DS HW4 Tree Based Models

JunLu

4/16/2019

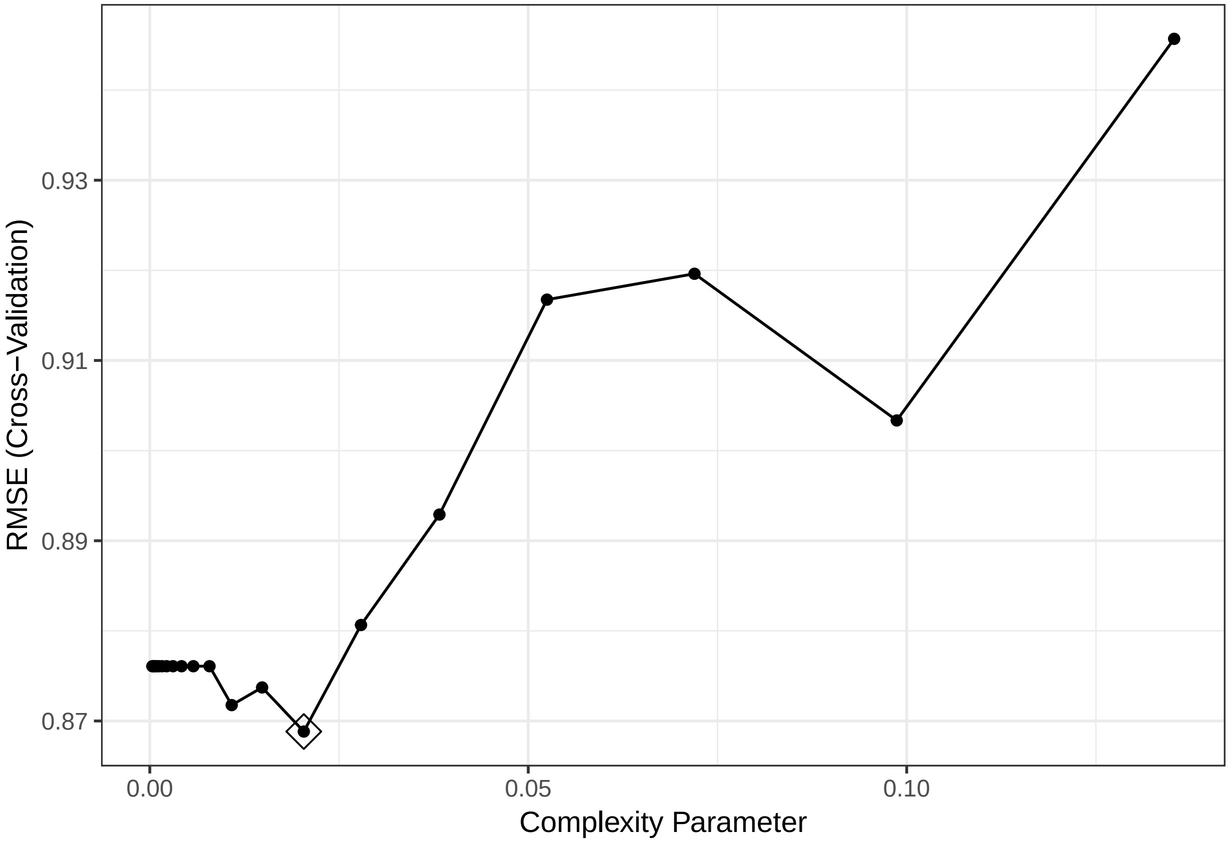
Problem 1

a) Fit a regression tree

The lowest cross-validation error

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

```r
set.seed(1)

tree_fit = train(lpsa~.,
            data = Prostate,
            method = "rpart",
            tuneGrid = data.frame(cp =exp(seq(-8,-2, length = 20))),
            trControl = ctrl)

ggplot(tree_fit, highlight = TRUE)
```



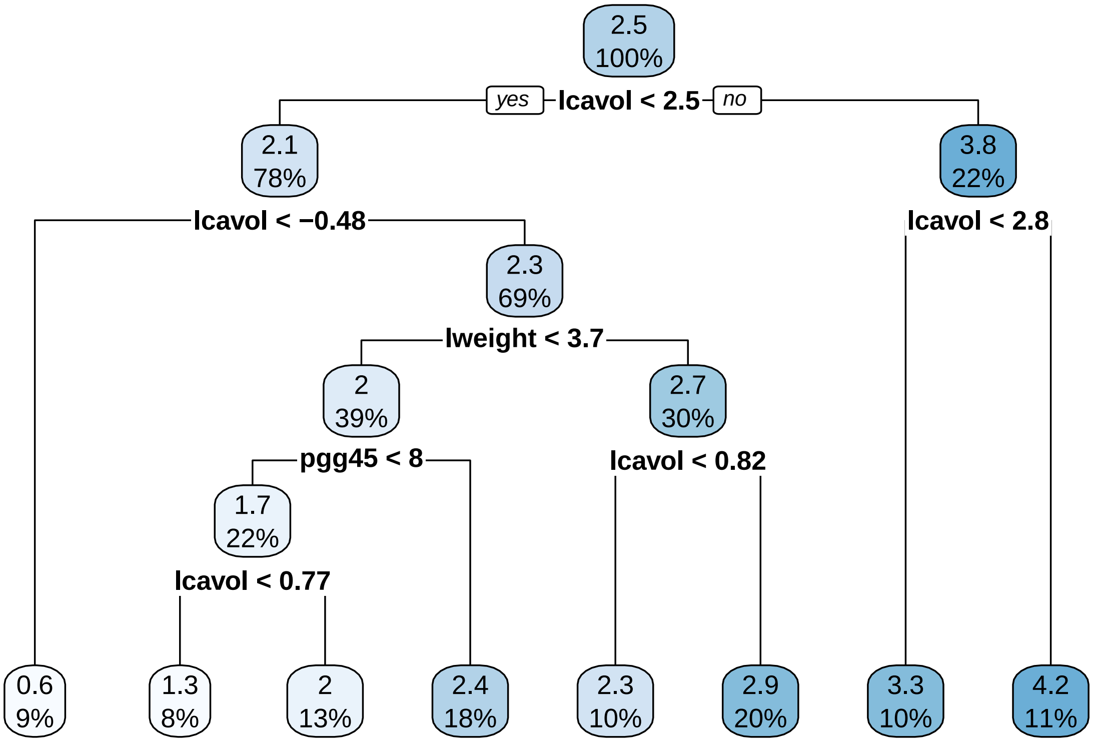
```r
tree_fit$bestTune
```

```
##            cp
## 14 0.02034873
```

```
tree_fit$finalModel$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 0.02156368      5 0.3395631
## 7 0.02146995      6 0.3179994
## 8 0.00000000      7 0.2965295
```

