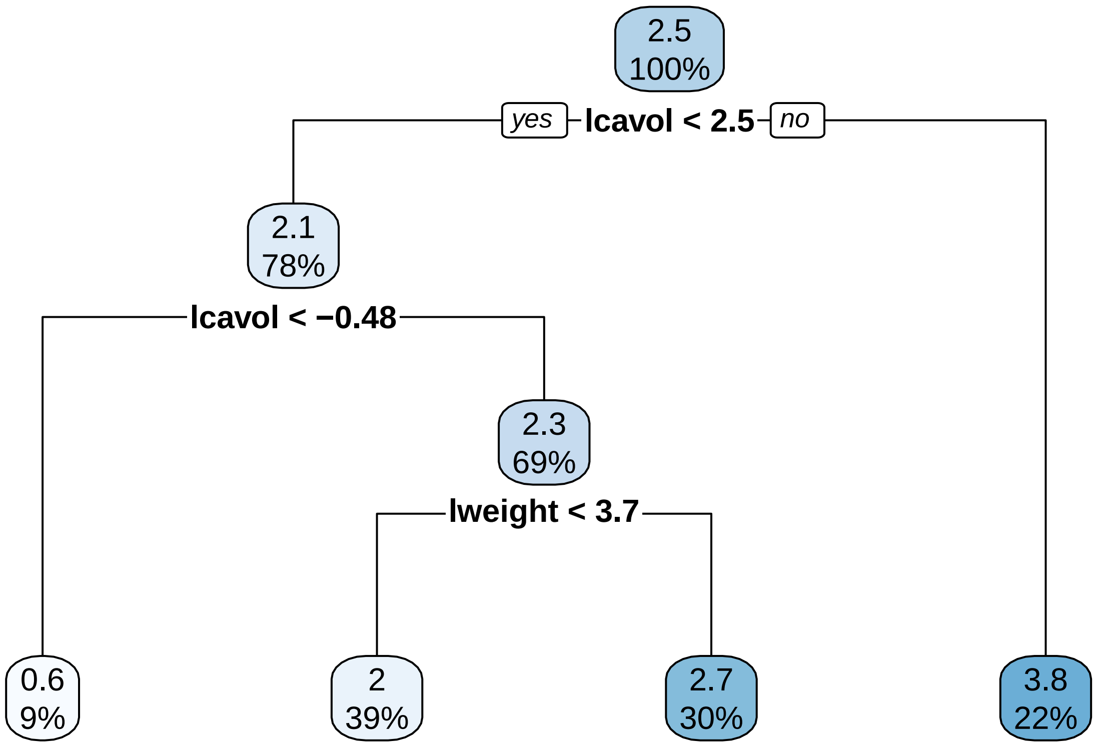
```
## 4
## 4
```

The tree size obtained using the 1 SE rule is tree of size 4. We can also see that from the plot since tree size of 4 is the leftmost value below the horizontal line. The tree corresponding to the lowest cross validation error (size 8) is different from the tree corresponding to the 1 SE rule (size 4)

Question 1b

I'll select and prune my tree using the 1 SE

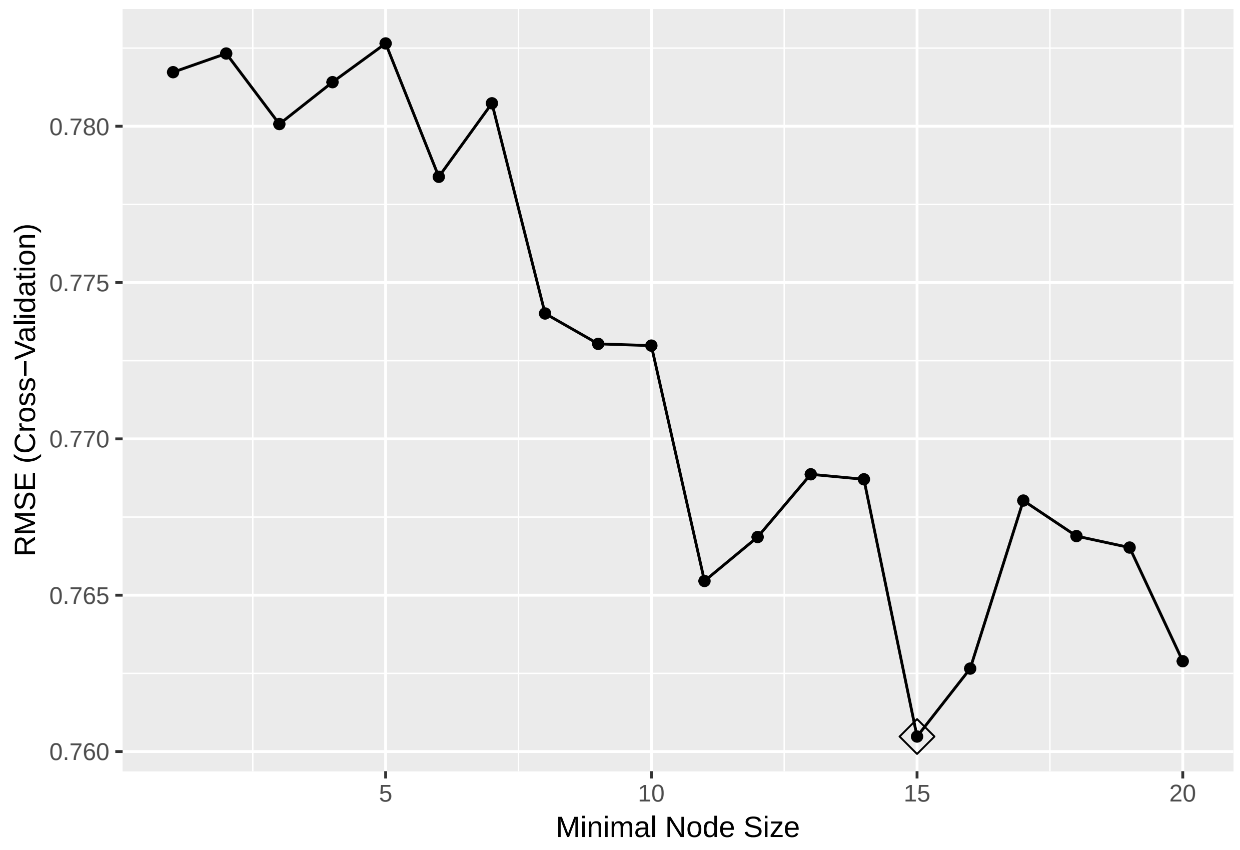
```
selected_tree = prune(tree, cp = cpTable[cpTable[,4] < cpTable[minErr,4] + cpTable[minErr,5], 1][1])
rpart.plot(selected_tree)
```



