Untitled

Yue Lai

5/16/2020

import data

```
red = read_excel(path = "./data/wine.xlsx", sheet = "red") %>%
  janitor::clean_names()
```

```
## Warning in FUN(X[[i]], ...): strings not representable in native encoding will
## be translated to UTF-8
```

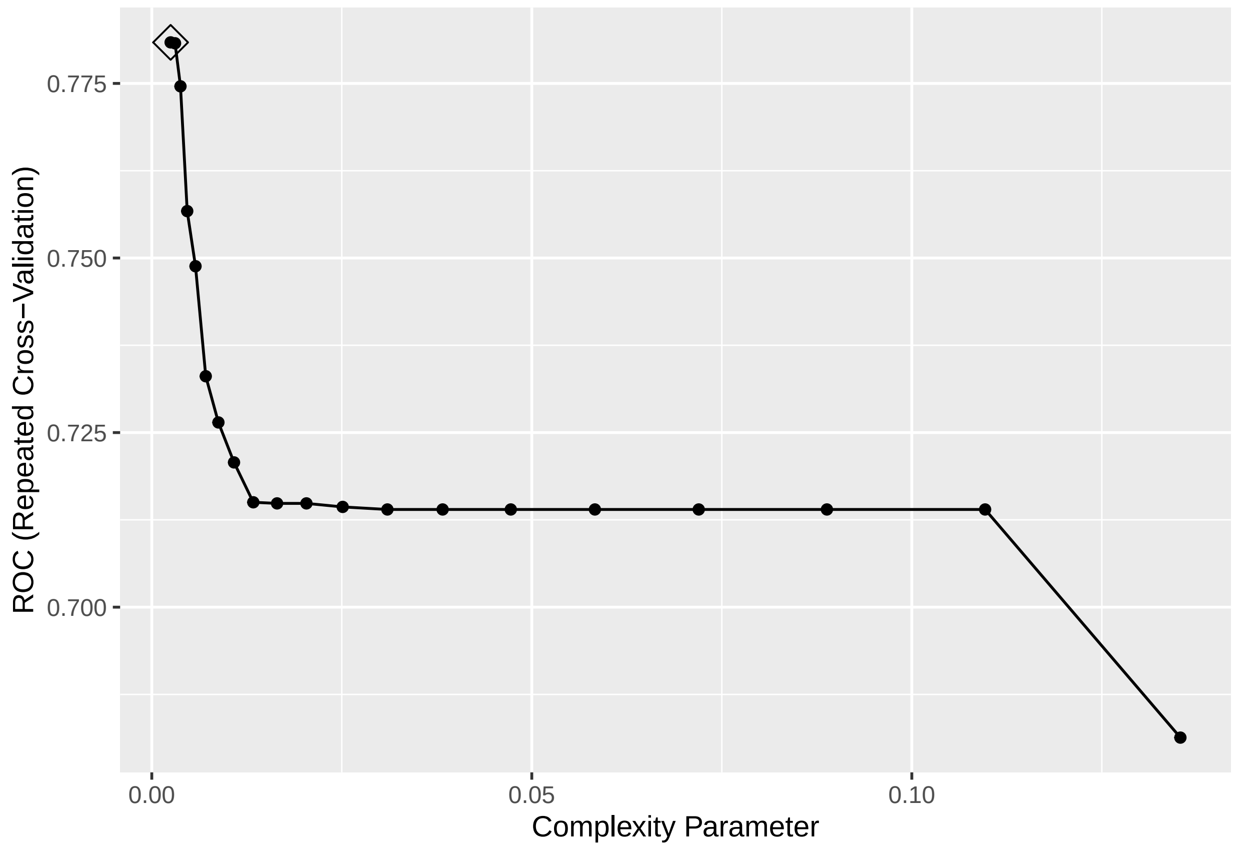
```
white = read_excel(path = "./data/wine.xlsx", sheet = "white") %>%
  janitor::clean_names()

wine = data.frame(rbind(red, white))

wine = wine %>%
  mutate(quality = as.factor(ifelse(quality > 5, "good", "bad")))
```