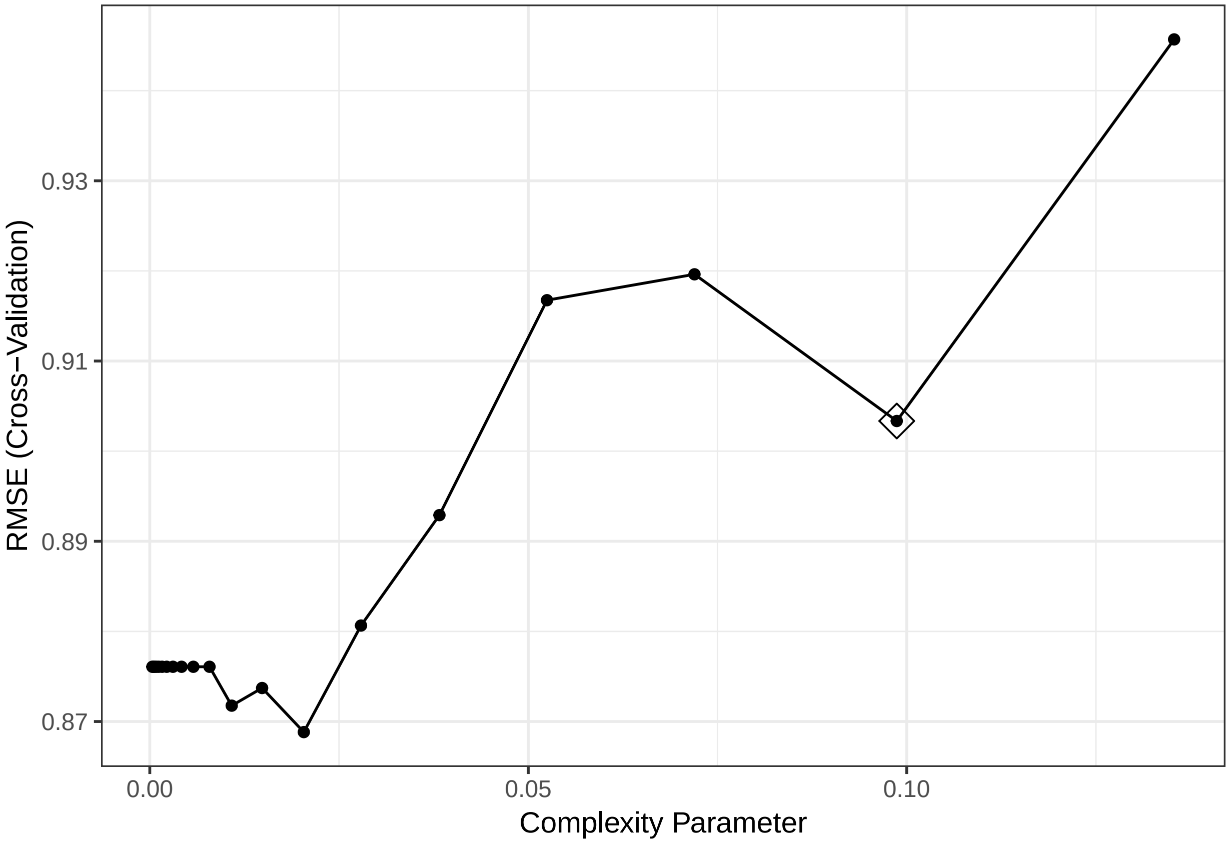
```
rpart.plot(tree_fit$finalModel)
```



The tree size corresponds to the lowest cross-validation error is 8

The 1 SE rule

```
set.seed(1)
tree_fit_2 = train(lpsa~.,
            data = Prostate,
            method = "rpart",
            tuneGrid = data.frame(cp =exp(seq(-8,-2, length = 20))),
            trControl = trainControl(method = "cv",
                                     number = 10,
                                     selectionFunction = "oneSE"))

ggplot(tree_fit_2, highlight = TRUE)
```



```
tree_fit_2$finalModel$cptable
```

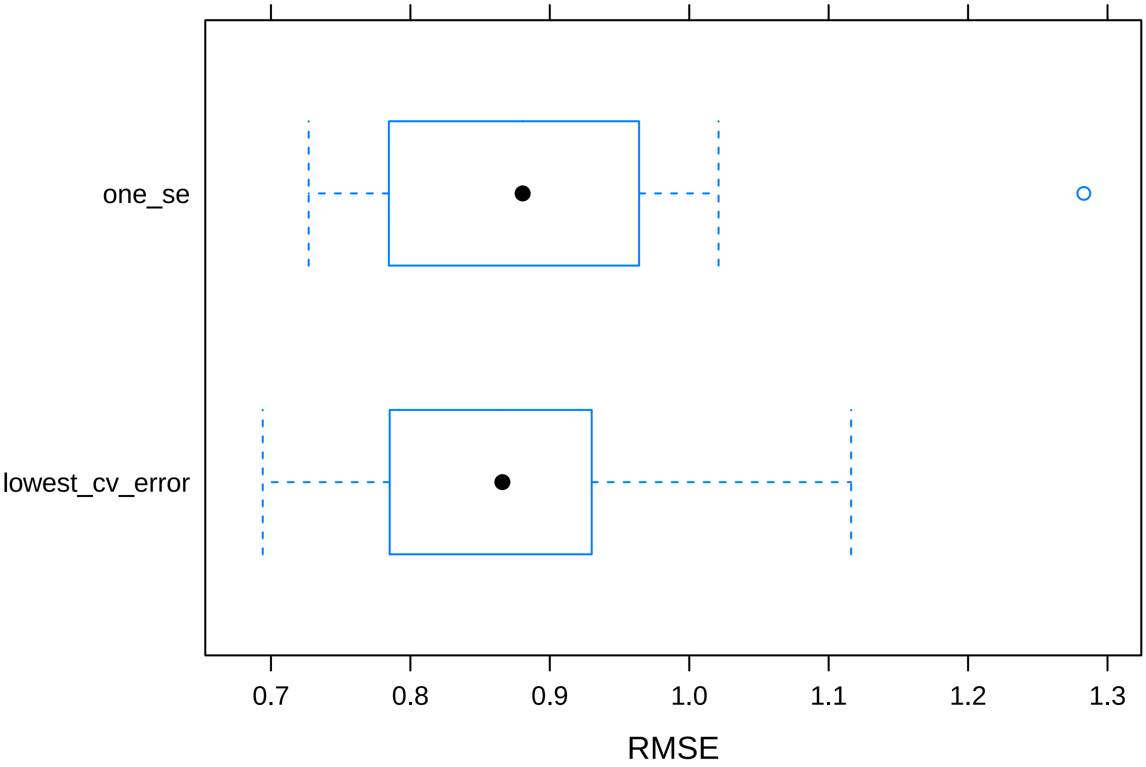
```
##           CP nsplit rel error
## 1 0.34710828      0 1.0000000
## 2 0.18464743      1 0.6528917
## 3 0.09868824      2 0.4682443
```

The tree size obtained using the 1 SE rule is 3.

The two tree sizes obtained by different selection functions are different.

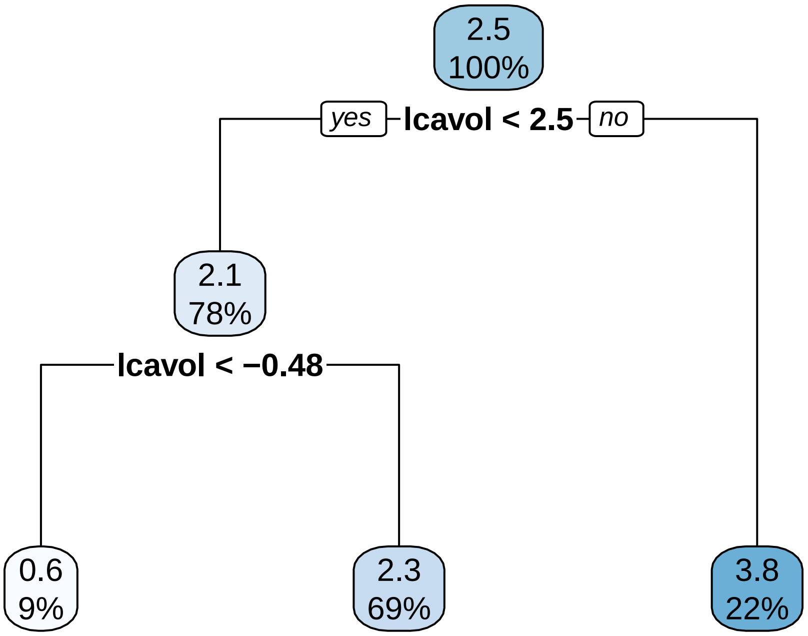
b) Choose one decision tree model

```
resamp <- resamples(list(lowest_cv_error = tree_fit,
                         one_se = tree_fit_2
                         ))
bwplot(resamp, metric = "RMSE")
```



I used tree size 3 as it has a similar performance with size 8 and it is simpler.