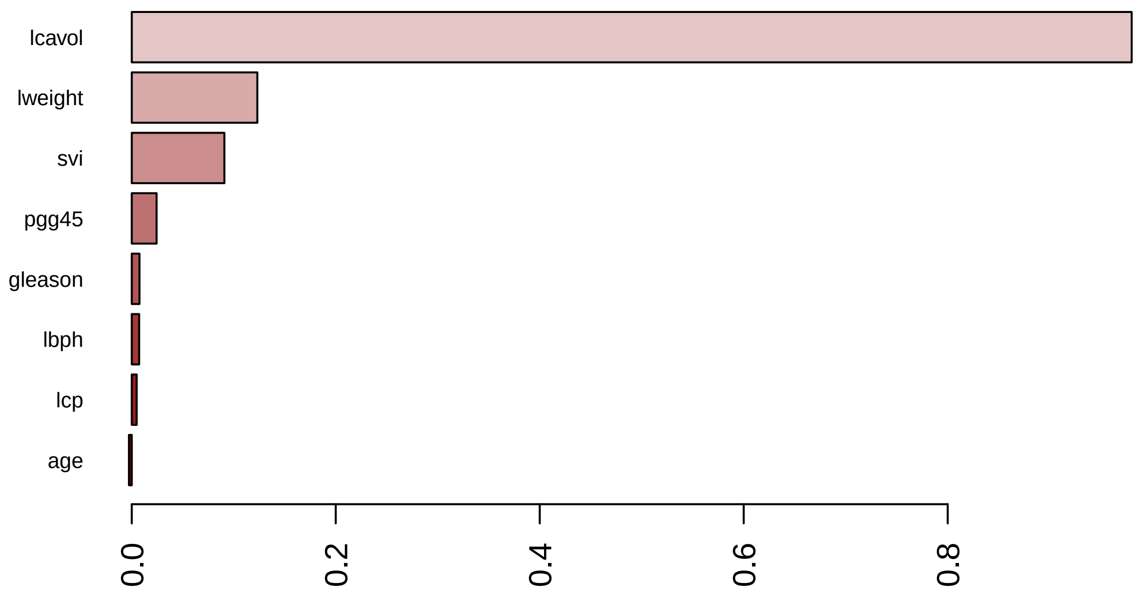
The mean `lpsa` for observations with less than 2.5 of `lcavol` and further less than -0.48 of `lcavol` is 0.6. 9% of the total observations are in this terminal node.

Question 1c - Bagging

```
bagging.grid <- expand.grid(mtry = 8,
                            splitrule = "variance",
                            min.node.size = 1:20)
set.seed(seed)
bagging <- train(lpsa~., Prostate,
                 method = "ranger",
                 tuneGrid = bagging.grid,
                 trControl = ctrl,
                 importance = 'permutation')

ggplot(bagging, highlight = TRUE)
```



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



```
# bagging$results[which.min(bagging$results[,5]),]
```

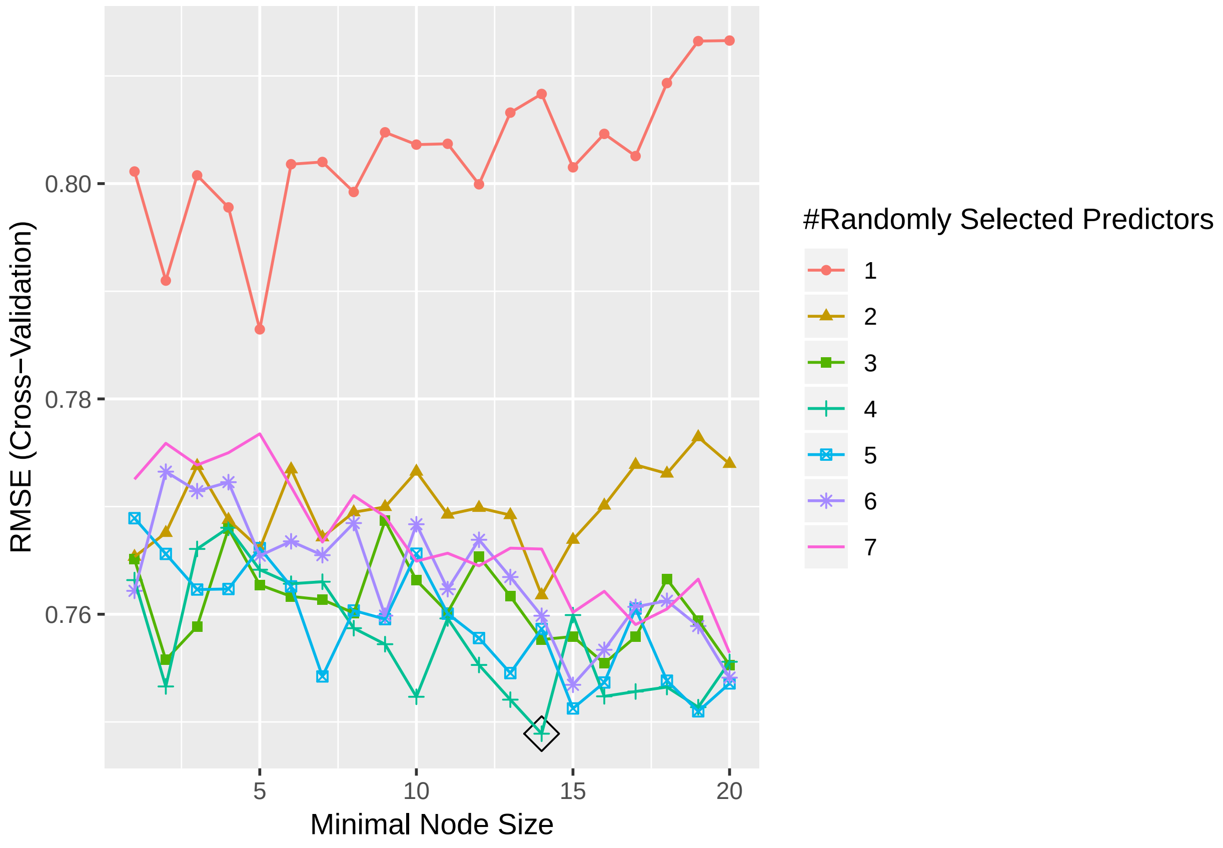
From the variable important plot, we see that the 3 most important variables are `lcavol`, `lweight` and `svi`.

