Tree-based Methods

Hana Akbarnejad

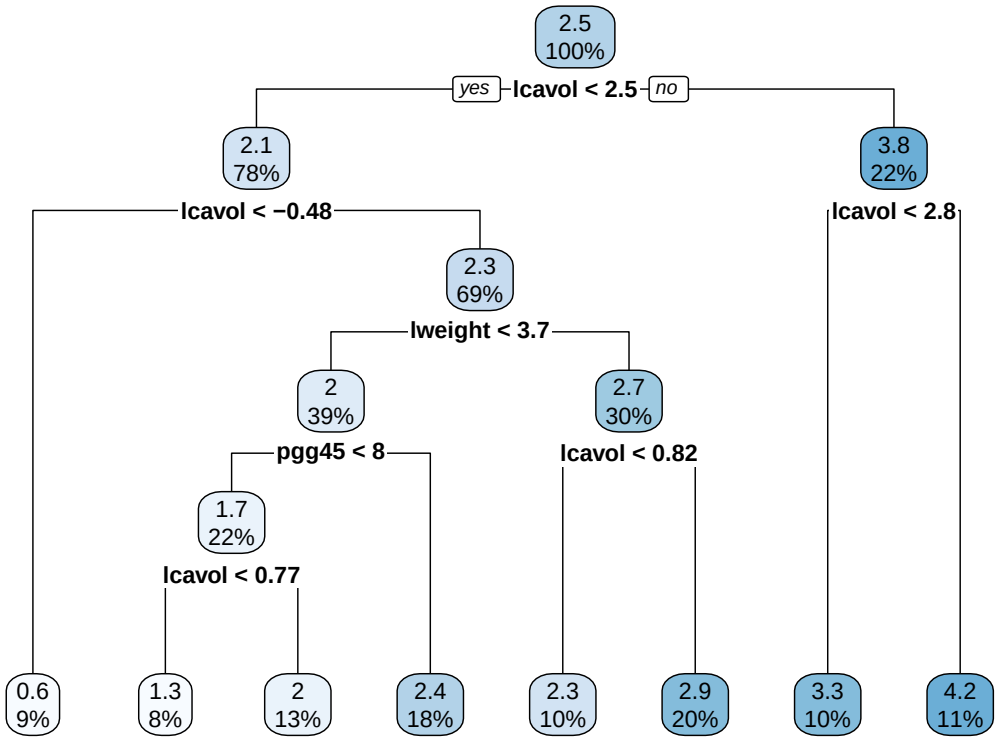
4/26/2020

Problem 1

a

Fit a regression tree with lpsa as the response and the other variables as predictors. 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?

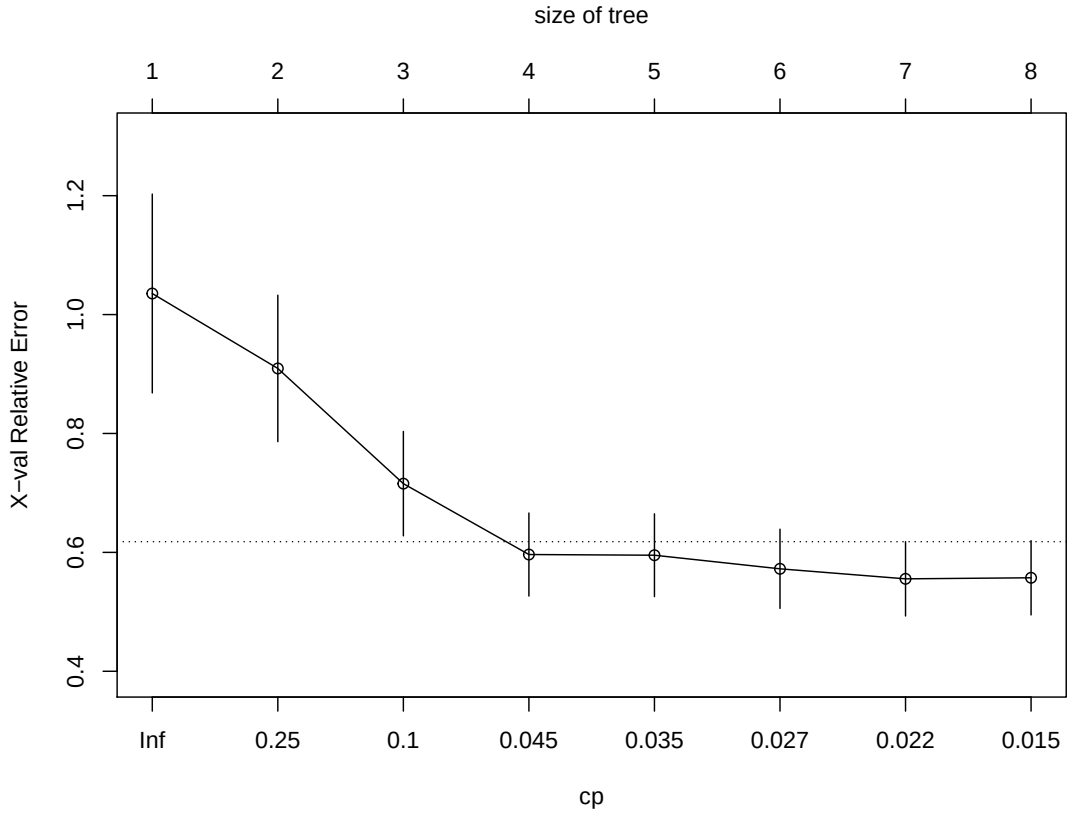
```
set.seed(2020)

# fitting initial tree
tree1 = rpart(formula = lpsa~., data = prostate_data)
rpart.plot(tree1)
```