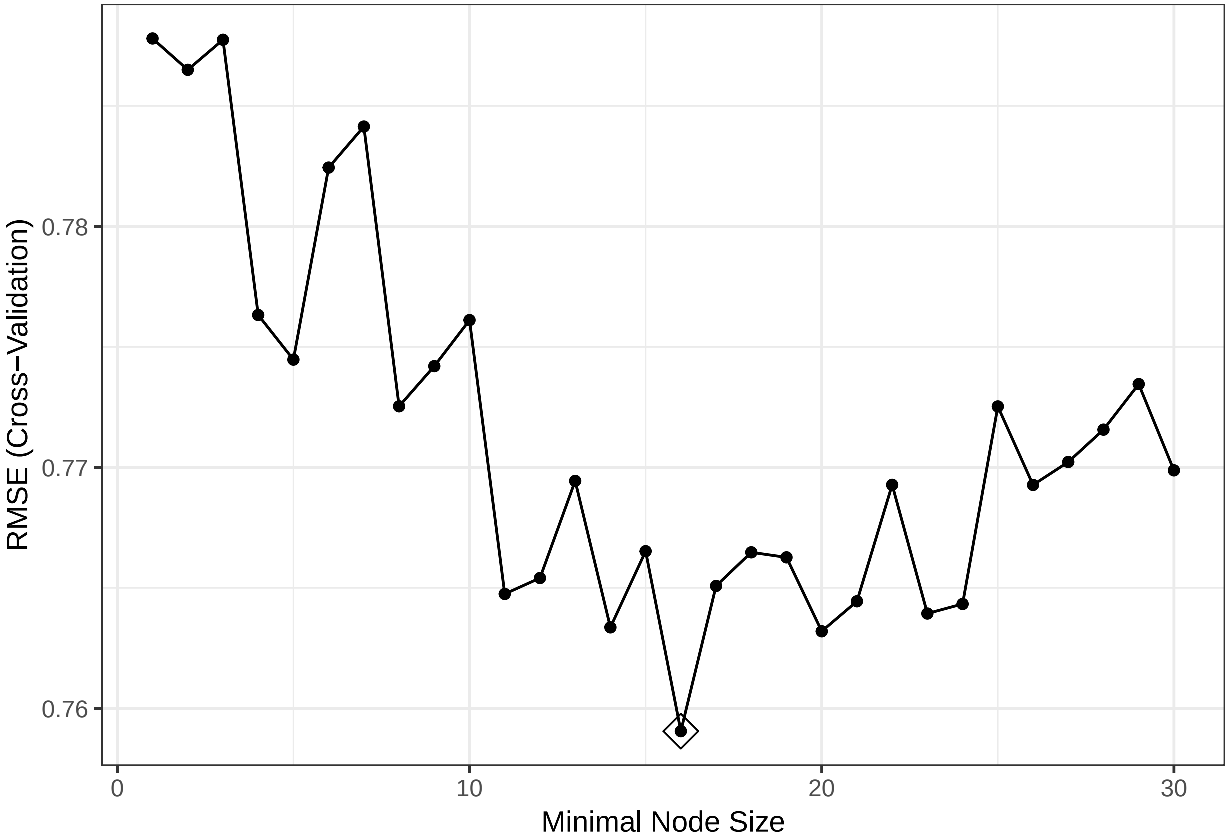
```r
rpart.plot(tree_fit_2$finalModel)
```



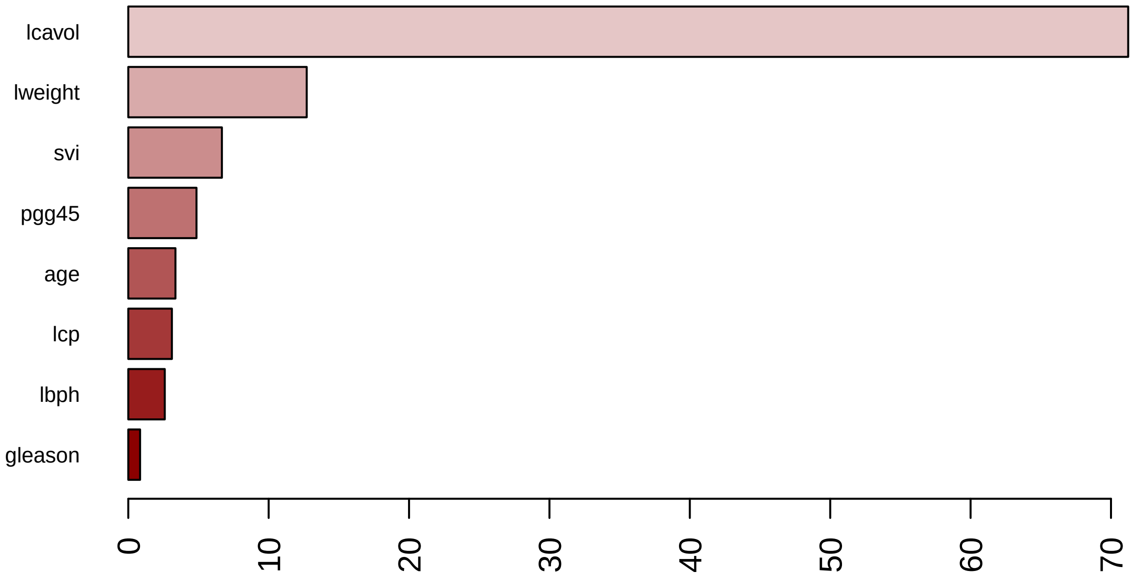
The first terminal node in the plot: When the lcavol is smaller than -0.48 (firstly smaller than 2.5), the predicted value (or the mean of observations in this terminal node) is 0.6. This terminal node contains 9% training data observations.

c) Bagging

```
set.seed(1)

bagging.grid <- expand.grid(mtry = 8,
                            splitrule = "variance",
                            min.node.size = 1:30
                            )

bag_fit = train(lpsa~.,
                data = Prostate,
                method = "ranger",
                tuneGrid = bagging.grid,
                importance = "impurity",
                trControl = ctrl)

ggplot(bag_fit, highlight = T)
```