Question 1d - Random Forest

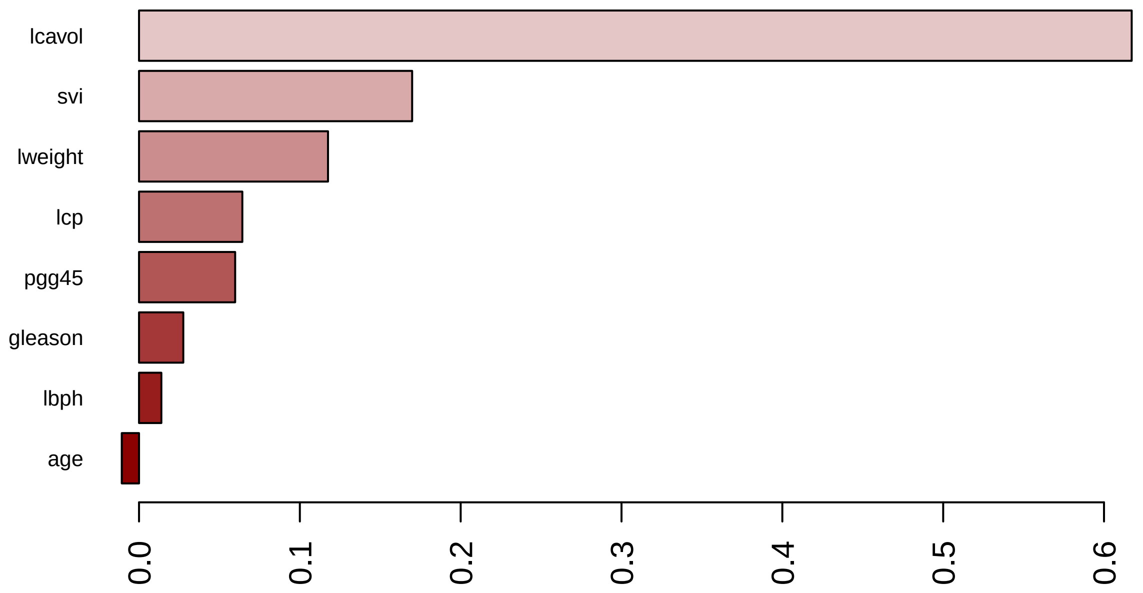
I'll randomly select between 1 to 7 of the variables for each split to see which number works best.

```
rf.grid = expand.grid(mtry = 1:7,
                      splitrule = "variance",
                      min.node.size = 1:20)
set.seed(seed)
rf.fit = train(lpsa~., Prostate,
               method = "ranger",
               tuneGrid = rf.grid,
               trControl = ctrl,
               importance = 'permutation')

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

```
## Warning: The shape palette can deal with a maximum of 6 discrete values
## because more than 6 becomes difficult to discriminate; you have 7.
## Consider specifying shapes manually if you must have them.

## Warning: Removed 20 rows containing missing values (geom_point).
```



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

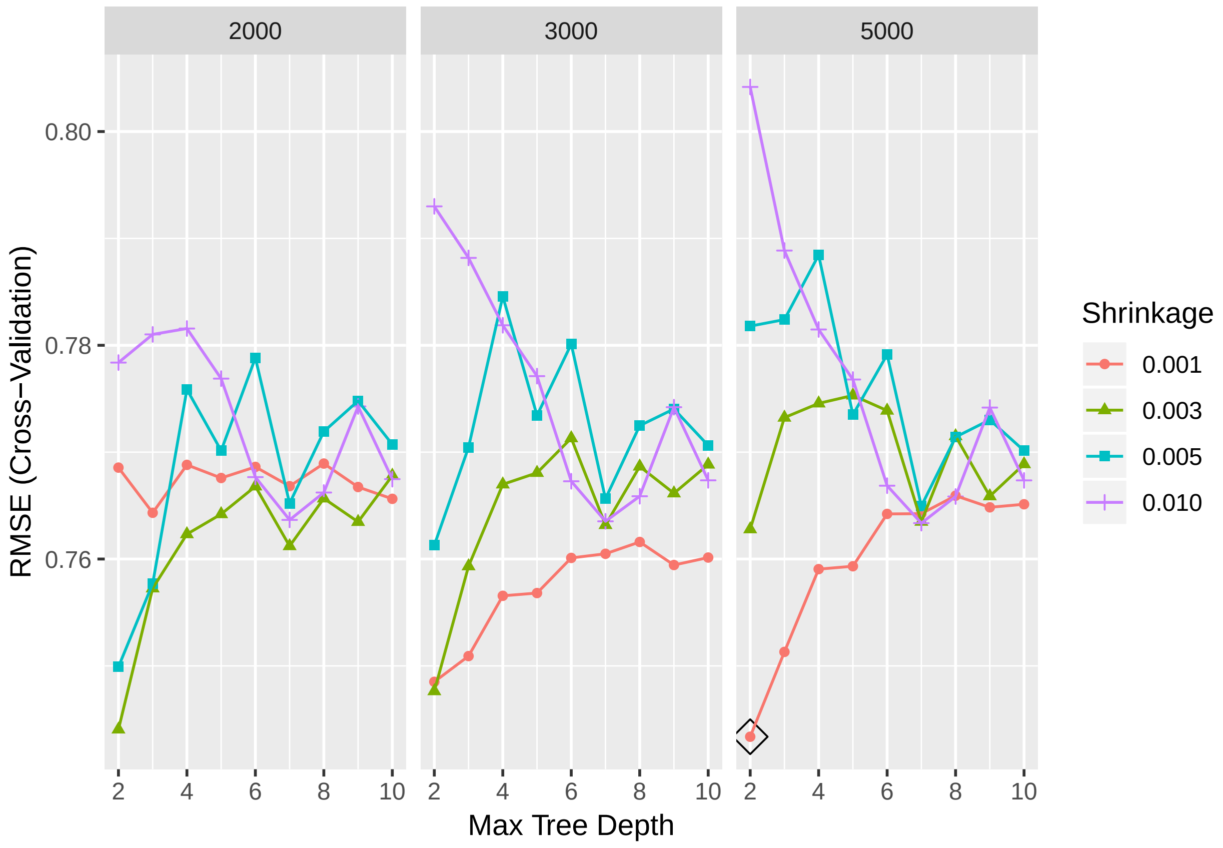