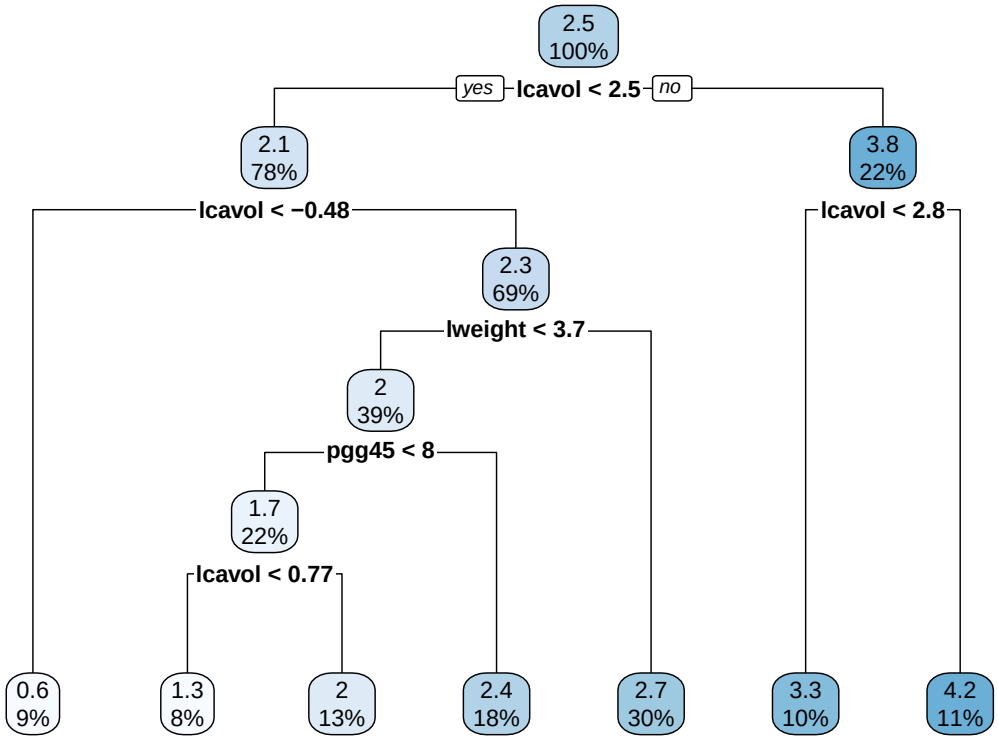
```r
# using CV to find optimal tree size
cp_table = printcp(tree1) # smallest error of 0.60554 corresponds to cp of  0.021470 which is tree size
```

```
##
## Regression tree:
## rpart(formula = lpsa ~ ., data = prostate_data)
##
## 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 1.03542 0.167254
## 2 0.184647      1   0.65289 0.90937 0.123206
## 3 0.059316      2   0.46824 0.71562 0.087849
## 4 0.034756      3   0.40893 0.59638 0.069851
## 5 0.034609      4   0.37417 0.59521 0.069659
## 6 0.021564      5   0.33956 0.57235 0.066634
## 7 0.021470      6   0.31800 0.55546 0.062470
## 8 0.010000      7   0.29653 0.55725 0.062470
```