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

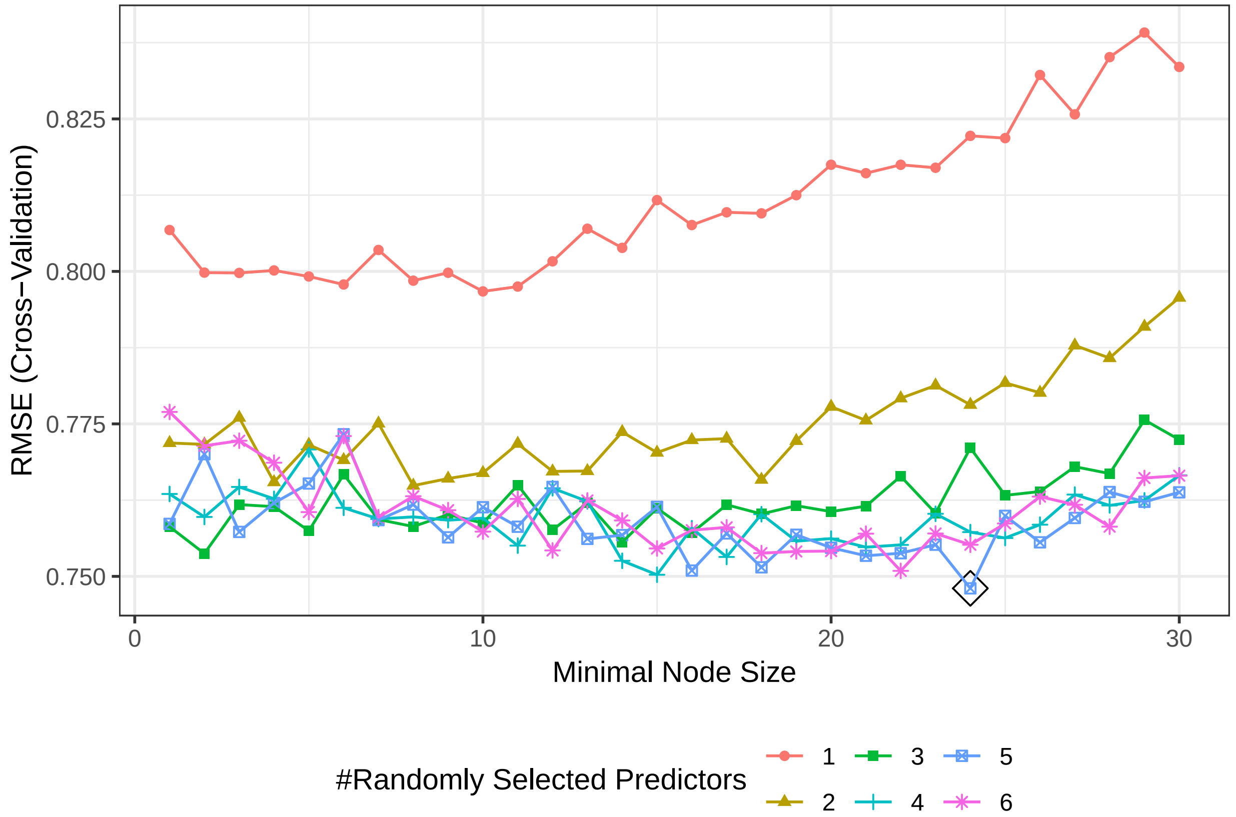


The lcavol is the most importance variable in this bagging model.

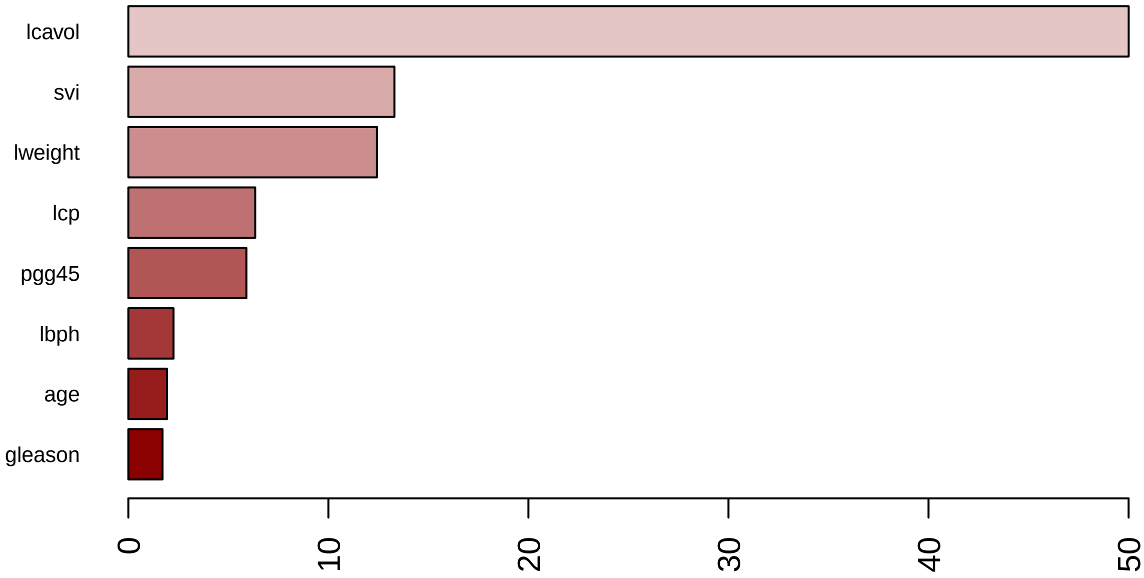
Importance: lcavol > lweight > svi > pgg45 > age > lcp > lbph > gleason

d) Random Forests

```
set.seed(1)

rf.grid <- expand.grid(mtry = 1:6,
                       splitrule = "variance",
                       min.node.size = 1:30
                       )

rf_fit = train(lpsa~.,
                data = Prostate,
                method = "ranger",
                tuneGrid = rf.grid,
                importance = "impurity",
                trControl = ctrl)

ggplot(rf_fit, highlight = T)
```



```r
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))
```

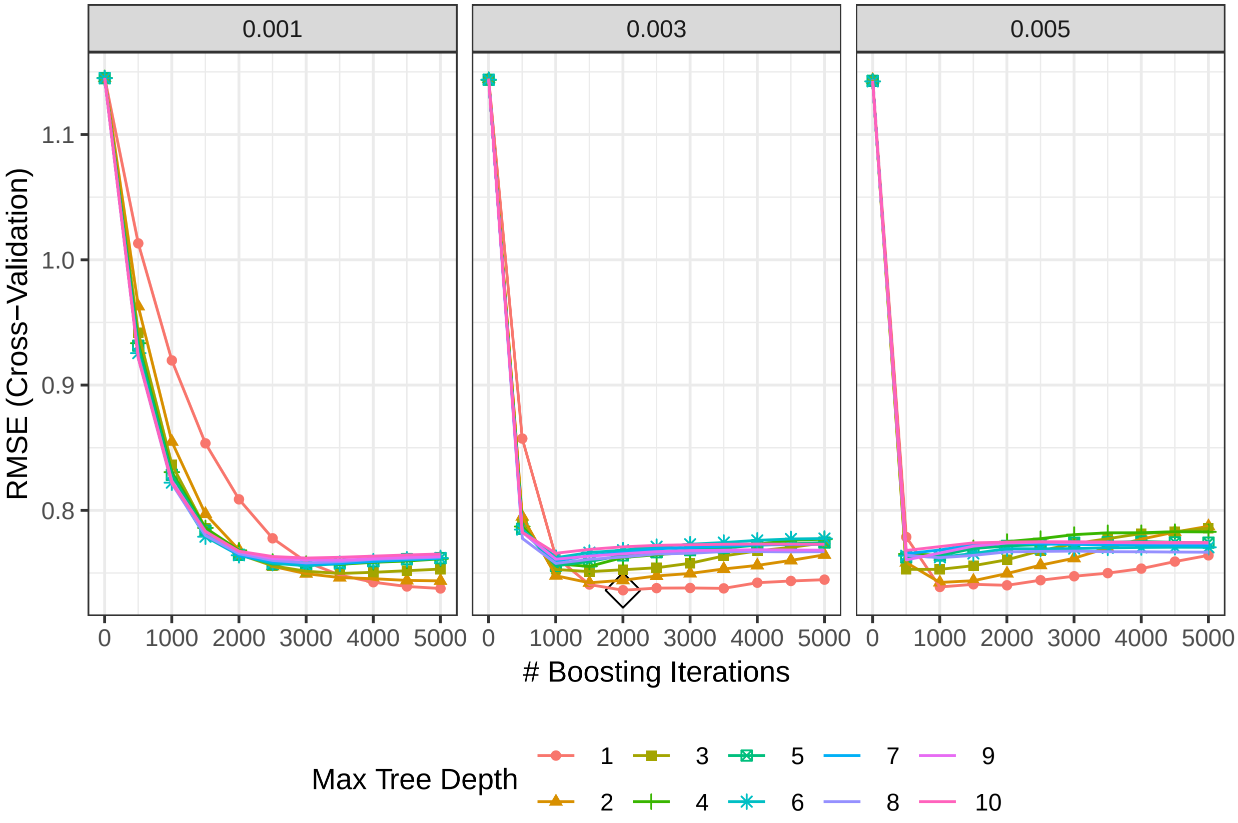


The lcavol is the most importance variable in this random forests model.