```r
# rf.fit$results[which.min(rf.fit$results[,5]),]
```

The best model used has `mtry` of 4 and `min.node.size` of 14 The result of the random forest shows that the most important variables are `lcavol`, `svi` and `lweight`. In this case `svi` is more important that `lweight`.

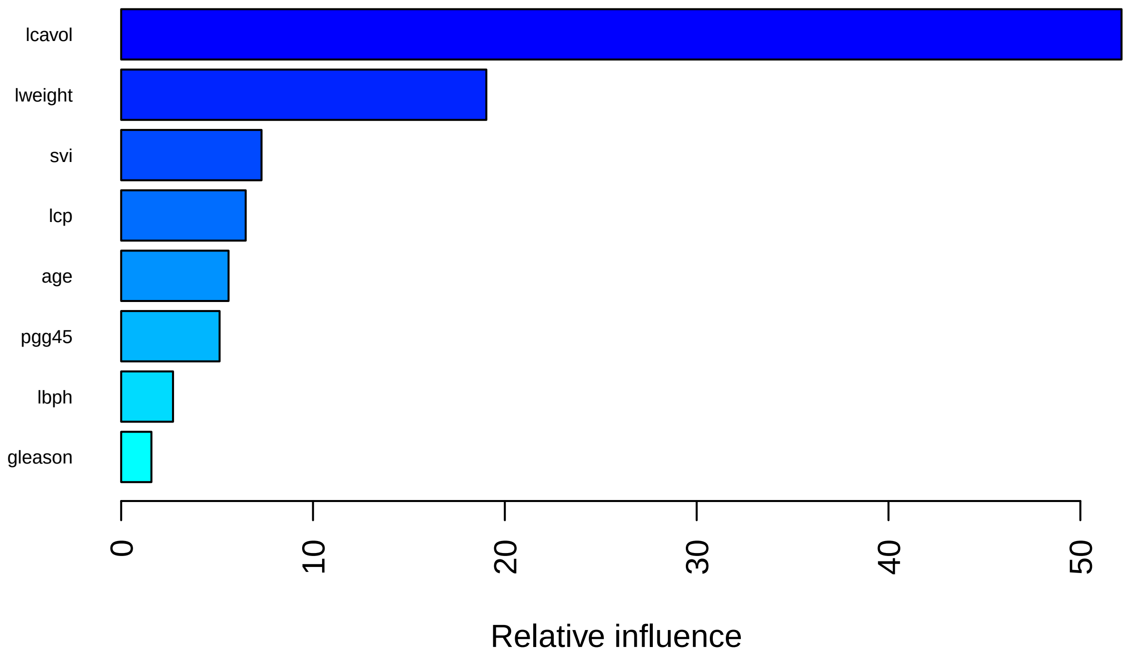
Question 1e

```r
gbm.grid = expand.grid(n.trees = c(2000,3000, 5000),
                        interaction.depth = 2:10,
                        shrinkage = c(0.01, 0.001,0.003,0.005),
                        n.minobsinnode = 1)
set.seed(seed)
gbm.fit = train(lpsa ~., Prostate,
                 method = "gbm",
                 tuneGrid = gbm.grid,
                 verbose = FALSE,
                 trControl = ctrl)

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



The selected final model has 5000 trees with a depth of 2 with a learning rate (shrinkage) of 0.001.

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



```
##              var   rel.inf
## lcavol    lcavol 52.147135
## lweight lweight 19.035587
## svi         svi  7.316331
## lcp         lcp  6.491809
## age         age  5.597326
## pgg45     pgg45  5.130606
## lbph       lbph  2.703727
## gleason gleason  1.577479
```

```
# gbm.fit$results[which.min(gbm.fit$results[,5]),]
```

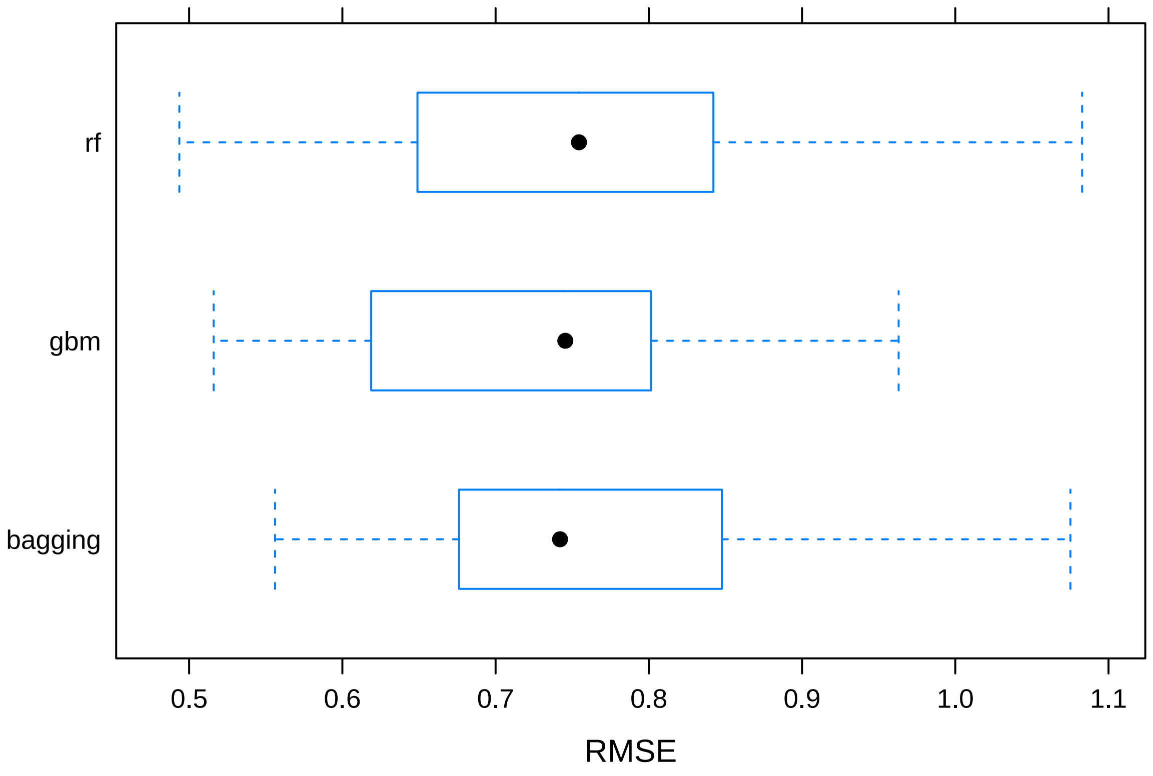
The most important variables using boosting are `lcavol`, `lweight` and `svi`. The order is similar to the bagging method.