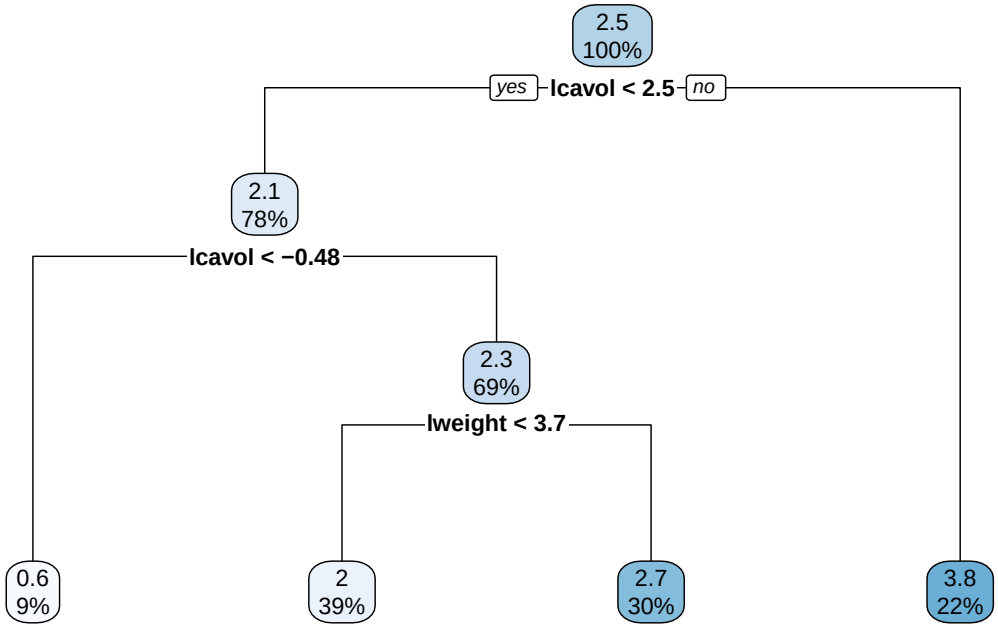
```r
plotcp(tree1)
```



```
#prune tree with obtained cp:
min_error = which.min(cp_table[,4])   # shows that minimum error belongs to tree size 7
# minimum cross-validation error
tree2 = prune(tree1, cp = cp_table[min_error,1])
rpart.plot(tree2)
```




```
# 1SE rule...
tree3 = prune(tree1, cp = cp_table[cp_table[,4]<cp_table[min_error,4]+cp_table[min_error,5],1][1])
rpart.plot(tree3) # 1SE rule shows tree size of 4 which is different from cp method
```