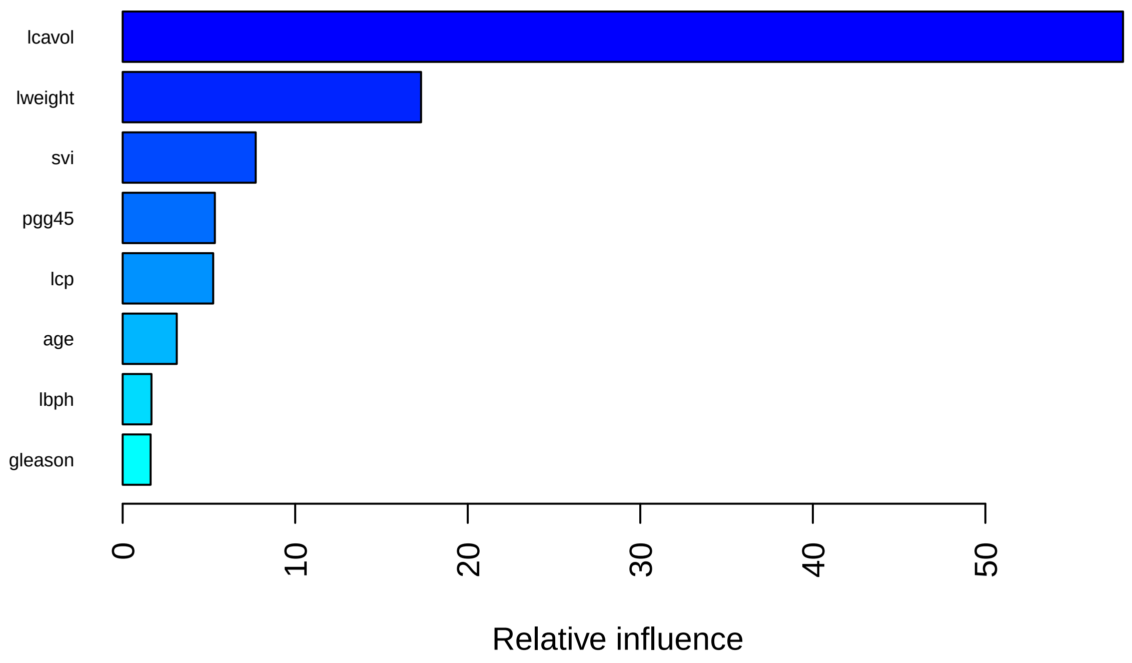
Importance: lcavol > svi > lweight > lcp > pgg45 > lbph > age > gleason

e) Boosting

```
set.seed(1)
gbm.grid = expand.grid(
    n.trees = seq(1,5001, by = 500),
    interaction.depth = 1:10,
    shrinkage = c(0.001, 0.003, 0.005),
    n.minobsinnode = 1
    )

gbm_fit = train(lpsa~.,
                data = Prostate,
                method = "gbm",
                tuneGrid = gbm.grid,
                trControl = ctrl,
                verbose = FALSE)

ggplot(gbm_fit, highlight = T)
```



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



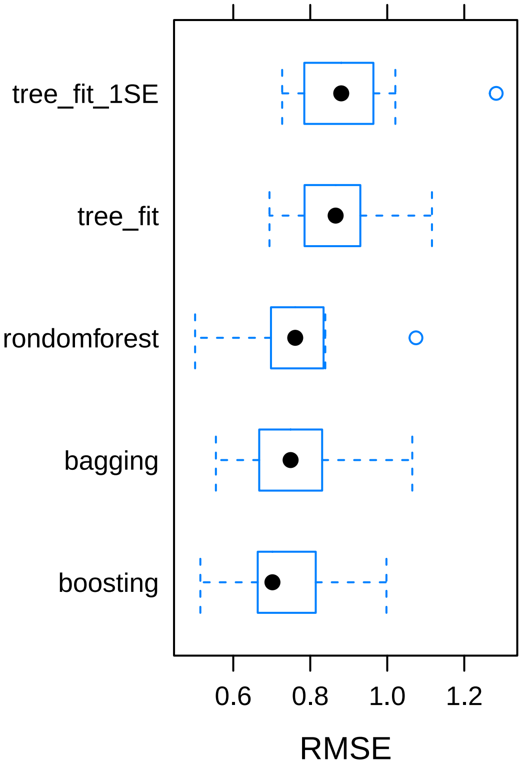
```
##             var   rel.inf
## lcavol   lcavol 57.977658
## lweight lweight 17.294196
## svi         svi  7.711160
## pgg45     pgg45  5.343532
## lcp         lcp  5.247288
## age         age  3.135510
## lbph       lbph  1.669569
## gleason gleason  1.621087
```

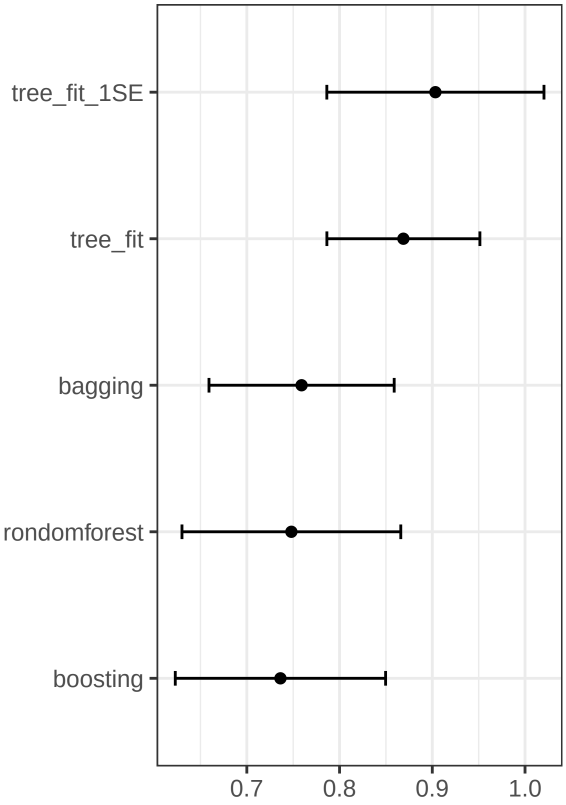
The lcavol is the most importance variable in this boosting model.

Importance: lcavol > lweight > svi > lcp > pgg45 > age > lbph > gleason

d) Compare Models

```
resamp <- resamples(list(tree_fit = tree_fit,
                         tree_fit_1SE = tree_fit_2,
                         bagging = bag_fit,
                         rondomforest = rf_fit,
                         boosting = gbm_fit
                         ))
a = bwplot(resamp, metric = "RMSE")
b = ggplot(resamp, metric = "RMSE")

gridExtra::grid.arrange(a,b,ncol = 2,nrow = 1)
```





From the boxplots of RMSE in the cross-vaildation, we can see that ensemble methods (bagging, random forerst and boosting) have a better performance in the cross-vaildation than the simple decision tree model. Comparing means of RMSE of different models in the cross-vaildation, we choose the boosting model to predict PSA level as it has the lowest mean.