Question 1f

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

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: rf, gbm, bagging
## Number of resamples: 10
##
## MAE
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf      0.4388518 0.5752939 0.5988010 0.6240935 0.6357321 0.9806087    0
## gbm     0.4191029 0.5282535 0.5641767 0.6076525 0.6528855 0.8839551    0
## bagging 0.4744007 0.5785223 0.5947370 0.6325892 0.6277988 0.9632561    0
##
## RMSE
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf      0.4936100 0.6643460 0.7544224 0.7489068 0.8336855 1.0827494    0
## gbm     0.5159758 0.6401619 0.7454831 0.7433731 0.8013531 0.9630145    0
## bagging 0.5560895 0.6843114 0.7420510 0.7604800 0.8229470 1.0751279    0
##
## Rsquared
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf      0.2612565 0.6004245 0.6769932 0.6350320 0.7105999 0.7609664    0
## gbm     0.3849614 0.5831853 0.6522304 0.6357958 0.7192895 0.7868307    0
## bagging 0.2115985 0.5719590 0.6745214 0.6139633 0.7080146 0.7496108    0
```

```
bwplot(resamp, metric = "RMSE")
```



I'll choose the regression tree method to predict PSA level. This is because we see that the cross validation is smallest for the bagging method (both using the 1SE rule and the mininum cross validation error). Also, in addition to having the smallest cross validation error it is easier to explain (more interpretable).

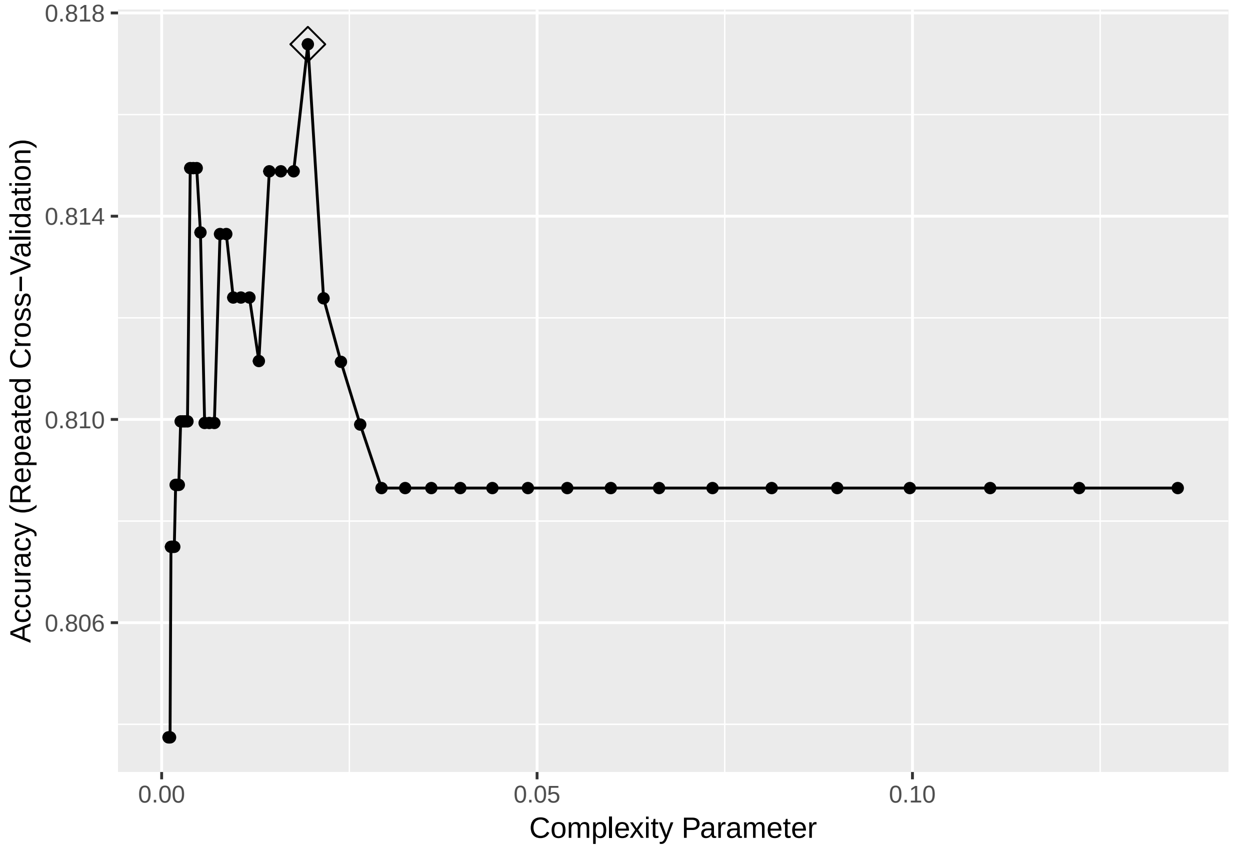
Question 2a - Classification Tree