Divide the data into two part (training and test)

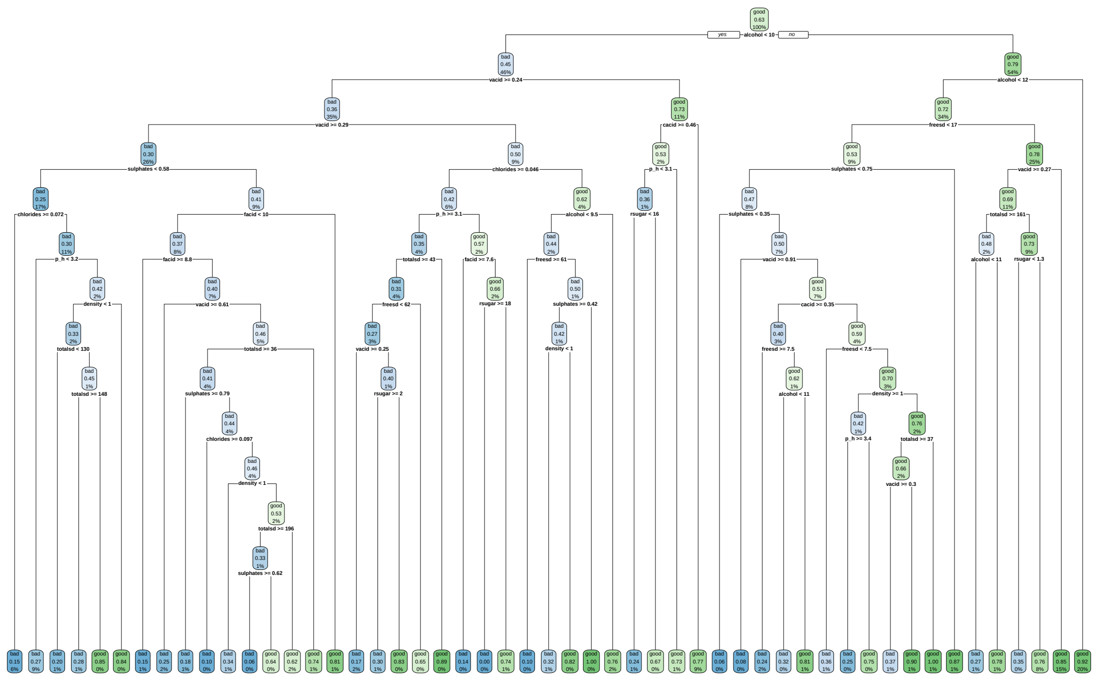
```
rowtrain = createDataPartition(y = wine$quality,
                               p = 2/3,
                               list = FALSE)

ctrl = trainControl(method = "repeatedcv",
                    repeats = 5,
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE)
```

tree

```
## using caret
rpart.fit = train(quality~., wine,
                  subset = rowtrain,
                  method = "rpart",
                  tuneGrid = data.frame(cp = exp(seq(-6, -2, len = 20))),
                  trControl = ctrl,
                  metric = "ROC")
ggplot(rpart.fit, highlight = TRUE)
```