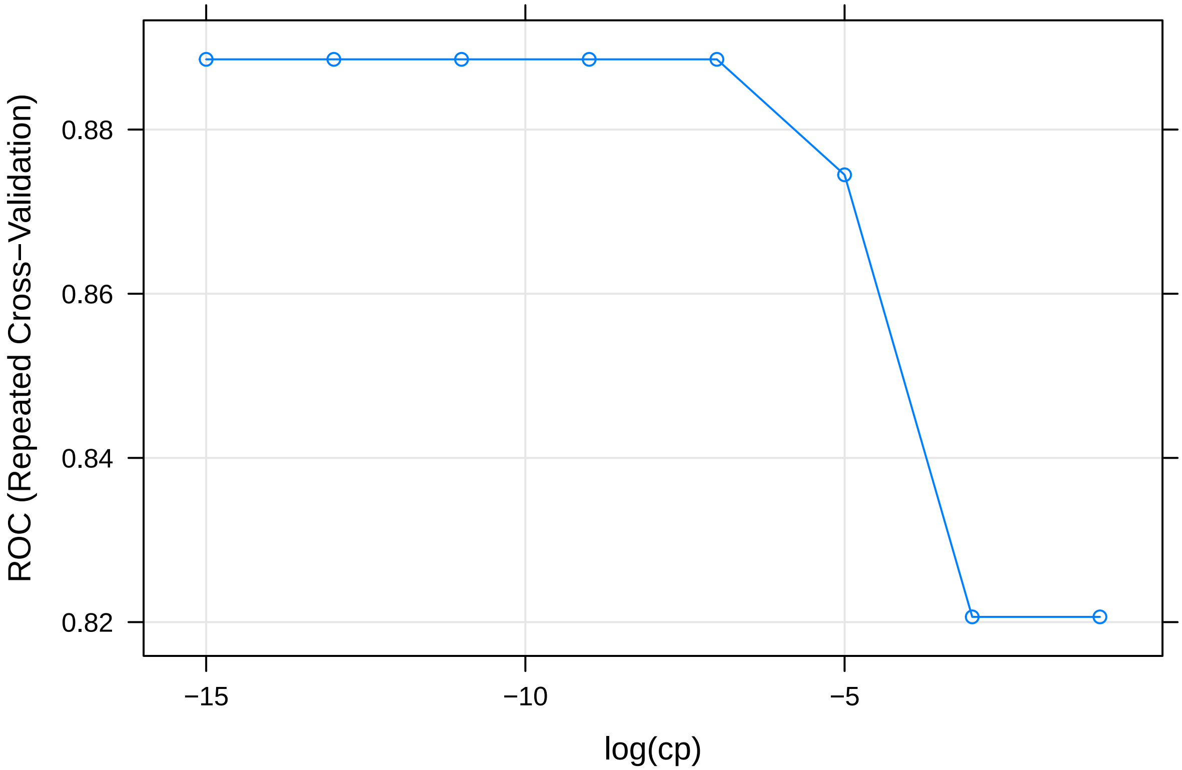
Problem 2

a) Decision Tree

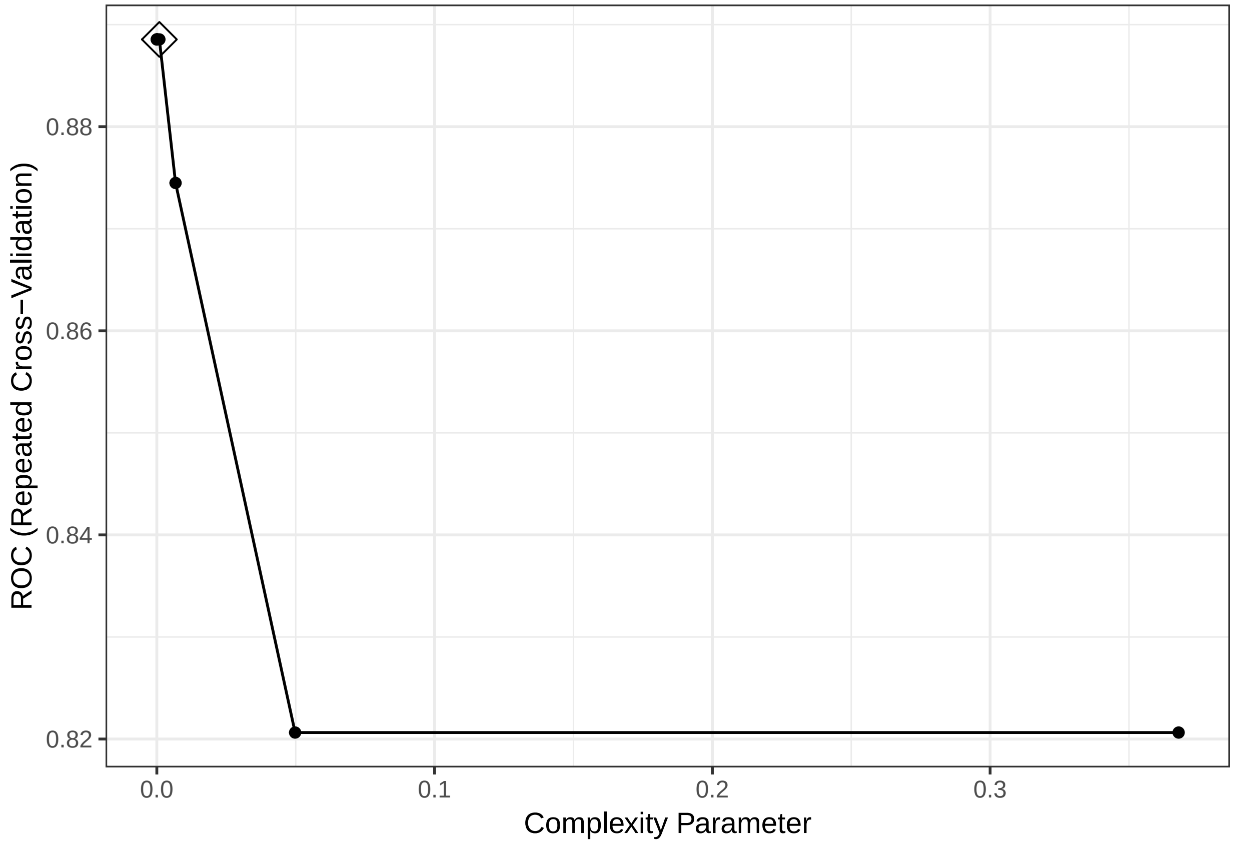
```
data(OJ)
```

```
set.seed(1)

train_ind <- sample(seq_len(nrow(OJ)), size = 800)

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

```
set.seed(1)
ctrl = trainControl(method = "repeatedcv",
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE)

tree_fit_c = train(Purchase~.,
                   data = training,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-15,0, by = 2))),
                   trControl = ctrl,
                   metric = "ROC"
                   )

plot(tree_fit_c, xTrans = function(x)log(x), xlab = "log(cp)")
```



```
ggplot(tree_fit_c, highlight = T)
```



```
tree_fit_c$bestTune
```

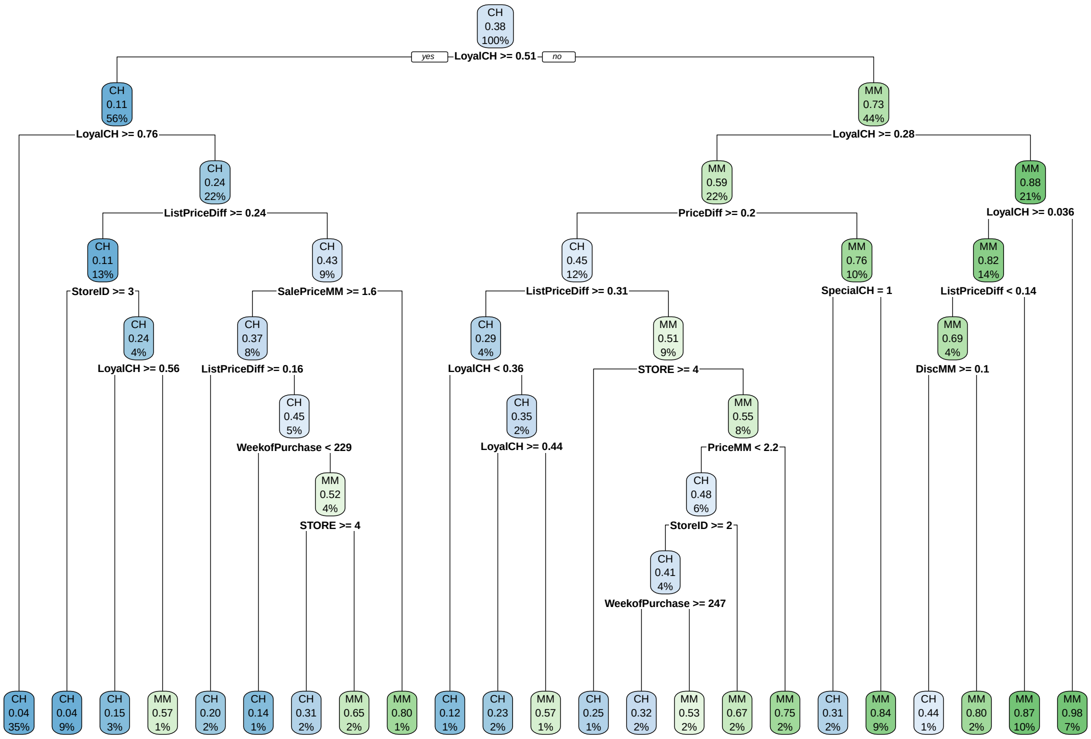
```
##            cp
## 5 0.000911882
```