```r
data("OJ")
```

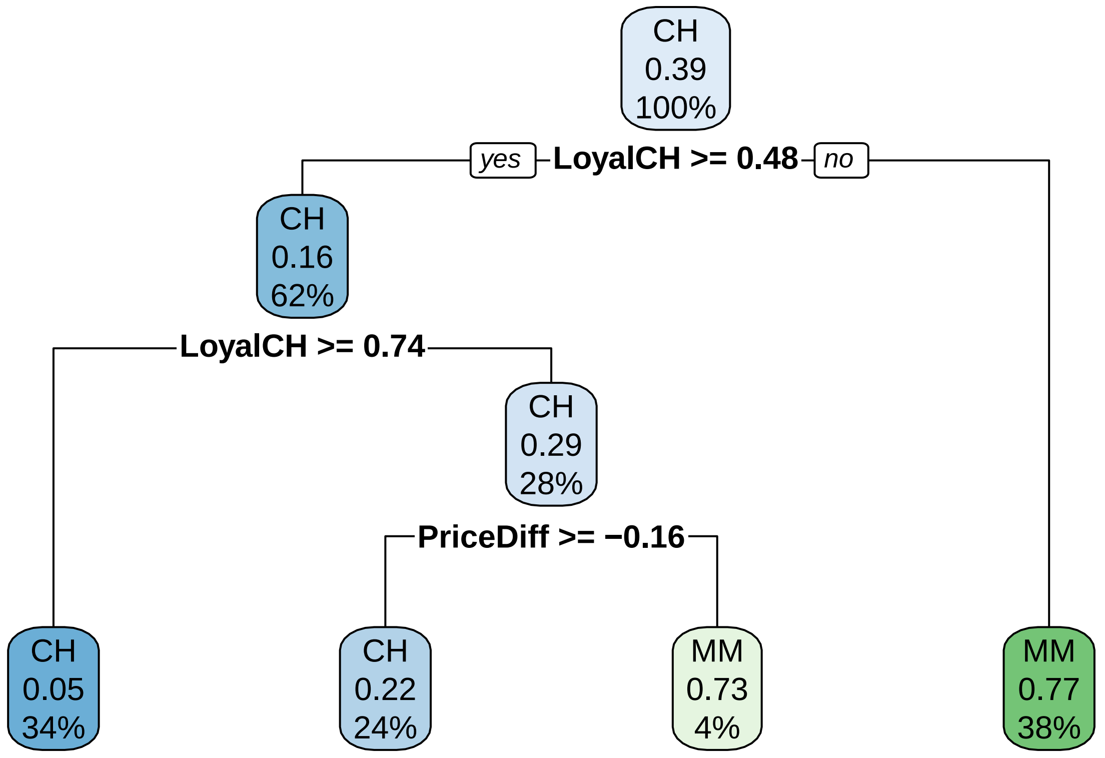
```r
set.seed(seed)
rowTrain = createDataPartition(y = OJ$Purchase,
                               p = 0.747,
                               list = FALSE)

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

Since we are interested in missclassification rate, I'll use accuracy as the metric in the cross validation to select the model.

```r
set.seed(seed)
rpart.class <- train(Purchase ~., OJ,
                   subset = rowTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-7,-2, len = 50))),
                   trControl = ctrl,
                   metric = "Accuracy")

ggplot(rpart.class, highlight = T)
```



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



```
rpart.class$bestTune
```

```
##            cp
## 31 0.01947204
```

The best tree size is 4 (number of splits + 1) which corresponds to a complexity parameter of 0.01947204.

Now let's use the model to predict the test data.

```
rpart.pred <- predict(rpart.class, newdata = OJ[-rowTrain,])

confusionMatrix(rpart.pred,
                reference = OJ$Purchase[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 138  23
##         MM  27  82
##
##                Accuracy : 0.8148
##                  95% CI : (0.7633, 0.8593)
##     No Information Rate : 0.6111
##     P-Value [Acc > NIR] : 4.049e-13
##
##                   Kappa : 0.6131
##  Mcnemar's Test P-Value : 0.6714
##
##             Sensitivity : 0.8364
##             Specificity : 0.7810
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.7523
##              Prevalence : 0.6111
##          Detection Rate : 0.5111
##    Detection Prevalence : 0.5963
##       Balanced Accuracy : 0.8087
##
##        'Positive' Class : CH
##
```

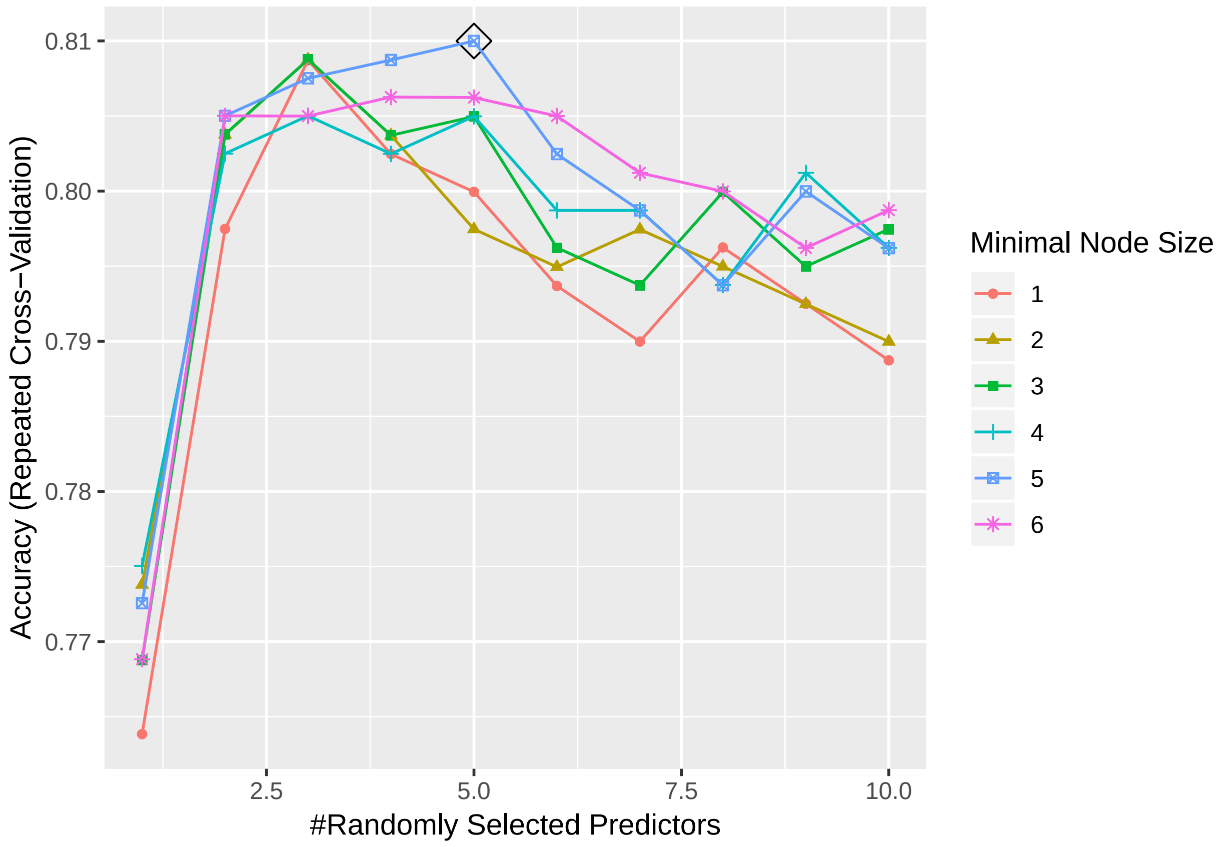
```
# Error rate
error_rate = mean(rpart.pred != OJ$Purchase[-rowTrain]) * 100

cat(c("The error rate for the classification tree is", error_rate, '%'))
```

```
## The error rate for the classification tree is 18.5185185185185 %
```