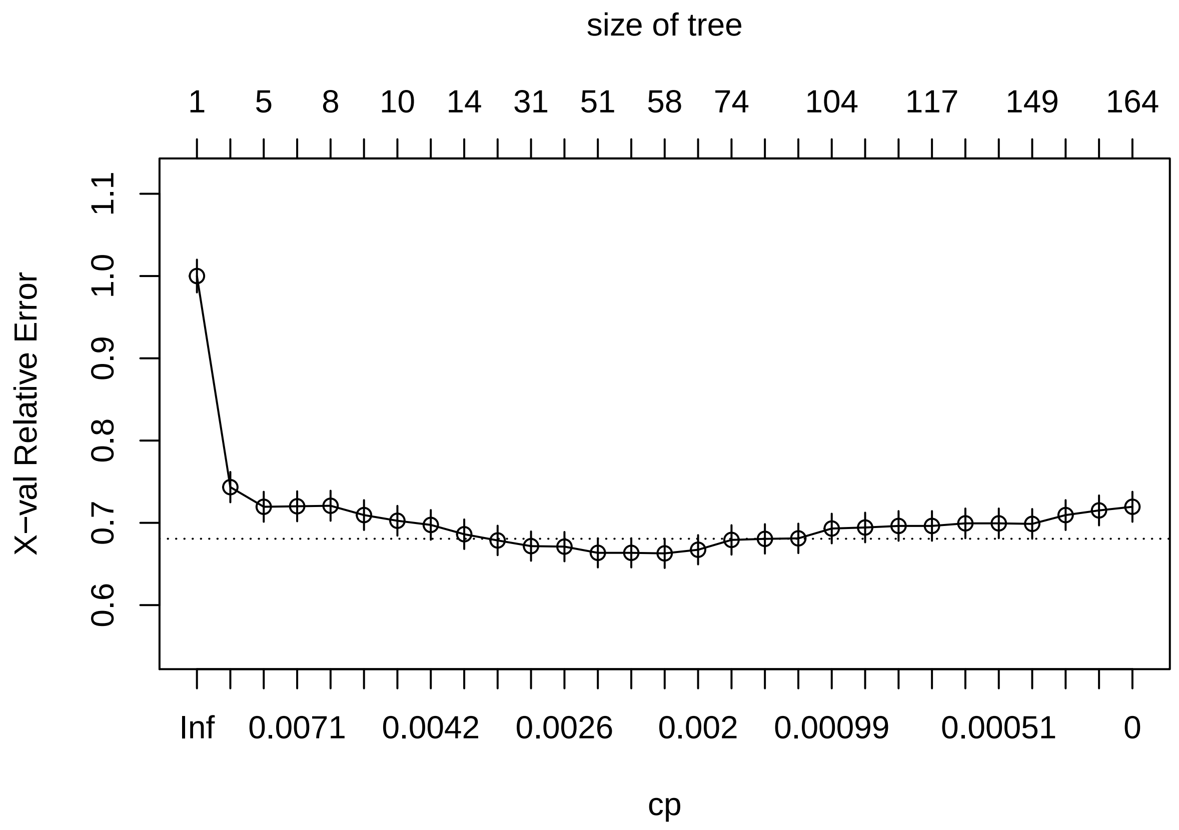
```r
rpart.plot(rpart.fit$finalModel)
```



