# p8106_hw4

*David DeStephano*

*April 25, 2020*

```
library(caret) # only for plot
```

```
## Warning: package 'caret' was built under R version 3.6.2
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(lasso2) # only for data
```

```
## 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(tidyverse)
```

```
## -- Attaching packages ------------------------- tidyverse 1.2.1 --
```

```
## v tibble  2.1.3     v purrr   0.3.3
## v tidyr   1.0.0     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v tibble  2.1.3     v forcats 0.4.0
```

```
## -- Conflicts ---------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.6.2
```

```
library(rpart) #cart
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.3
```

```
library(party)
```

```
## Warning: package 'party' was built under R version 3.6.3
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Warning: package 'mvtnorm' was built under R version 3.6.2
```

```
## Loading required package: modeltools
```

```
## Warning: package 'modeltools' was built under R version 3.6.3
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Warning: package 'strucchange' was built under R version 3.6.3
```

```
## Loading required package: zoo
```

```
## 
## Attaching package: 'zoo'
```

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

```
## Loading required package: sandwich
```

```
## Warning: package 'sandwich' was built under R version 3.6.3
```

```
## 
## Attaching package: 'strucchange'
```

```
## The following object is masked from 'package:stringr':
## 
##     boundary
```

```
library(partykit)
```

```
## Warning: package 'partykit' was built under R version 3.6.3
```

```
## Loading required package: libcoin
```

```
## Warning: package 'libcoin' was built under R version 3.6.3
```

```
##
## 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)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

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

```r
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 3.6.3
```

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

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

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.6.3
```

```
## Loaded gbm 2.1.5
```

```r
library(plotmo)
```

```
## Warning: package 'plotmo' was built under R version 3.6.3
## Loading required package: Formula
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```
## Warning: package 'TeachingDemos' was built under R version 3.6.2
```

```r
library(pdp)
```

```
## Warning: package 'pdp' was built under R version 3.6.3
```

```
##
## Attaching package: 'pdp'
```

```
## The following object is masked from 'package:purrr':
##
##     partial
```

```r
library(lime)
```

```
## Warning: package 'lime' was built under R version 3.6.3
```

```
## This version of Shiny is designed to work with 'htmlwidgets' >= 1.5.
##     Please upgrade via install.packages('htmlwidgets').
```

```
##
## Attaching package: 'lime'
```

```
## The following object is masked from 'package:dplyr':
##
##     explain
```

```r
library(ModelMetrics)
```

```
## Warning: package 'ModelMetrics' was built under R version 3.6.2
```

```
##
## Attaching package: 'ModelMetrics'
```

```
## The following objects are masked from 'package:caret':
##
##     confusionMatrix, precision, recall, sensitivity, specificity
```

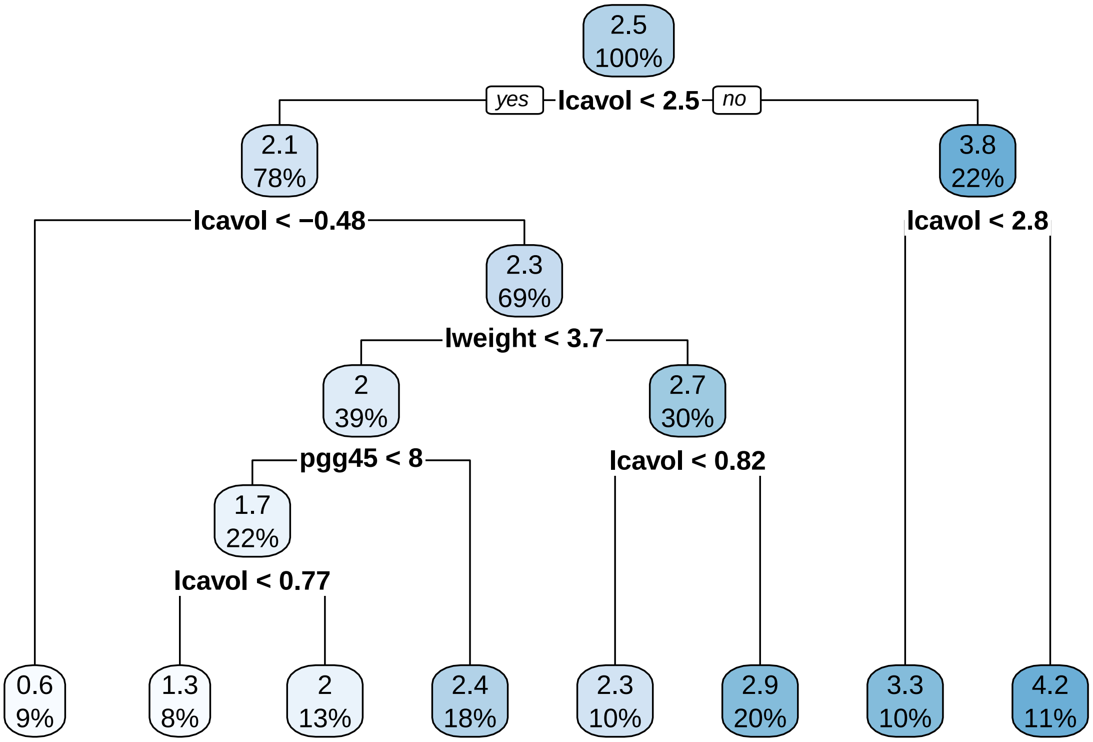
```
## The following object is masked from 'package:base':
##
##     kappa
```