Question 2b - Random Forest

```
rf.grid <- expand.grid(mtry = 1:10,
                       splitrule = "gini",
                       min.node.size = 1:6)
set.seed(seed)
```

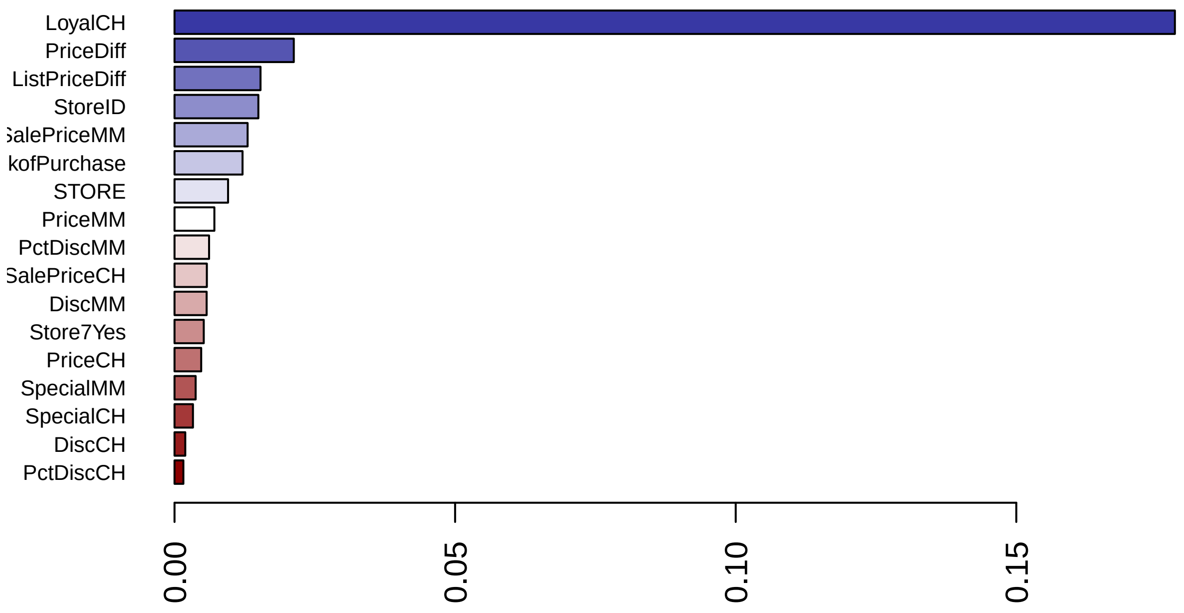
```
rf.class <- train(Purchase ~., OJ,
                  subset = rowTrain,
                  method = "ranger",
                  tuneGrid = rf.grid,
                  metric = "Accuracy",
                  trControl = ctrl,
                  importance = 'permutation')

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



The selected model using cross validation has `mtry` of 5 `and` `min.node.size` of 5 as well.

Variable Importance

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



From the variable important plot, we see that the variable `LoyalCH` is the most important.

Now let's use the random forest model selected using cross validation to predict the outcome for the test data

```r
rf.pred = predict(rf.class, newdata = OJ[-rowTrain,])

confusionMatrix(rf.pred,
                reference = OJ$Purchase[-rowTrain])
```

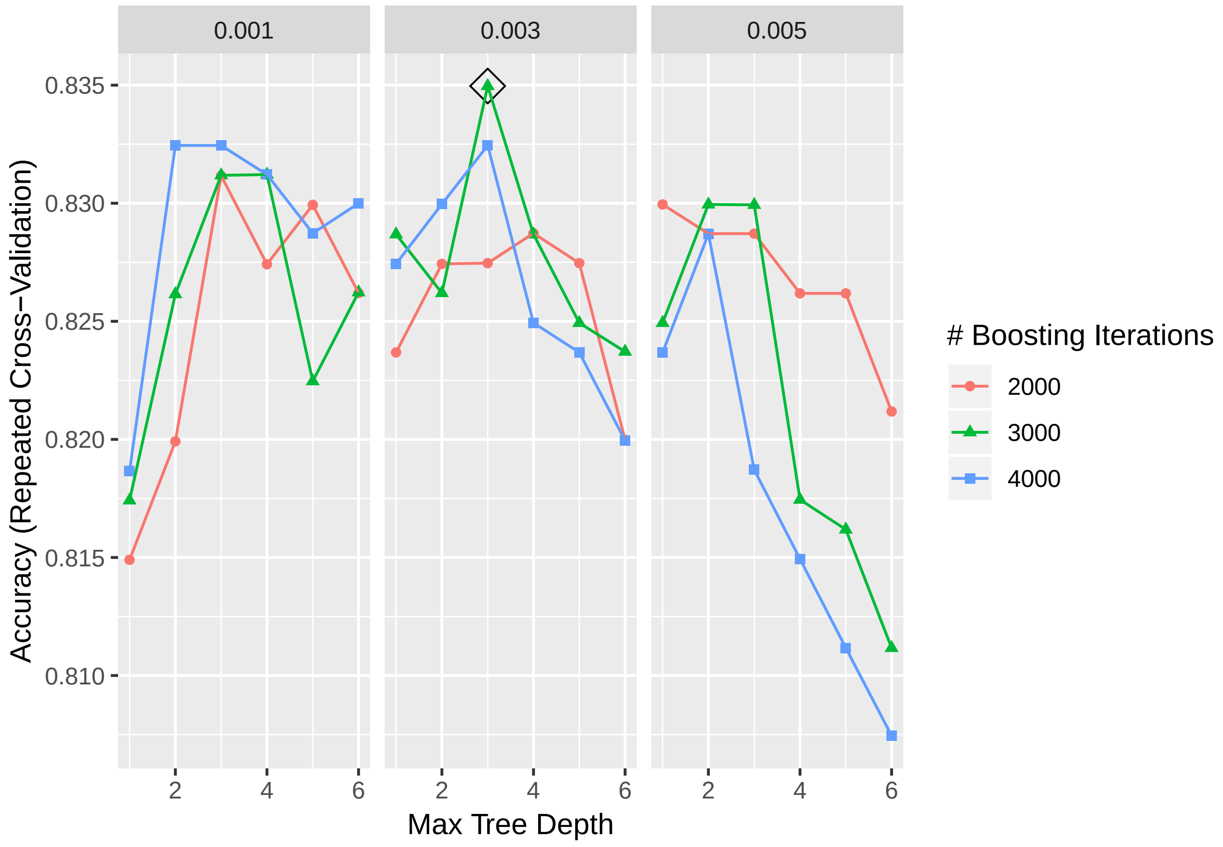
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 143  30
##         MM  22  75
##
##                Accuracy : 0.8074
##                  95% CI : (0.7552, 0.8527)
##     No Information Rate : 0.6111
##     P-Value [Acc > NIR] : 3.059e-12
##
##                   Kappa : 0.5891
##  Mcnemar's Test P-Value : 0.3317
##
##             Sensitivity : 0.8667
##             Specificity : 0.7143
##          Pos Pred Value : 0.8266
##          Neg Pred Value : 0.7732
##              Prevalence : 0.6111
##          Detection Rate : 0.5296
##    Detection Prevalence : 0.6407
##       Balanced Accuracy : 0.7905
##
##        'Positive' Class : CH
##
```

```
# Error rate
error_rate = mean(rf.pred != OJ$Purchase[-rowTrain]) * 100

cat(c("The error rate for the random forest model is", error_rate, '%'))
```

```
## The error rate for the random forest model is 19.2592592592593 %
```