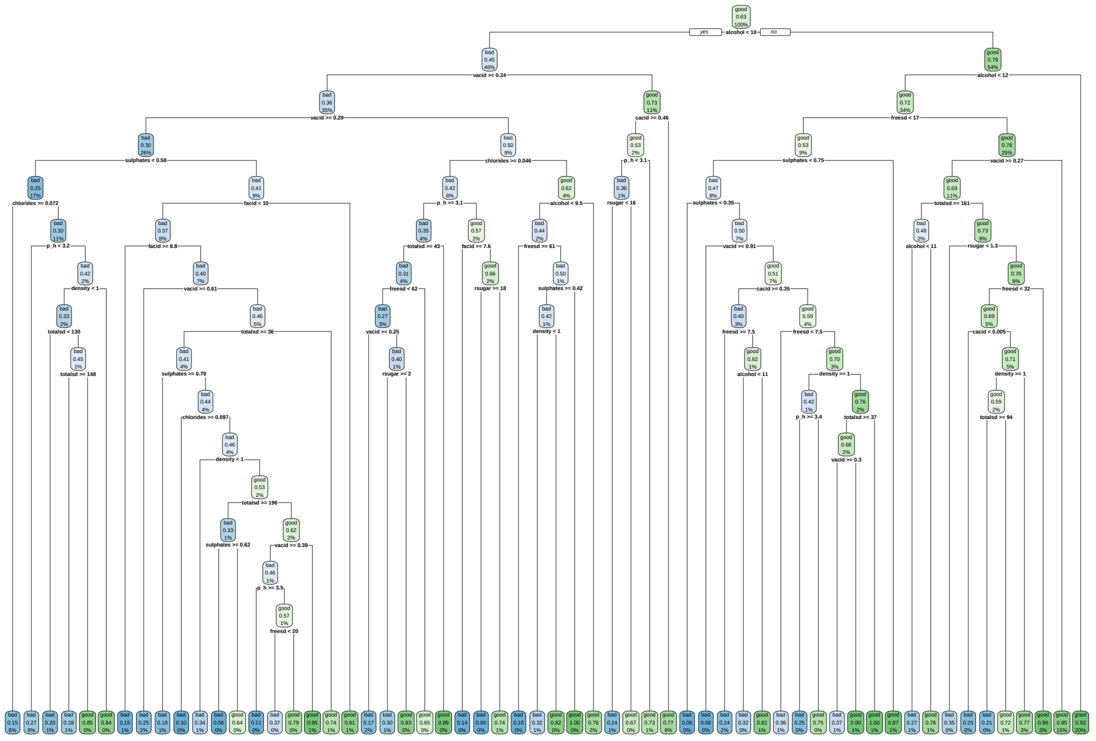
```r
library(rpart)
tree1 = rpart(formula = quality~., data = wine,
              subset = rowtrain,
              control = rpart.control(cp = 0))
cpTable = printcp(tree1)
```

```
##
## Classification tree:
## rpart(formula = quality ~ ., data = wine, subset = rowtrain,
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
##  [1] alcohol   cacid     chlorides density   facid     freesd    p_h
##  [8] rsugar    sulphates totalsd   vacid
##
## Root node error: 1590/4332 = 0.36704
##
## n= 4332
##
##             CP nsplit rel error  xerror     xstd
## 1  0.13710692      0   1.00000 1.00000 0.019952
## 2  0.01194969      2   0.72579 0.74340 0.018438
## 3  0.00723270      4   0.70189 0.71950 0.018249
## 4  0.00691824      6   0.68742 0.72013 0.018254
## 5  0.00628931      7   0.68050 0.72075 0.018259
## 6  0.00503145      8   0.67421 0.70943 0.018166
## 7  0.00440252      9   0.66918 0.70252 0.018108
## 8  0.00408805     11   0.66038 0.69748 0.018066
## 9  0.00356394     13   0.65220 0.68616 0.017968
## 10 0.00314465     29   0.56918 0.67862 0.017902
## 11 0.00272537     30   0.56604 0.67170 0.017841
## 12 0.00251572     35   0.55220 0.67107 0.017835
## 13 0.00235849     50   0.51258 0.66352 0.017767
## 14 0.00230608     54   0.50314 0.66352 0.017767
## 15 0.00220126     57   0.49623 0.66289 0.017762
## 16 0.00188679     61   0.48742 0.66730 0.017802
## 17 0.00157233     73   0.46352 0.67925 0.017908
## 18 0.00125786     89   0.43459 0.68050 0.017919
## 19 0.00104822    100   0.42075 0.68113 0.017925
## 20 0.00094340    103   0.41761 0.69308 0.018028
## 21 0.00083857    109   0.41195 0.69434 0.018039
## 22 0.00078616    112   0.40943 0.69623 0.018055
## 23 0.00075472    116   0.40629 0.69623 0.018055
## 24 0.00062893    126   0.39874 0.69937 0.018082
## 25 0.00041929    145   0.38428 0.69937 0.018082
## 26 0.00031447    148   0.38302 0.69874 0.018076
## 27 0.00020964    156   0.38050 0.70943 0.018166
## 28 0.00015723    159   0.37987 0.71509 0.018213
## 29 0.00000000    163   0.37925 0.71950 0.018249
```

```
plotcp(tree1)
```



```
minErr = which.min(cpTable[,4])

tree2 = prune(tree1, cp = cpTable[minErr, 1])
rpart.plot(tree2)
```