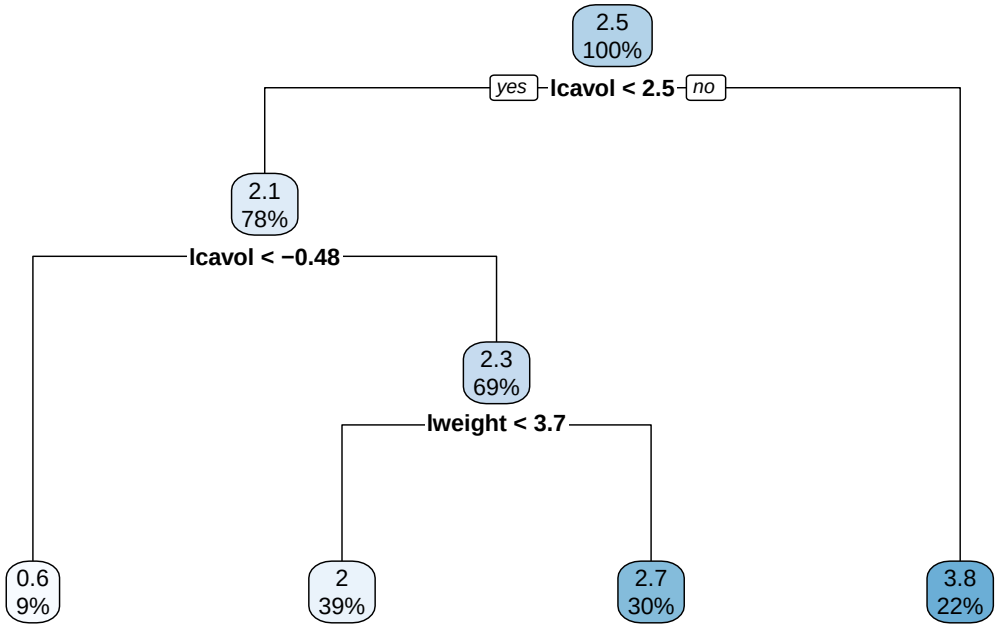
It can be observed that he tree obtained with the minimum CV error has different size compared to the tree that is obtained from the 1 SE rule. According to the minimum CV error, the optimal tree size is 7, while the size fo the tree obtained from 1 SE rule is 4.

b

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

Looking at the Cp plot created above, it can be observed that the file point below horizental line belongs to the cp of 0.045 and tree size 4 which is the same as what we got from 1 SE rule. So, I prune the tree using this cp value and choose it as my final tree. Chosing higher complexity results in a smaller and more interpretible tree.

```
set.seed(2020)
tree4 =  rpart(formula = lpsa~., data = prostate_data,    # refit it in caret
               control = rpart.control(cp = 0.045))
rpart.plot(tree4)
```



We can observe that there are four terminal nodes which have 9%, 39%, 30%, and 22% of observations, from left to right. The other number in terminal nodes contain the average response level of the observations that fall within tat node. For example, the righter most node contains 22% of observations with mean IPSA level of 3.8. The splitting steps of trees were based on lcavol and lweight values cut-offs.