# Question 1

```
data(Prostate)
```

### Part A: Fitting a regression tree to prostate data

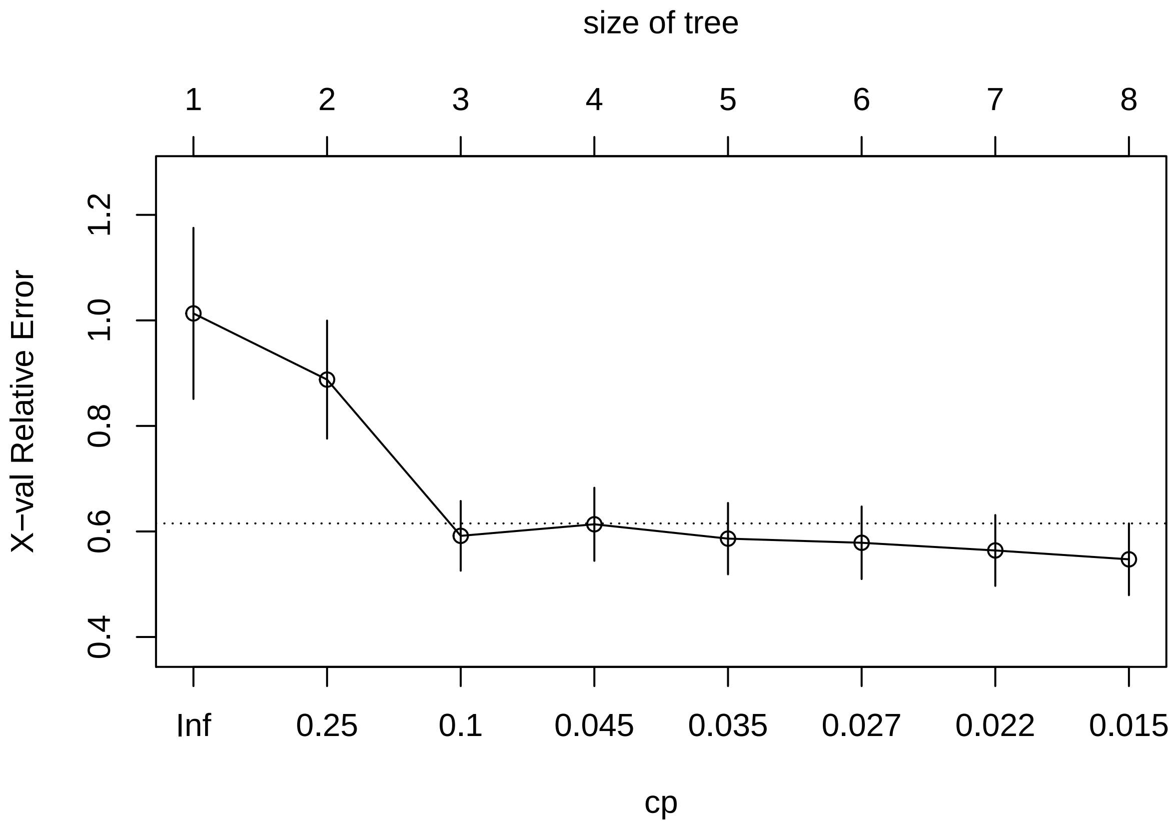
```
set.seed(1)
tree1 <-rpart(formula = lpsa~., data = Prostate)
rpart.plot(tree1)
```



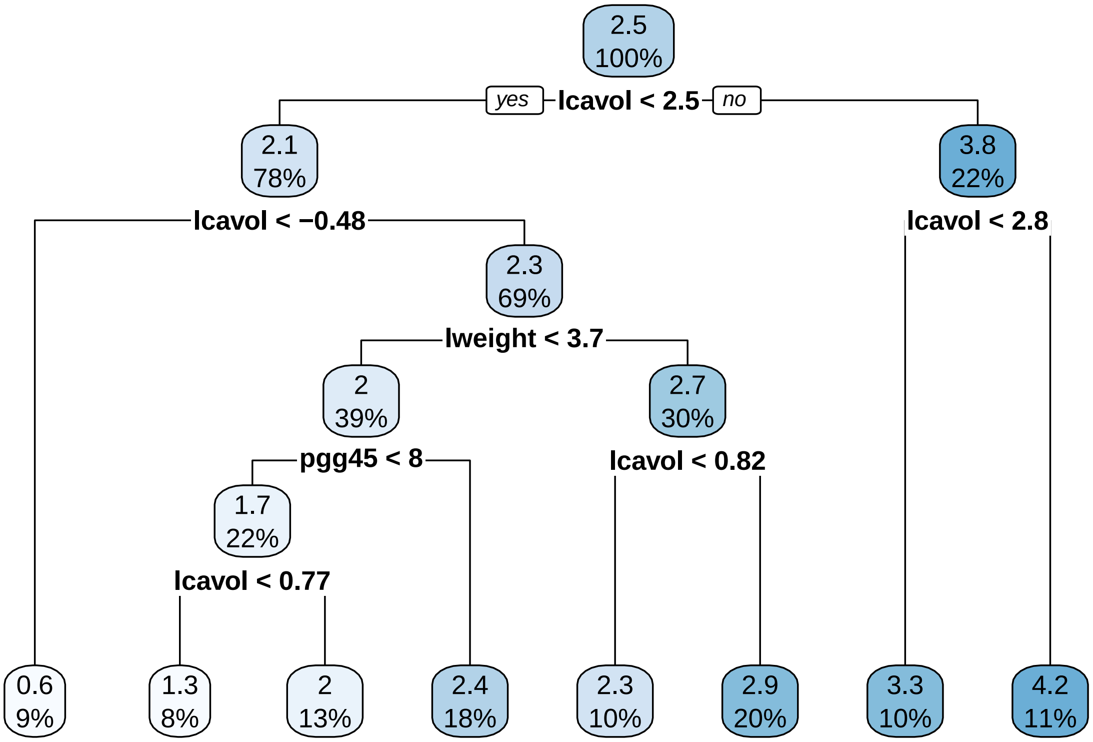
```
cpTable <-printcp(tree1)
```

```
##
## 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 1.01323 0.162162
## 2 0.184647      1   0.65289 0.88779 0.111915
## 3 0.059316      2   0.46824 0.59168 0.066102
## 4 0.034756      3   0.40893 0.61359 0.069269
## 5 0.034609      4   0.37417 0.58640 0.067630
## 6 0.021564      5   0.33956 0.57853 0.068772
## 7 0.021470      6   0.31800 0.56398 0.067155
## 8 0.010000      7   0.29653 0.54721 0.068034
```

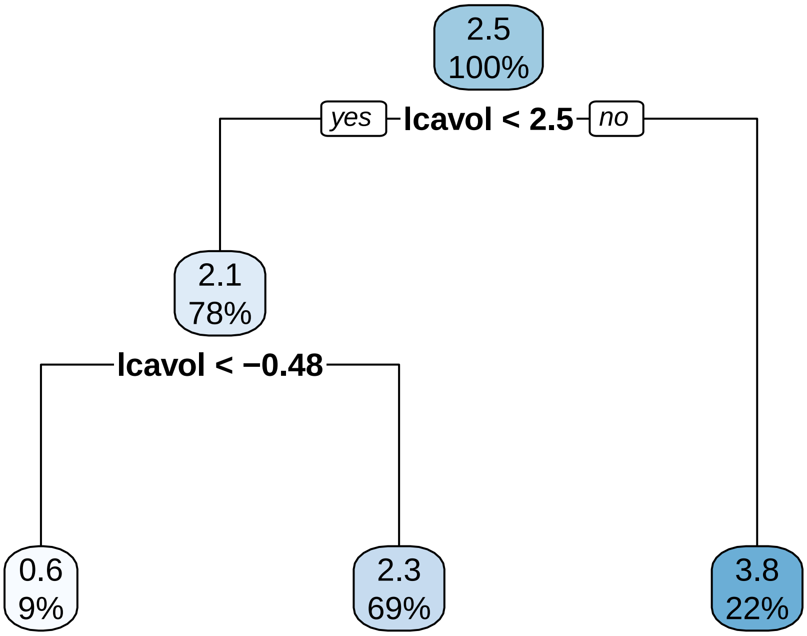
```
plotcp(tree1)
```



```
minErr <-which.min(cpTable[,4])
#minimum cross-validation error
min_tree <-prune(tree1, cp = cpTable[minErr,1])
rpart.plot(min_tree)
```



```
# 1SE rule
se_tree <-prune(tree1, cp =
                 cpTable[cpTable[,4]<cpTable[minErr,4]+cpTable[minErr,5],1][1])


rpart.plot(se_tree)
```