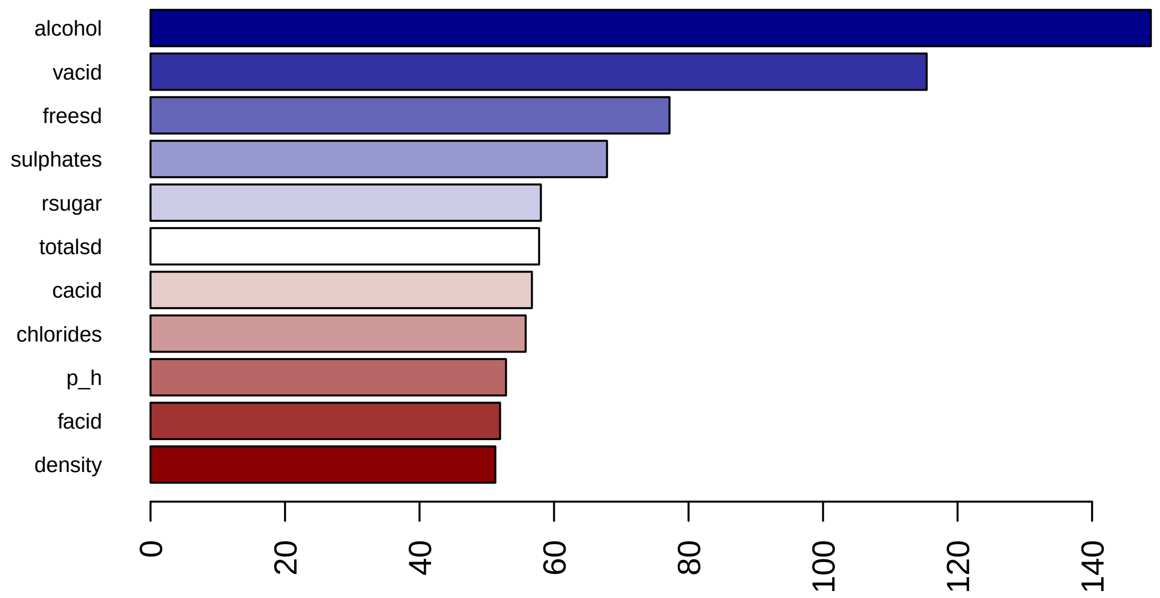
bagging and random forests

```
## using caret
#rf.grid = expand.grid(mtry = 1:6,
 #                       splitrule = "gini",
  #                      min.node.size = 1:6)

#rpart.fit = train(quality~., wine,
 #                   subset = rowtrain,
  #                  method = "ranger",
   #                 tuneGrid = rf.grid,
    #                trControl = ctrl,
     #               metric = "ROC")

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

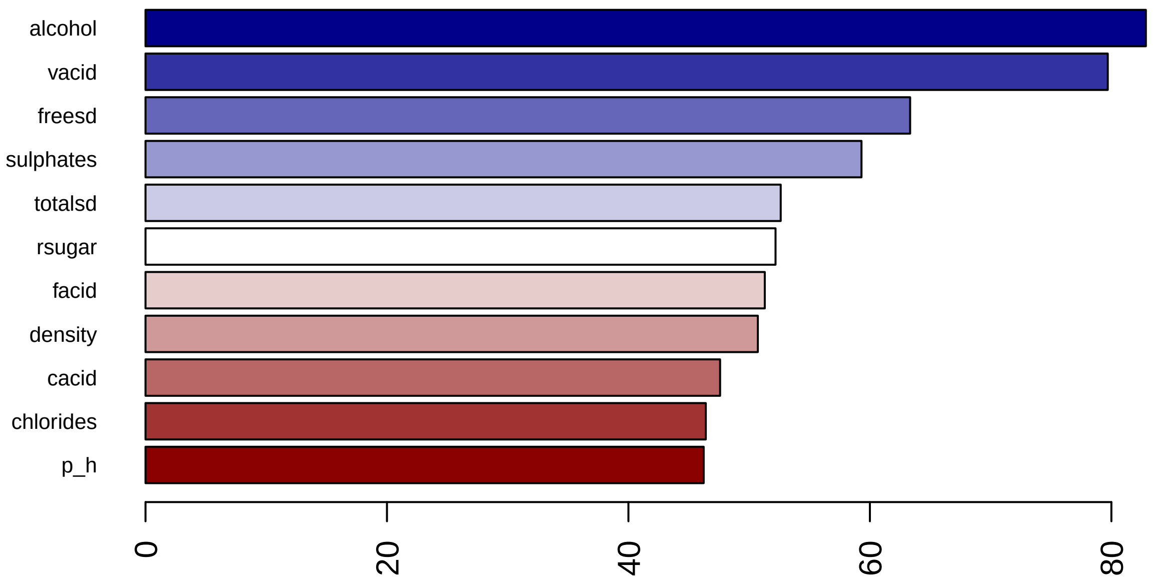
```r
library(randomForest)
```