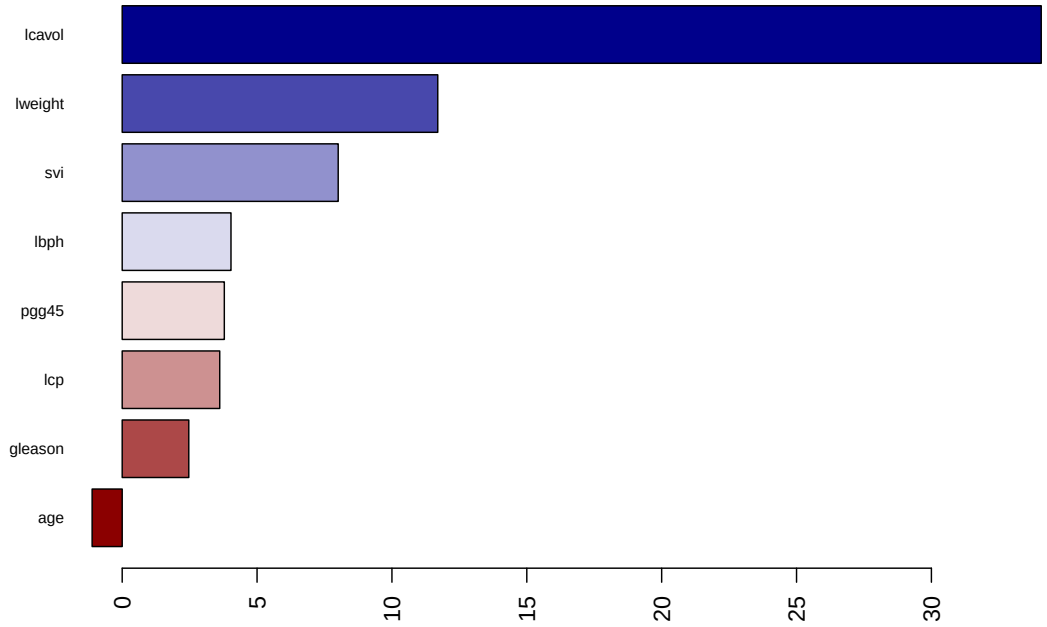
c

Perform bagging and report the variable importance.

```
set.seed(2020)
bagging = randomForest(lpsa ~ ., data = prostate_data, mtry = 8)
bagging2 = ranger(lpsa ~ ., data = prostate_data, mtry = 8,
             splitrule = "variance",
             importance = "permutation",
             scale.permutation.importance = TRUE)

importance(bagging2)
```

```
##    lcavol   lweight       age      lbph       svi       lcp   gleason     pgg45
## 34.071310 11.699317 -1.114481  4.032895  8.007999  3.616086  2.468427  3.786048
```

```
barplot(sort(ranger::importance(bagging2), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(8))
```



After performing Bagging method, the variable importance value can be observed above for each variable. Also, the above barplot shows that top three variables are *lcavol*, *lweight*, and *svi*.

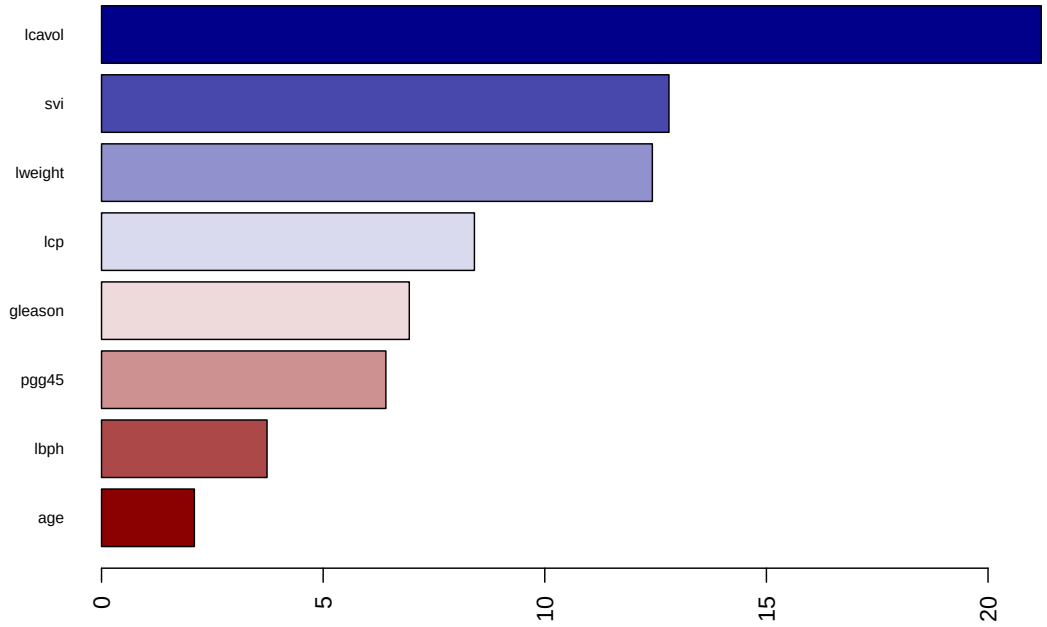
d

Perform random forests and report the variable importance.

```
set.seed(2020)
rf = randomForest(lpsa ~ ., data = prostate_data, mtry = 2)
rf2 = ranger(lpsa ~ ., data = prostate_data, mtry = 2, #mtry
             splitrule = "variance",
             importance = "permutation",
             scale.permutation.importance = TRUE)

importance(rf2)
```

```
##    lcavol   lweight       age      lbph       svi       lcp   gleason     pgg45
## 21.199916 12.427092  2.094390  3.734066 12.801809  8.412550  6.943221  6.414958
```

```
barplot(sort(ranger::importance(rf2), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(8))
```