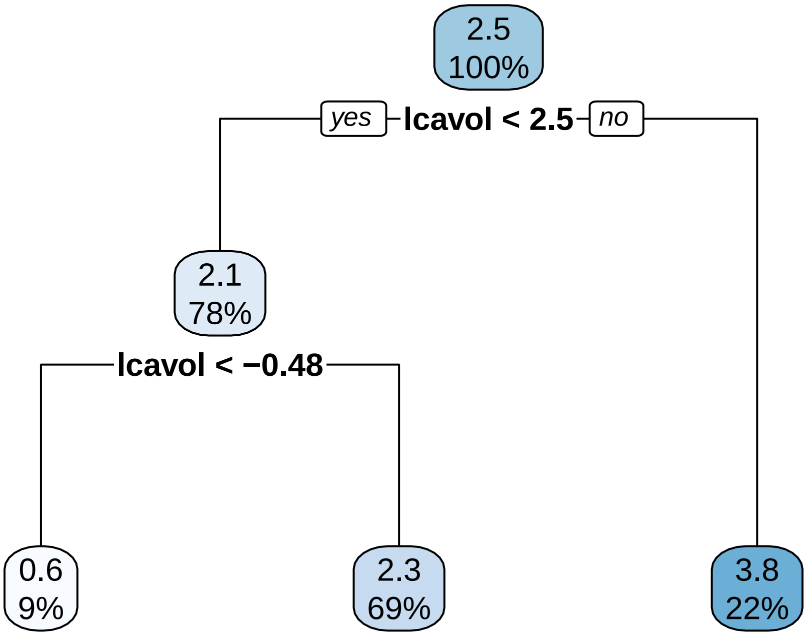


The tree size that corresponds to the lowest cross-validation error is 8, using the 1 SE error rule the tree size is 3.

## Part B: plot the final tree

When looking at the plotcp, the left most value where the mean is below the horizontal line is 3. This corresponds to the pruned tree using the 1 SE rule,so this is the tree that will be used. This model suggests that lcavol is the only predictor of importance when predicting lpsa and keeping at a reasonable error rate.

```r
rpart.plot(se_tree)
```



Interpretation of terminal node 1: When lcval is less than -.48, the mean lpsa is 2.3. This node contains 9% of the sample.

## Part C Bagging

```r
#Create training and test data
set.seed(1)
train = sample(1:nrow(Prostate), nrow(Prostate)/2)
#Use all variables for mtry
bagging <-randomForest(lpsa~., Prostate, subset=train,
                       mtry = 8, importance=TRUE)

bagging$importance
```

```
##              %IncMSE IncNodePurity
## lcavol    0.806039041    38.8769717
## lweight   0.088681017     6.2988494
## age      -0.043813386     2.7824295
## lbph      0.009322002     1.2658944
## svi       0.074341780     4.0489805
## lcp       0.003032361     2.8294784
## gleason   0.009627956     0.4638028
## pgg45     0.164816193     6.1918214
```

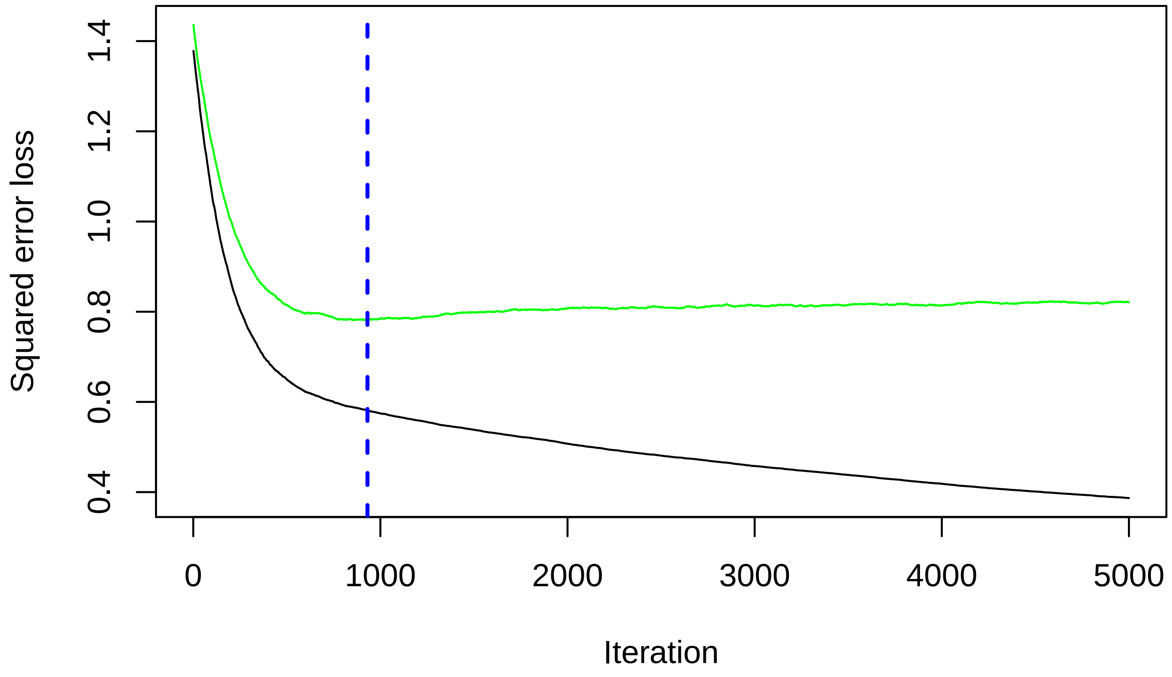
## Part D Random Forest

```
set.seed(1)
#Use number of variables divided by three rounded down
rf <-randomForest(lpsa~., Prostate, subset=train,
                  mtry = 2, importance=TRUE)
rf$importance
```