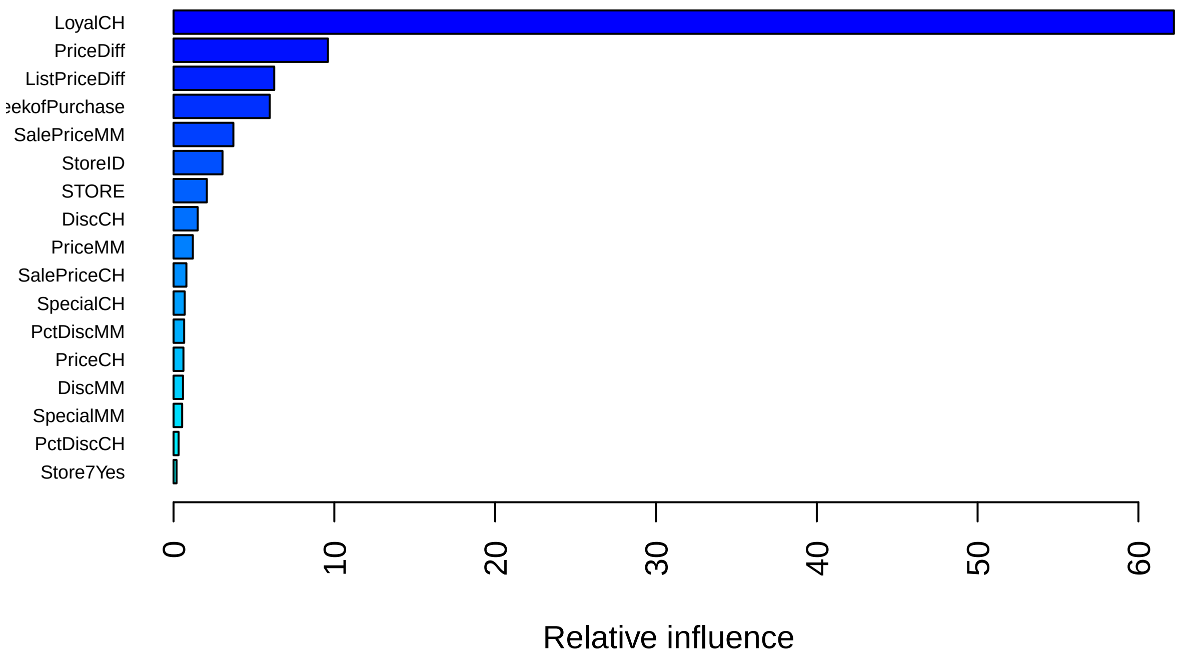
Question 3c - Boosting

```
gbmB.grid <- expand.grid(n.trees = c(2000,3000,4000),
                         interaction.depth = 1:6,
                         shrinkage = c(0.001,0.003,0.005),
                         n.minobsinnode = 1)
set.seed(seed)
gbmB.fit <- train(Purchase ~., OJ,
                  subset = rowTrain,
                  tuneGrid = gbmB.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "Accuracy",
                  verbose = FALSE)

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



The selected model had a maximum tree depth of 3 and learning rate of 0.003 with 3000 boosting iterations.

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



```
##                           var    rel.inf
## LoyalCH               LoyalCH 62.2056179
## PriceDiff           PriceDiff  9.6005501
## ListPriceDiff   ListPriceDiff  6.2640002
## WeekofPurchase WeekofPurchase  5.9799203
## SalePriceMM       SalePriceMM  3.7228741
## StoreID               StoreID  3.0450950
## STORE                   STORE  2.0699390
## DiscCH                 DiscCH  1.5026626
## PriceMM               PriceMM  1.2015768
## SalePriceCH       SalePriceCH  0.8010728
## SpecialCH           SpecialCH  0.6952829
## PctDiscMM           PctDiscMM  0.6633311
## PriceCH               PriceCH  0.6174462
## DiscMM                 DiscMM  0.5890194
## SpecialMM           SpecialMM  0.5397378
## PctDiscCH           PctDiscCH  0.3136858
## Store7Yes           Store7Yes  0.1881880
```

The variable importance is very similar that of random forest in that the most important variable is `LoyalCH`.

Next, I'll predict and compute the missclassification rate of the boosted model.

```r
gbm.pred = predict(gbmB.fit, newdata = OJ[-rowTrain,])

confusionMatrix(gbm.pred,
                reference = OJ$Purchase[-rowTrain])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CH  MM
##         CH 147  24
##         MM  18  81
##
##                Accuracy : 0.8444
##                  95% CI : (0.7956, 0.8855)
##     No Information Rate : 0.6111
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6693
##  Mcnemar's Test P-Value : 0.4404
##
##             Sensitivity : 0.8909
##             Specificity : 0.7714
##          Pos Pred Value : 0.8596
##          Neg Pred Value : 0.8182
##              Prevalence : 0.6111
##          Detection Rate : 0.5444
##    Detection Prevalence : 0.6333
##       Balanced Accuracy : 0.8312
##
##        'Positive' Class : CH
##
```

```
# Error rate
error_rate = mean(gbm.pred != OJ$Purchase[-rowTrain]) * 100

cat(c("The error rate for the GBM model is", error_rate,'%'))
```

```
## The error rate for the GBM model is 15.5555555555556 %
```

The missclassification rate for the GBM model (15.55%) is much better than the classification tree (18.518%) and the random forest (19.2592%).