```
tree_fit_c$finalModel$cptable
```

```
##            CP nsplit rel error
## 1 0.522875817      0 1.0000000
## 2 0.016339869      1 0.4771242
## 3 0.009803922      4 0.4281046
## 4 0.006535948      8 0.3888889
## 5 0.003267974     14 0.3496732
## 6 0.001633987     15 0.3464052
## 7 0.001089325     19 0.3398693
## 8 0.000000000     22 0.3366013
```

The plot of the final tree

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



```
tree.pred = predict(tree_fit_c, newdata = test, type = "raw")

1 - sum(tree.pred == test$Purchase) / length(test$Purchase)
```

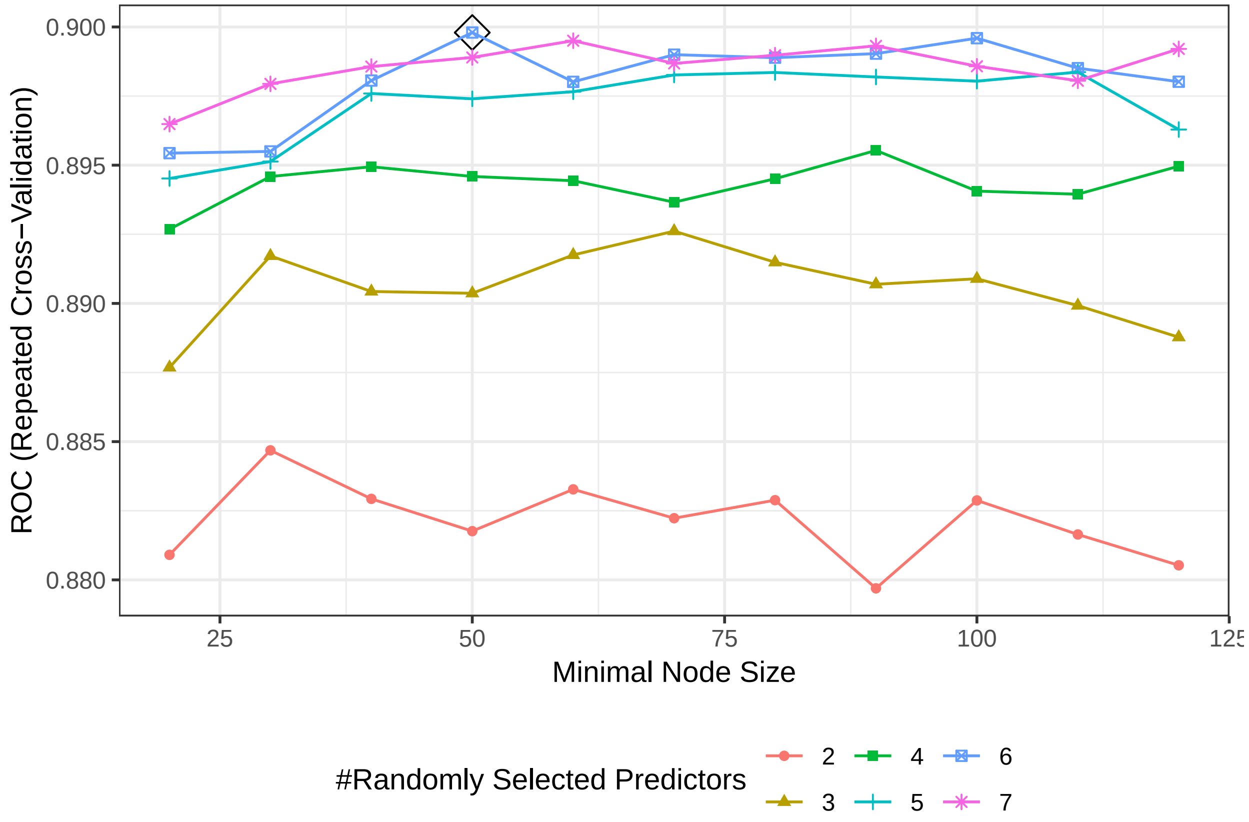
```
## [1] 0.1925926
```

Test classification error rate

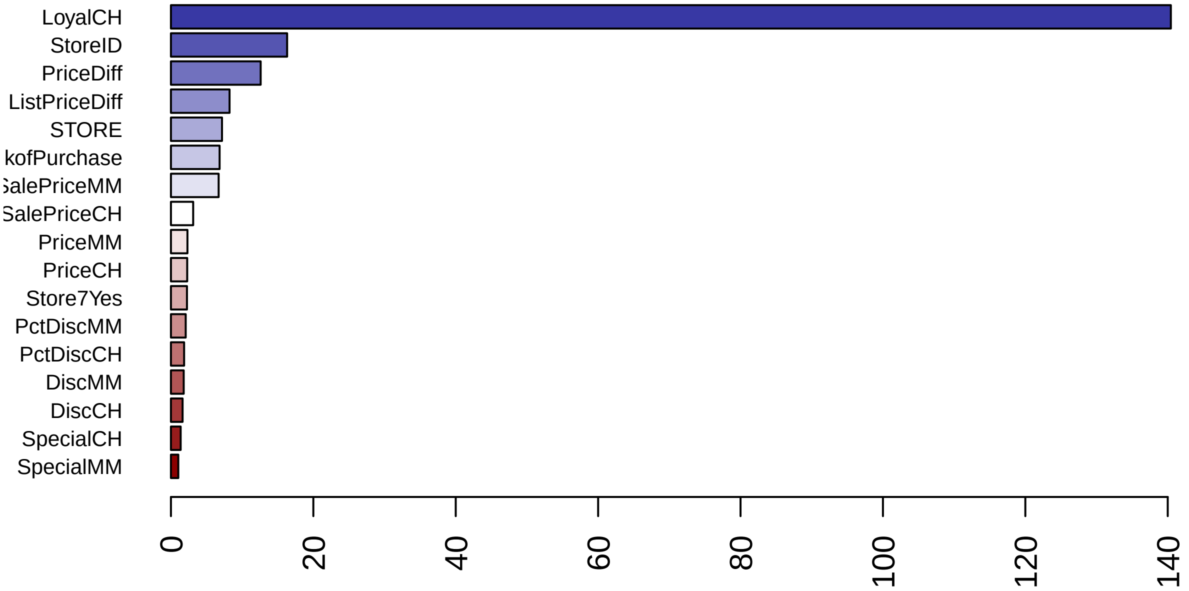
The tree size is 23. The test classification error rate is 19.26% for this tree model.

b)Random forests

```
set.seed(1)
rf.grid = expand.grid(mtry = 2:7,
                      splitrule = "gini",
                      min.node.size = seq(20,120, by = 10))

rf_fit_c = train(Purchase~.,
                 data = training,
                 method = "ranger",
                 tuneGrid = rf.grid,
                 metric = "ROC",
                 trControl = ctrl,
                 importance = "impurity")

ggplot(rf_fit_c, highlight = T)
```



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



```
rf.pred = predict(rf_fit_c, newdata = test, type = "raw")