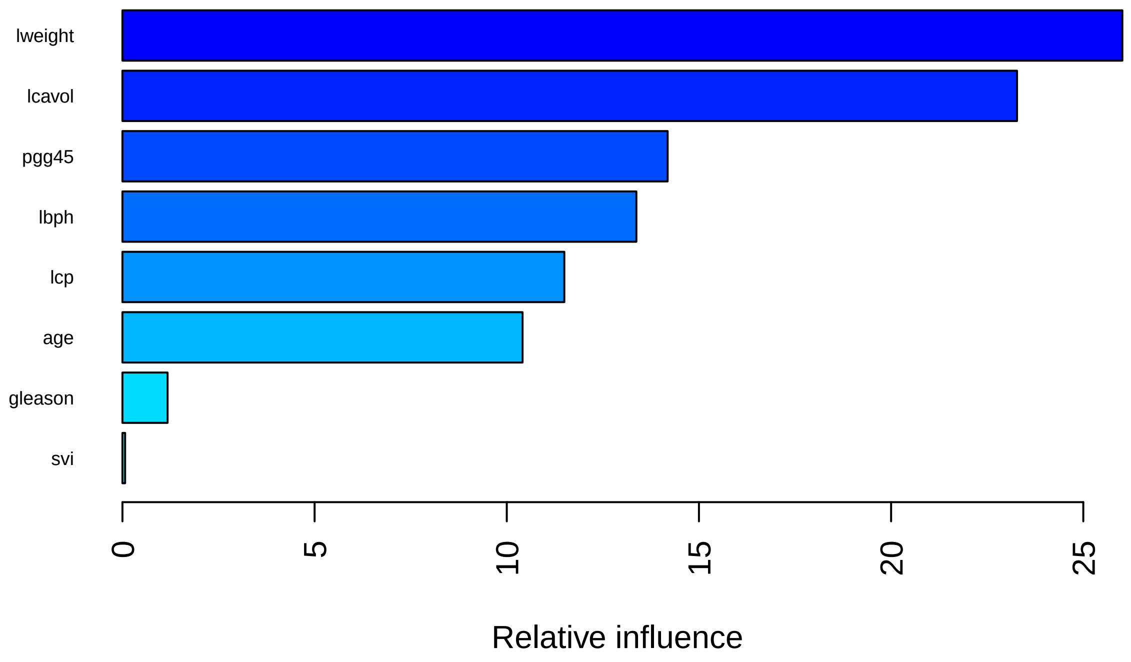
```
##              %IncMSE IncNodePurity
## lcavol   0.360217999     17.750096
## lweight  0.140601711      8.908025
## age     -0.031785756      4.647219
## lbph     0.003649344      2.835342
## svi      0.130686993      5.624124
## lcp      0.041385907      6.556900
## gleason  0.047201945      3.289278
## pgg45    0.142739905      7.809173
```

## Part E Boosting

```
set.seed(1)

bst <-gbm(lpsa~., data=Prostate[train,],
          distribution = "gaussian",
          n.trees = 5000,
          interaction.depth = 3,
          shrinkage = 0.005,
          cv.folds = 10)

ensemble.nt <-gbm.perf(bst, method = "cv")
```



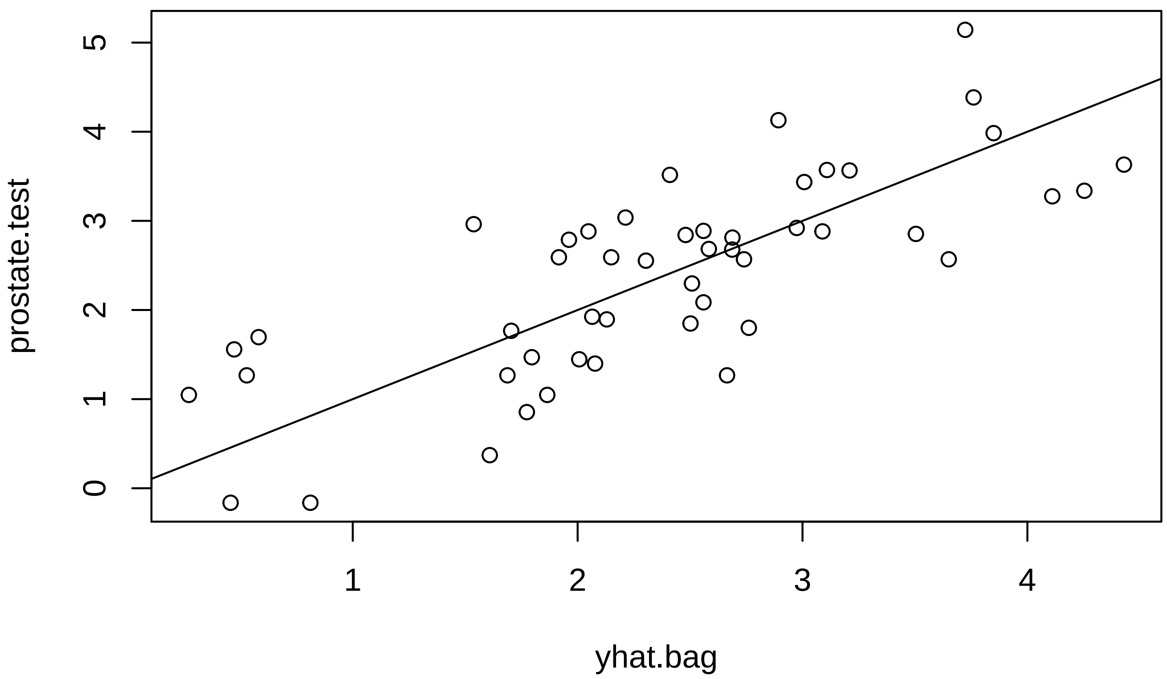
```r
summary(bst,las = 2, cBars = 19, cex.names = 0.6)
```



```
##             var     rel.inf
## lweight lweight 26.01662391
## lcavol   lcavol 23.27581748
## pgg45     pgg45 14.18539601
## lbph       lbph 13.37128891
## lcp         lcp 11.49657395
## age         age 10.41044346
## gleason gleason  1.17430737
## svi         svi  0.06954891
```