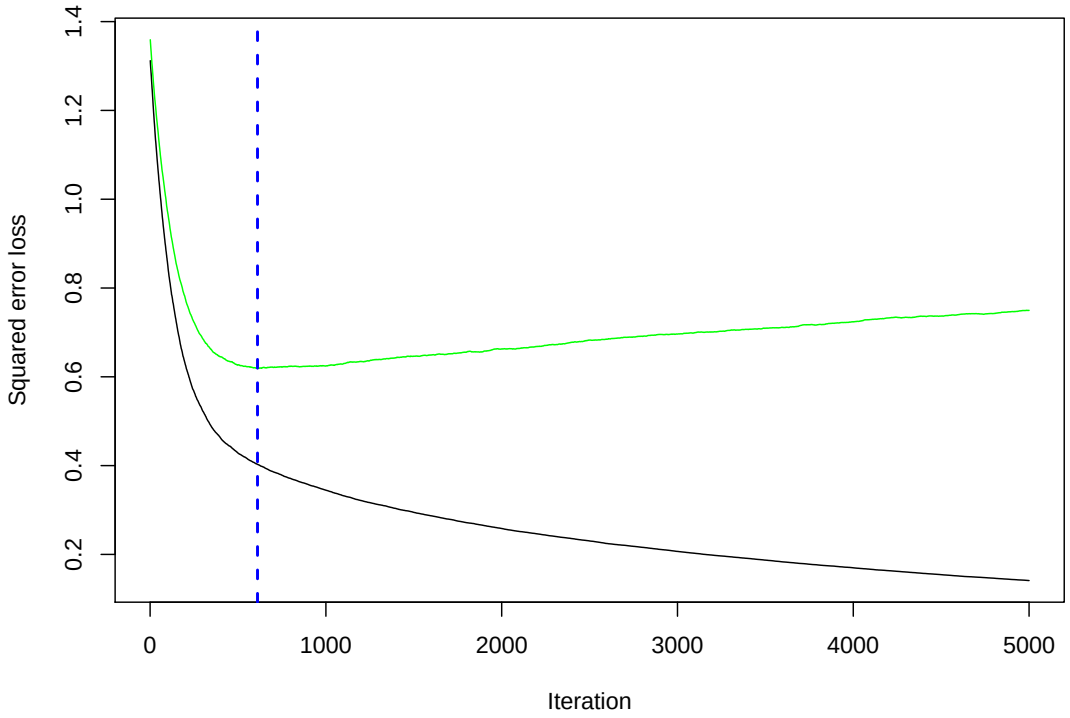


After performing Random Forests method, the variable importance value can be observed above for each variable. Also, the above barplot shows that top three variables are *lcavol*, *svi*, and *lweight*; the same variables as obtained through Bagging.