1 - sum(rf.pred == test$Purchase) / length(test$Purchase)
```

```
## [1] 0.1703704
```

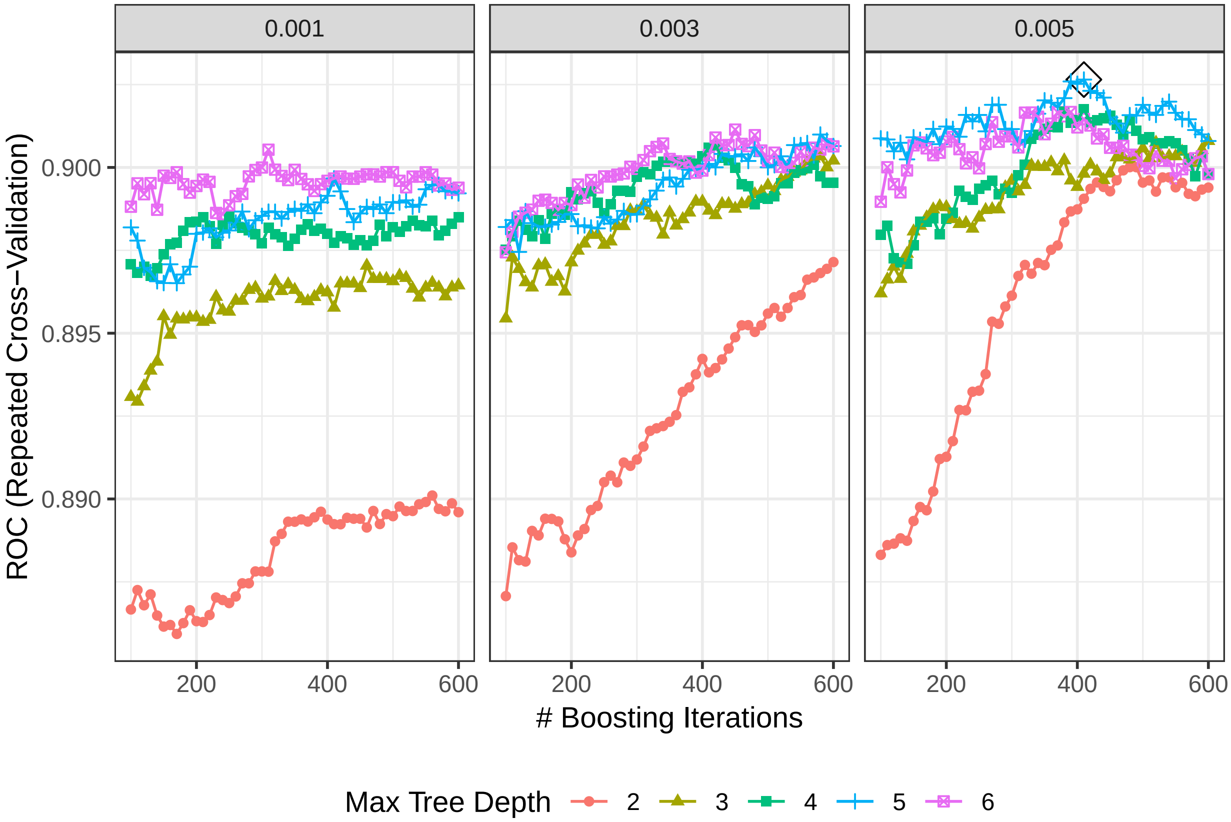
The test classification error rate is 17.04% for this tree model.

The loyalCH is the most importance variable in this boosting model.

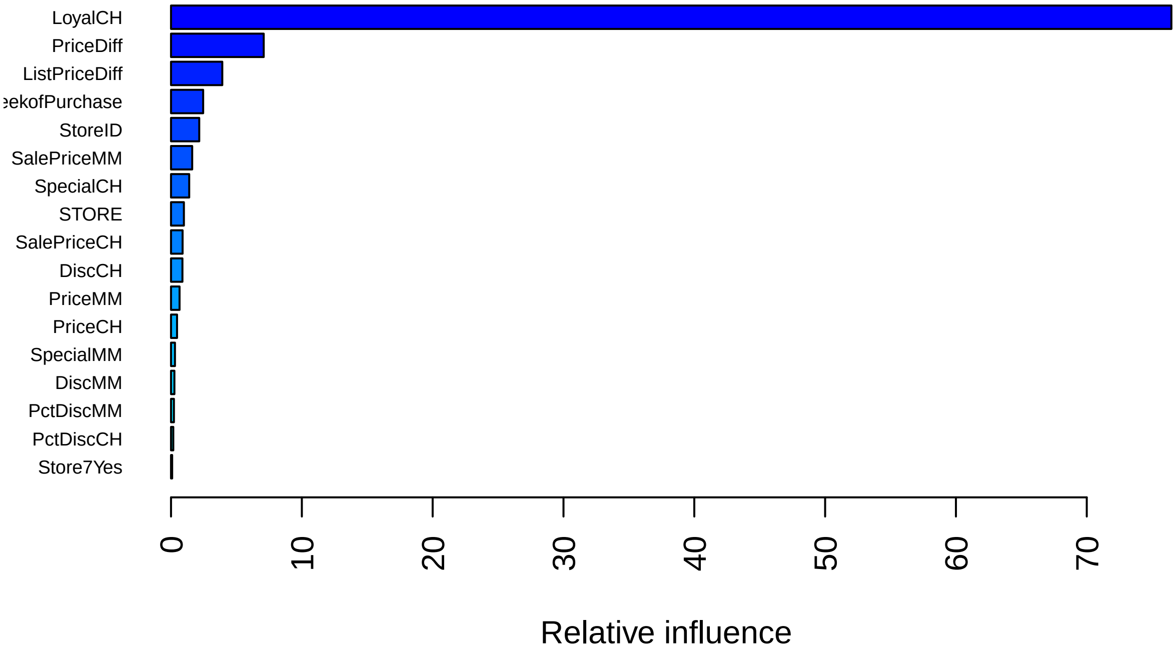
The top 5 most important variables: LoyalCH > StoreID > PriceDiff > ListPriceDiff > STORE

c) Boosting

```
set.seed(1)
gbm.grid = expand.grid(n.trees = seq(100, 600, by = 10),
                       interaction.depth = 2:6,
                       shrinkage = c(0.001, 0.003, 0.005),
                       n.minobsinnode = 1)

gbm_fit_c = train(Purchase~.,
                  data = training,
                  method = "gbm",
                  trControl = ctrl,
                  distribution = "bernoulli",
                  metric = "ROC",
                  tuneGrid = gbm.grid,
                  verbose = F
                  )

ggplot(gbm_fit_c, highlight = T)
```



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



```
##                             var     rel.inf
## LoyalCH                 LoyalCH 76.46936133
## PriceDiff             PriceDiff  7.08748777
## ListPriceDiff     ListPriceDiff  3.91872613
## WeekofPurchase WeekofPurchase  2.45739050
## StoreID                 StoreID  2.15767257
## SalePriceMM         SalePriceMM  1.61919790
## SpecialCH             SpecialCH  1.39132422
## STORE                     STORE  0.98430038
## SalePriceCH         SalePriceCH  0.88406034
## DiscCH                   DiscCH  0.87234595
## PriceMM                 PriceMM  0.65350459
## PriceCH                 PriceCH  0.46000519
## SpecialMM             SpecialMM  0.30142017
## DiscMM                   DiscMM  0.25856578
## PctDiscMM             PctDiscMM  0.22307879
## PctDiscCH             PctDiscCH  0.17375185
## Store7Yes             Store7Yes  0.08780651
```

```r
gbm.pred = predict(gbm_fit_c, newdata = test, type = "raw")

1 - sum(gbm.pred == test$Purchase) / length(test$Purchase)
```

```
## [1] 0.1814815
```

The test classification error rate is 18.15% for this boosting model.

The loyalCH is the most importance variable in this boosting model. The top 5 most important variables: LoyalCH > StoreID > PriceDiff > ListPriceDiff > STORE