### Part F compare models

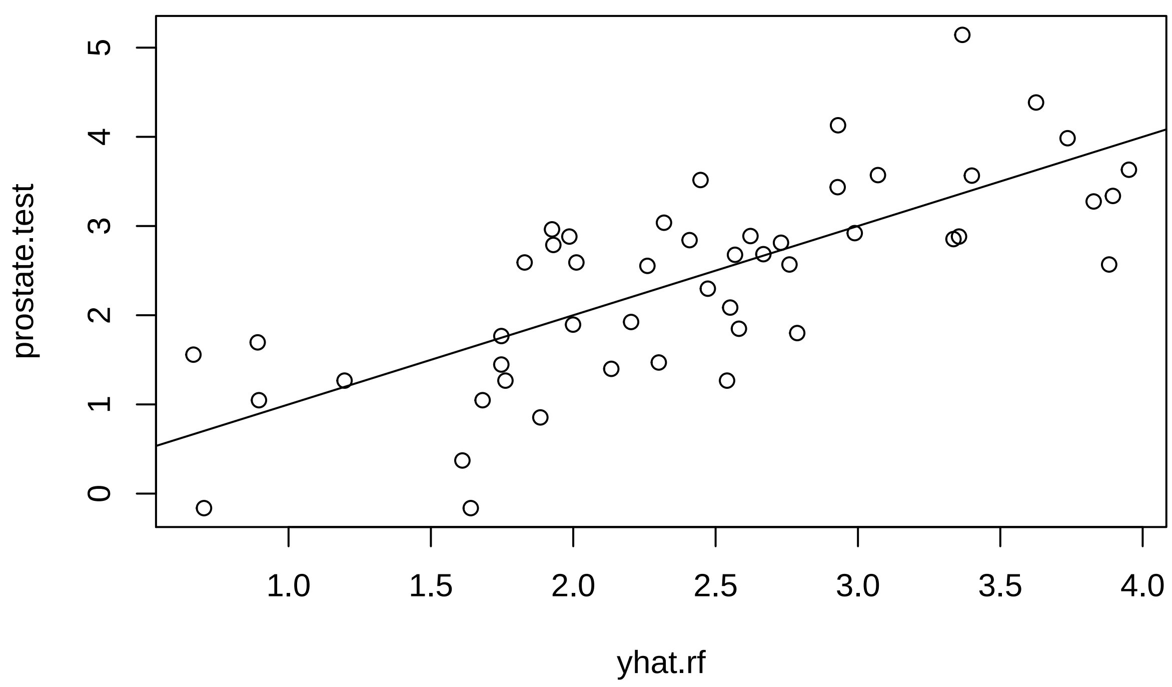
```
prostate.test=Prostate[-train,"lpsa"]

#Bag
yhat.bag = predict(bagging,newdata=Prostate[-train,])
plot(yhat.bag, prostate.test)
abline(0,1)
```



```
mean((yhat.bag-prostate.test)^2)
```

```
## [1] 0.5564165
```

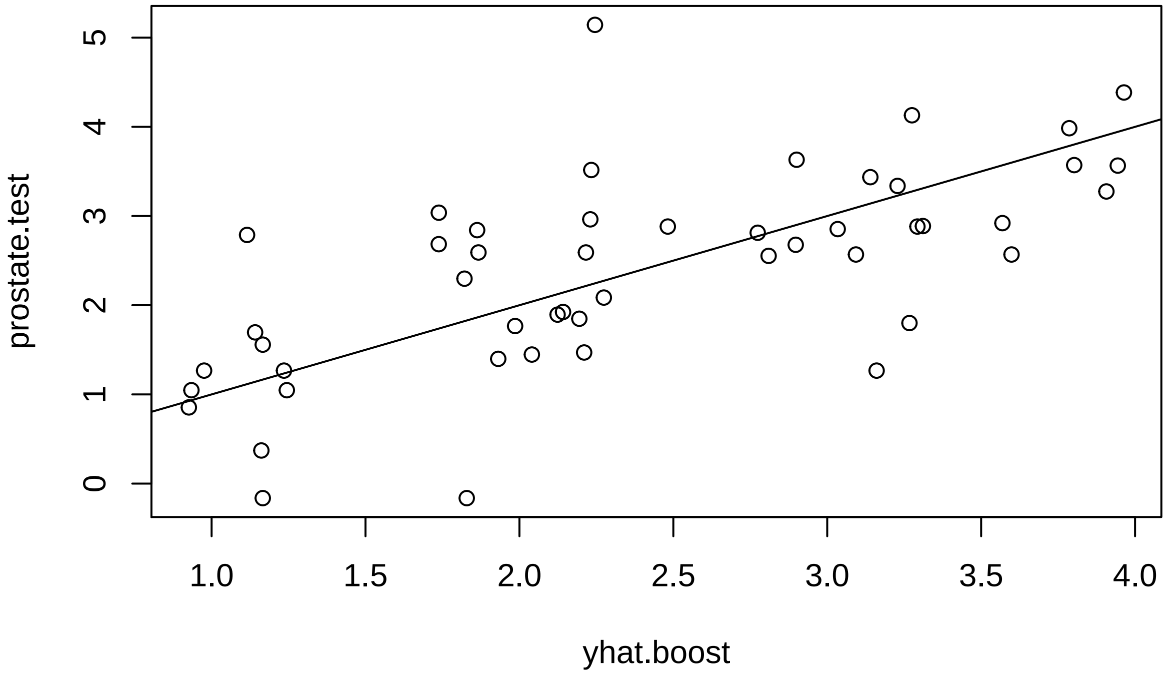
```
#RF
yhat.rf = predict(rf,newdata=Prostate[-train,])
plot(yhat.rf, prostate.test)
abline(0,1)
```



```
mean((yhat.rf-prostate.test)^2)
```

```
## [1] 0.5666352
```

```
#Boost
yhat.boost=predict(bst,newdata=Prostate[-train,],n.trees=5000)
plot(yhat.boost, prostate.test)
abline(0,1)
```



```
mean((yhat.boost-prostate.test)^2)
```

```
## [1] 0.746582
```

From the models fitted above, I would choose the bagged model as it has the lowest MSE, however, the boosted model should theoretically perform best, so if this were a project for work I would likely tune the model more carefully and chose a different corssvalidation method, and I would then expect the boosted model to perform best.