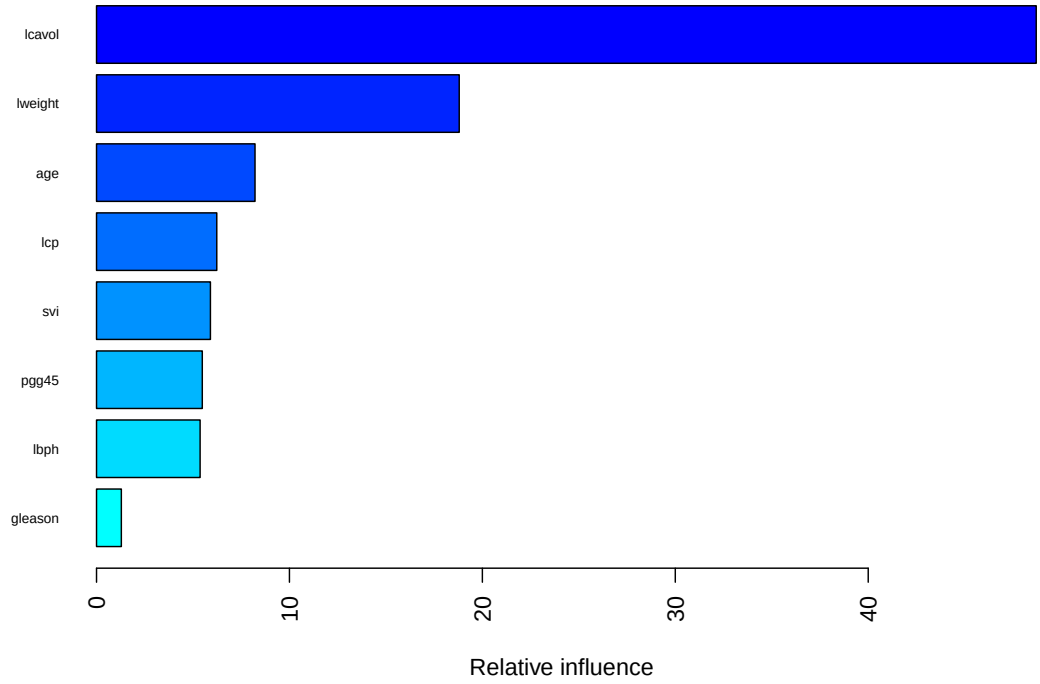
e

Perform boosting and report the variable importance.

```
set.seed(2020)
boosting = gbm(lpsa ~ ., prostate_data,
          distribution = "gaussian",
          n.trees = 5000,
          interaction.depth = 3,
          shrinkage = 0.005,
          cv.folds = 10)

nt = gbm.perf(boosting, method = "cv") # optimal number of trees
```




```r
# Grid search and find optimal tuning parameter
boosting_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(2020)
boosting_fit = train(lpsa ~ ., prostate_data,
                 method = "gbm",
                 tuneGrid = boosting_grid,
                 trControl = ctrl,
                 verbose = FALSE)

# summary of gbm from caret gives the variable importance for boosting
summary(boosting_fit$finalModel, las = 2, cBars = 8, cex.names = 0.6)
```




```
##             var   rel.inf
## lcavol   lcavol 48.709231
## lweight lweight 18.800391
## age         age  8.215881
## lcp         lcp  6.233042
## svi         svi  5.903586
## pgg45     pgg45  5.482422
## lbph       lbph  5.369109
## gleason gleason  1.286339
```

```
boosting_fit$finalModel$tuneValue
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 18    3000                10     0.001              1
```

After performing Boosting method, the variable importance value can be observed above for each variable. Also, the above barplot shows that top three variables are *lcavol*, *lweight*, and *svi*; the same variables as obtained through Bagging and Random Forests.

f

Which of the above models will you select to predict PSA level? Explain.

```r
# RF using caret
rf_grid = expand.grid(mtry = 1:7,
                      splitrule = "variance",
                      min.node.size = 1:15)
set.seed(2020)
rf_fit = train(lpsa ~ ., prostate_data,
               method = "ranger",
               tuneGrid = rf_grid,
               trControl = ctrl)

# bagging using caret
bagging_grid = expand.grid(mtry = 8,
                      splitrule = "variance",
                      min.node.size = 1:15)
set.seed(2020)
bagging_fit = train(lpsa ~ ., prostate_data,
               method = "ranger",
               tuneGrid = bagging_grid,
               trControl = ctrl)

resamp = resamples(list(RF = rf_fit, boosting = boosting_fit, bagging = bagging_fit))
summary(resamp) # boosting with Mean RMSE of 0.7341767 is has the lowest error
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: RF, boosting, bagging
## Number of resamples: 10
##
## MAE
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RF       0.4541196 0.5087609 0.6014615 0.6022065 0.6869817 0.7875705    0
## boosting 0.4396959 0.5188346 0.5925024 0.5978244 0.6818583 0.8061304    0
## bagging  0.4686512 0.5538388 0.6030797 0.6099779 0.6542961 0.8100561    0
##
## RMSE
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RF       0.5173723 0.6142811 0.7544033 0.7379605 0.8348934 0.9665076    0
## boosting 0.5219765 0.6579517 0.7464480 0.7392668 0.8274327 0.9936647    0
## bagging  0.5442065 0.6534052 0.7559904 0.7490233 0.8144530 0.9859971    0
##
## Rsquared
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RF       0.2699401 0.4446827 0.5855813 0.5758424 0.6374320 0.9125685    0
## boosting 0.3026998 0.4876535 0.5832761 0.5833689 0.7037340 0.8564062    0
## bagging  0.3042721 0.4474976 0.5507183 0.5611712 0.6268002 0.8732802    0
```

After comparing the three ensamble methods in caret and comapring them using resmples method, we can see that these methods have very close Mean RMSE, but RF and boosting show the smallest training error (0.74). So, when predicting PSA level, these are the best ensamble method we can choose.