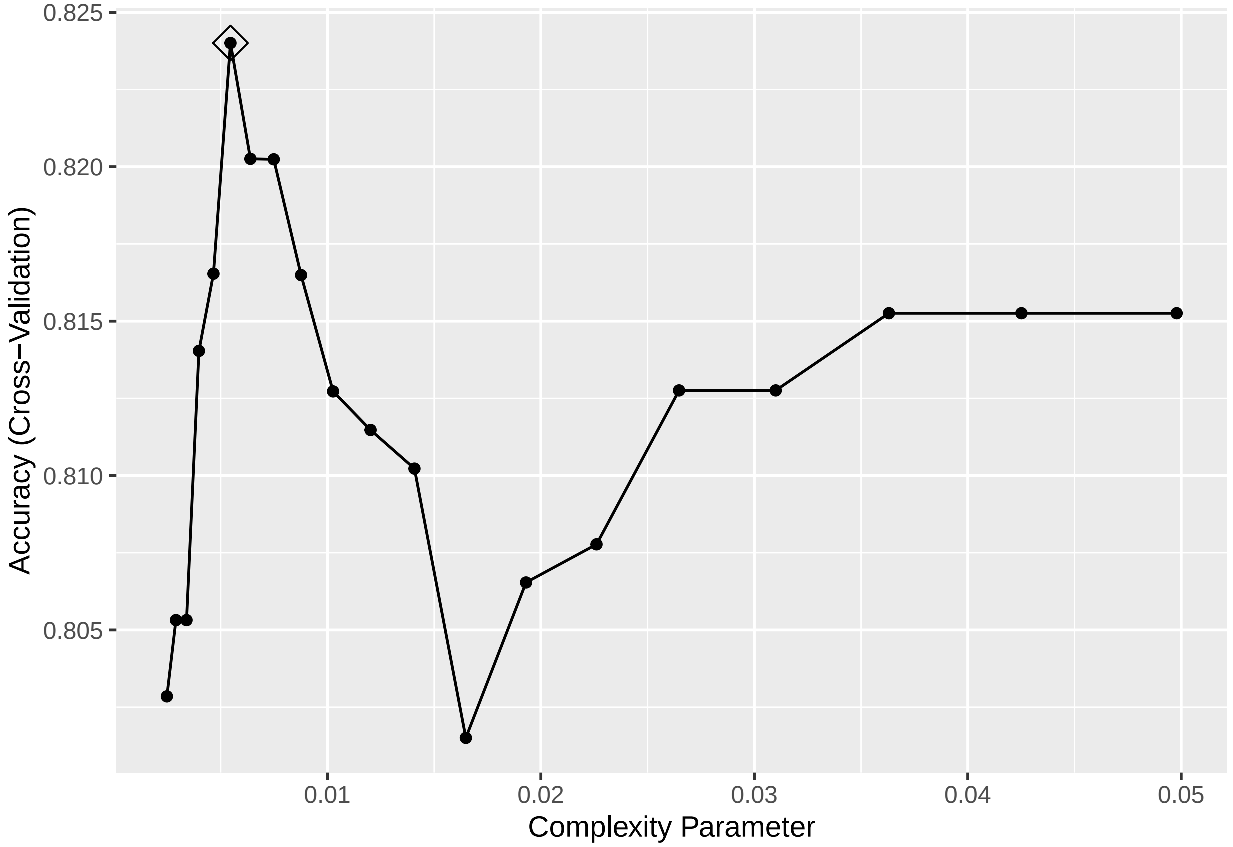
## Question 2

### Will be using Caret for this set of problems

```
data(OJ)

set.seed(1)
rowTrain = createDataPartition(OJ$Purchase,
                               p=800/1070,
                               list=F)

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

### Tree

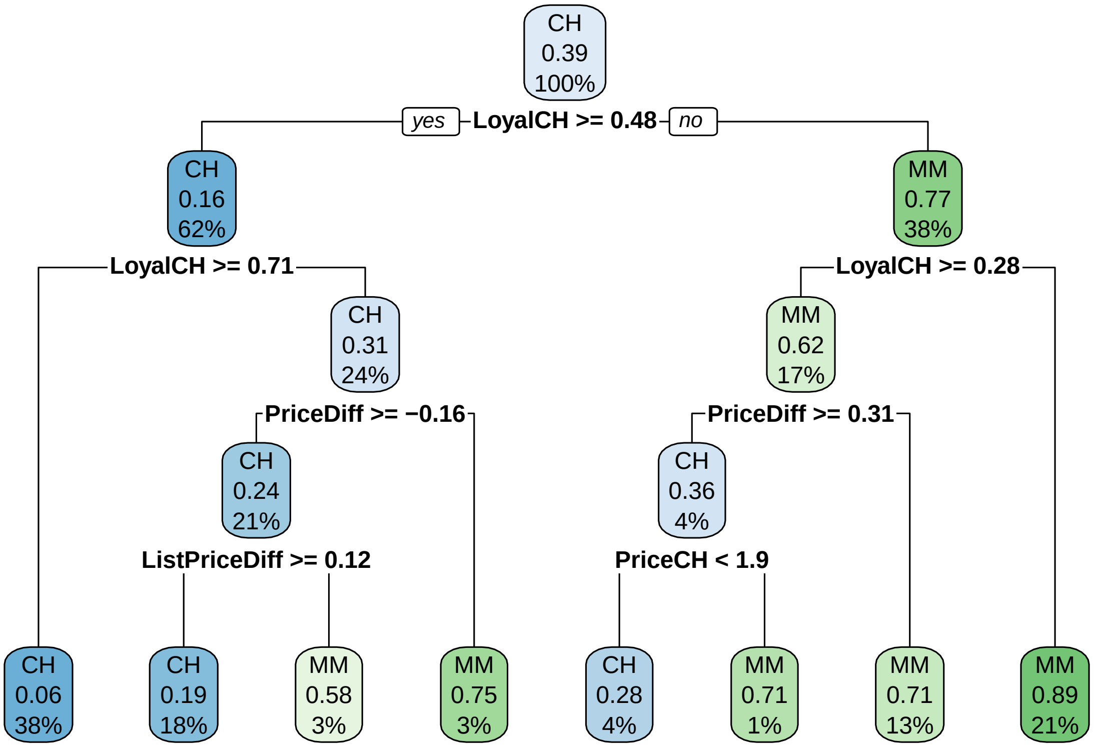
```
ctrl <-trainControl(method = "cv")

set.seed(1)
rpart.fit <-train(Purchase~., train,
                  method = "rpart",
                  tuneGrid =data.frame(cp =exp(seq(-6,-3, length = 20))),
                  trControl = ctrl)


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



```
rpart.fit$bestTune
```

```
##          cp
## 6 0.0054588
```

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



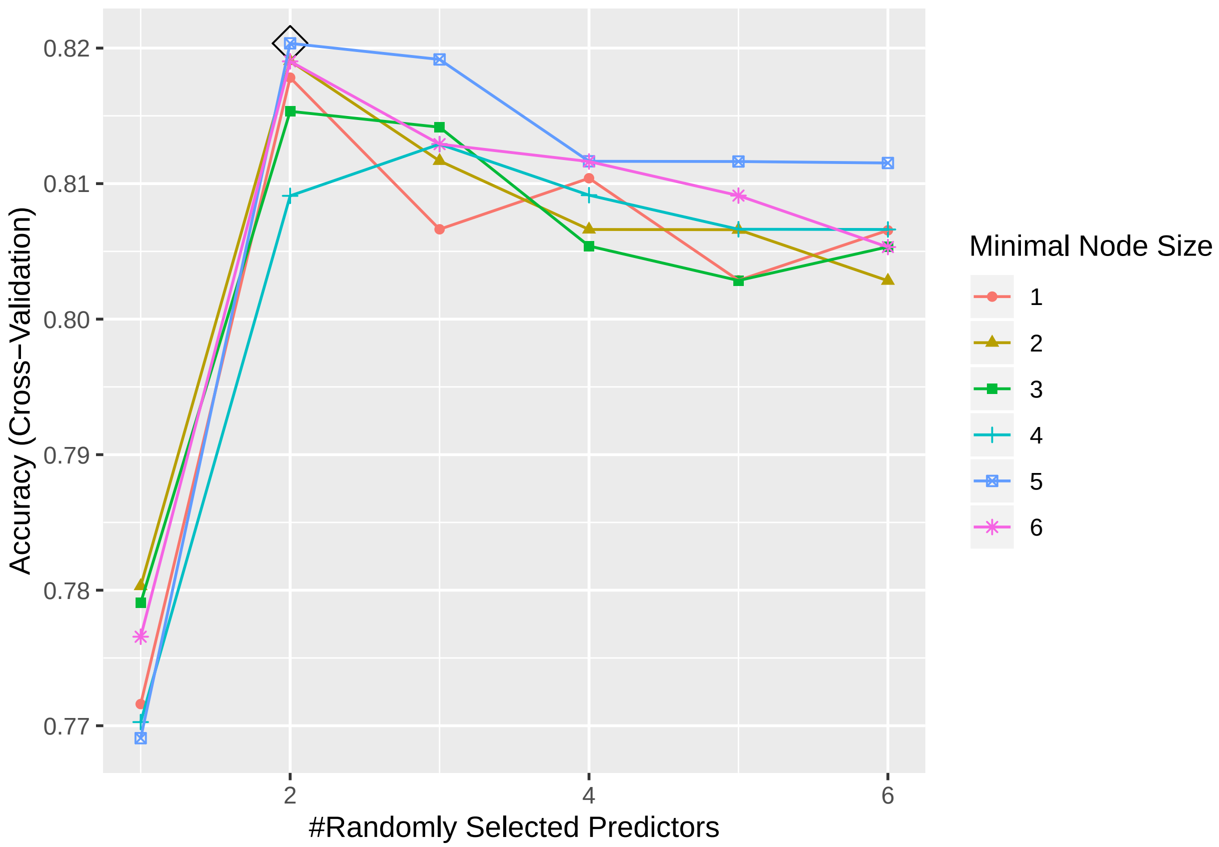
```
predy2.rpart <-predict(rpart.fit, newdata = test)

mse(predy2.rpart, test$Purchase)
```

```
## [1] 0.1895911
```

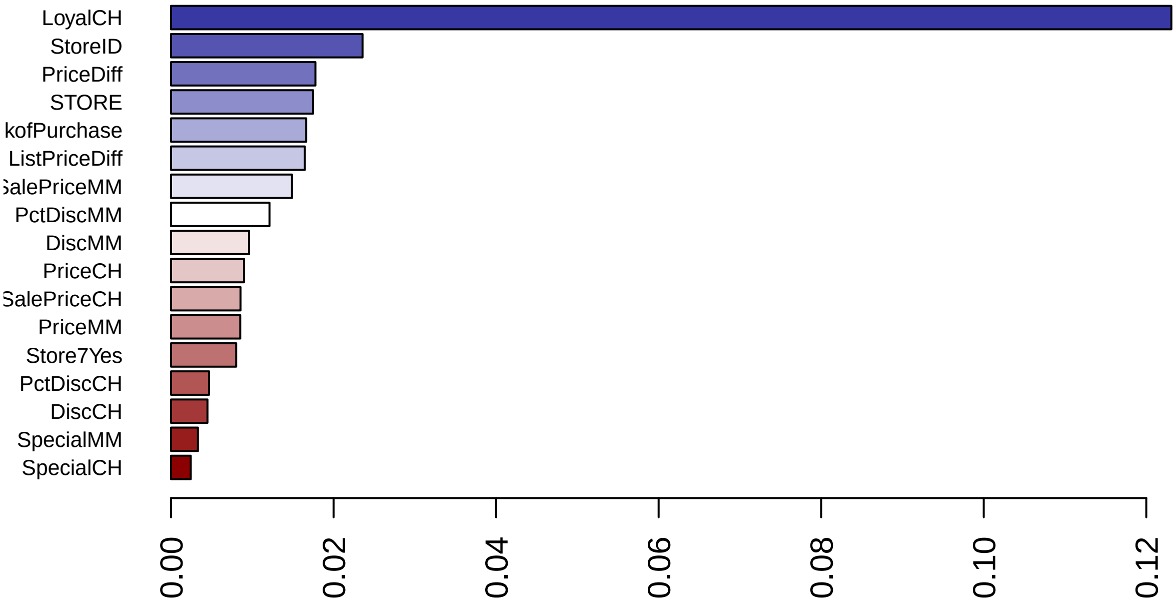
The test classification error rate is 18%

### Random Forest

```
# ctrl2 <-trainControl(method = "cv",
#                      classProbs=TRUE)

rf.grid <-expand.grid(mtry = 1:6,
                      splitrule = "gini",
                      min.node.size = 1:6)
set.seed(1)
rf.fit <-train(Purchase~., train,
               method = "ranger",
               tuneGrid=rf.grid,
               trControl=ctrl,
               importance="permutation")

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



```
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))
```



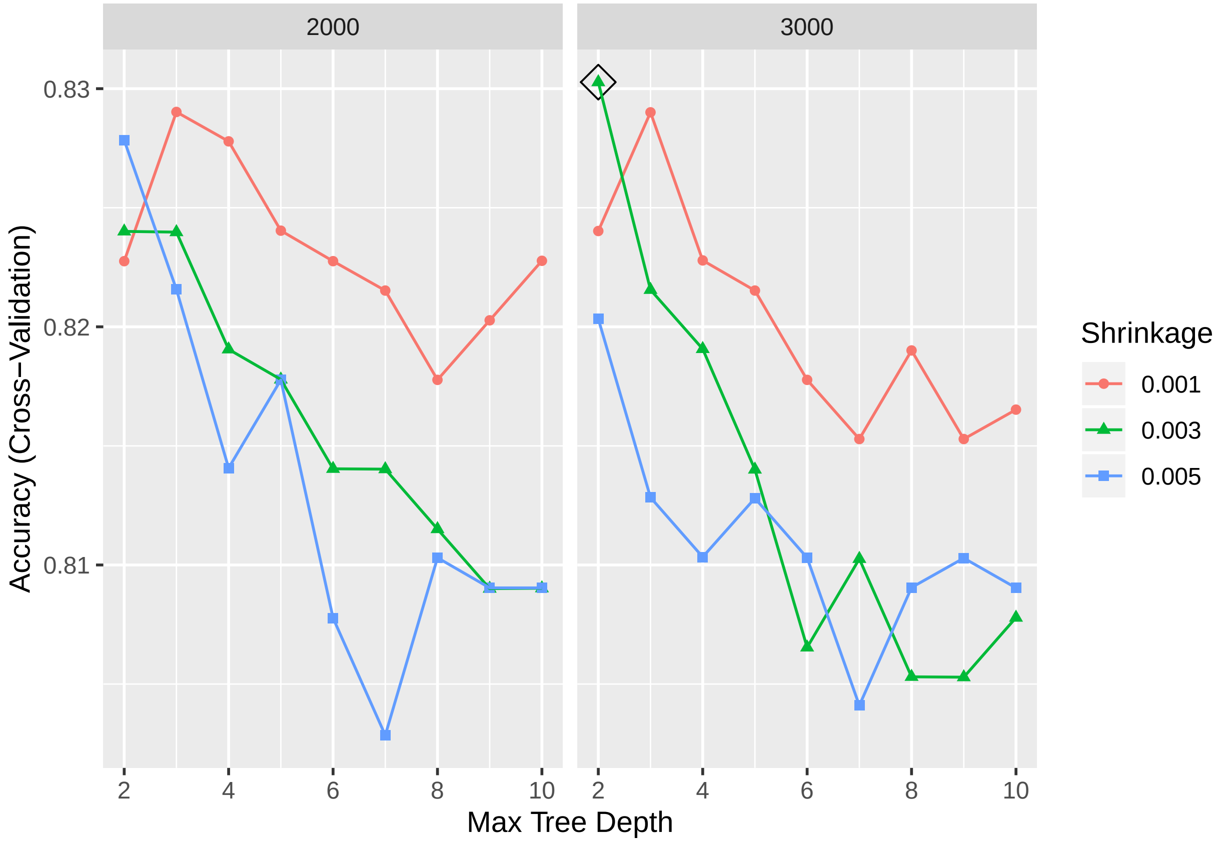
```
predy2.rf <-predict(rf.fit, newdata = test)

mse(predy2.rf, test$Purchase)
```

```
## [1] 0.1933086
```

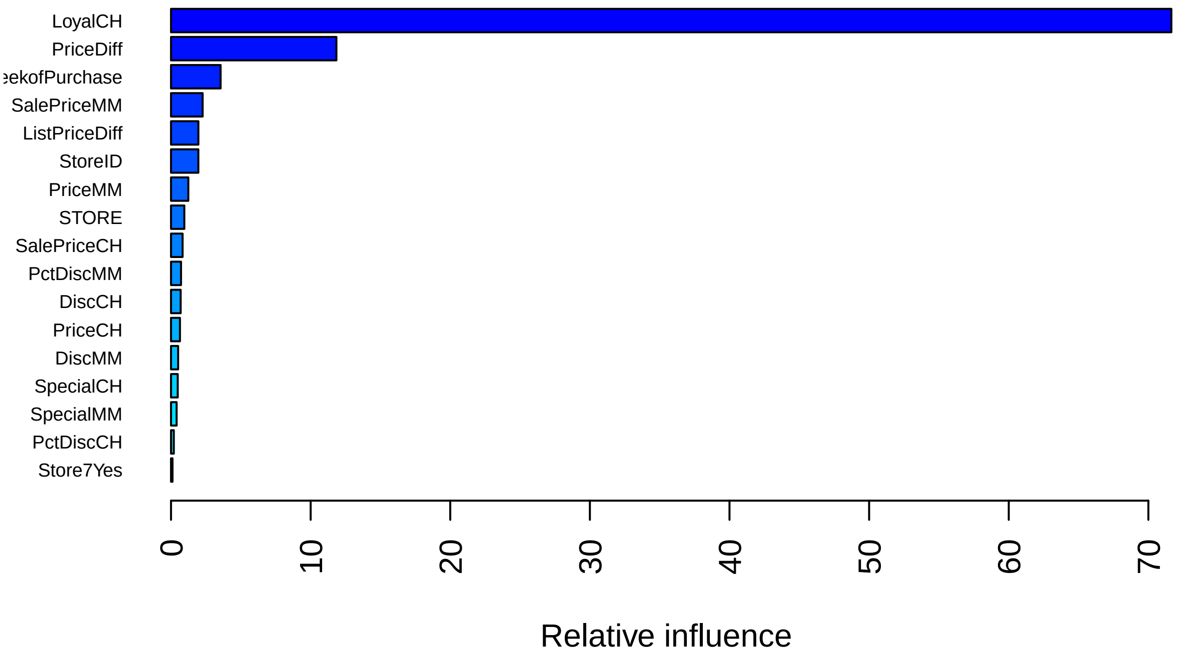
The MSE is 0.197

## Part C Boosting

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

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



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



```
##                           var    rel.inf
## LoyalCH               LoyalCH 71.6466744
## PriceDiff           PriceDiff 11.8464989
## WeekofPurchase WeekofPurchase  3.5513195
## SalePriceMM       SalePriceMM  2.2633542
## ListPriceDiff   ListPriceDiff  1.9609256
## StoreID               StoreID  1.9592341
## PriceMM               PriceMM  1.2391999
## STORE                   STORE  0.9539062
## SalePriceCH       SalePriceCH  0.8299615
## PctDiscMM           PctDiscMM  0.7168251
## DiscCH                 DiscCH  0.6911404
## PriceCH               PriceCH  0.6405940
## DiscMM                 DiscMM  0.5105453
## SpecialCH           SpecialCH  0.4813723
## SpecialMM           SpecialMM  0.4003391
## PctDiscCH           PctDiscCH  0.2014882
## Store7Yes           Store7Yes  0.1066213
```

```
predy2.gbm <-predict(gbm.fit, newdata = test)

mse(predy2.gbm, test$Purchase)
```

```
## [1] 0.1784387
```

The error is 0.17