```
## 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:ranger':
##
##     importance

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

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

```r
# bagging
bagging = randomForest(quality~., wine[rowtrain,],mtry = 11)

bagging.per = ranger(quality~., wine[rowtrain,],
                     mtry = 11,
                     splitrule = "gini",
                     min.node.size = 5,
                     importance = "permutation",
                     scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(bagging.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white", "darkblue"))(11))
```



```r
# random forest
rf = randomForest(quality~., wine[rowtrain,],mtry = 3)

rf.per = ranger(quality~., wine[rowtrain,],
                     mtry = 3,
                     splitrule = "gini",
                     min.node.size = 5,
                     importance = "permutation",
                     scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white", "darkblue"))(11))
```



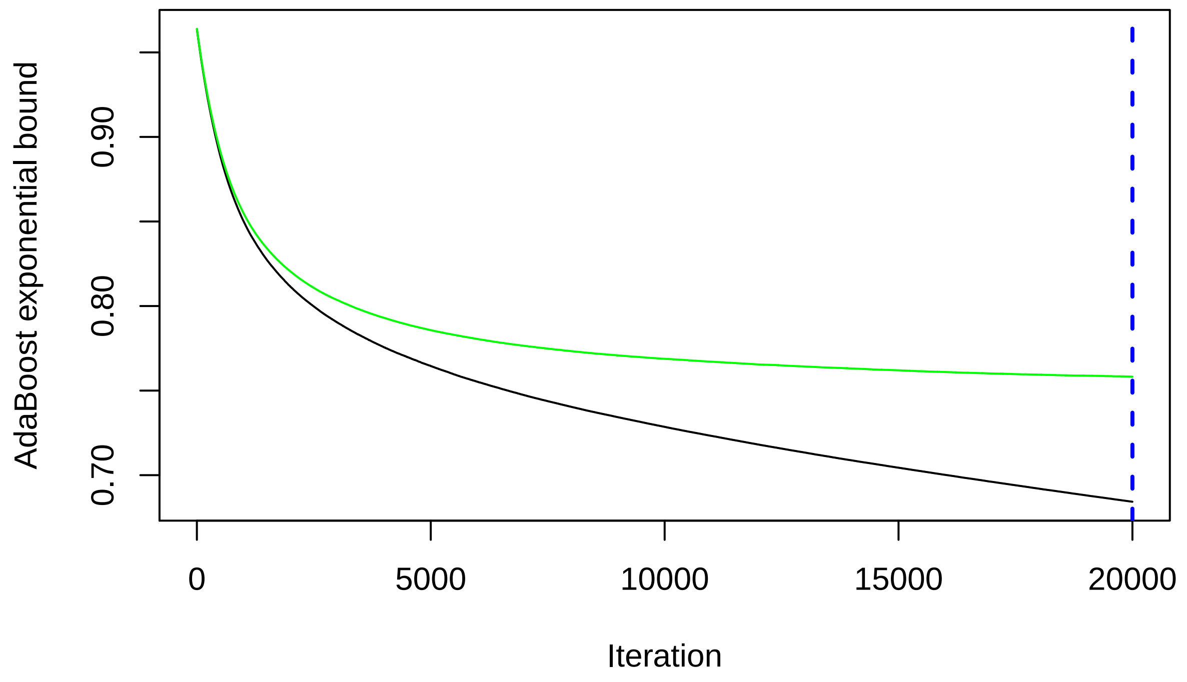
boosting

```
library(gbm)
```

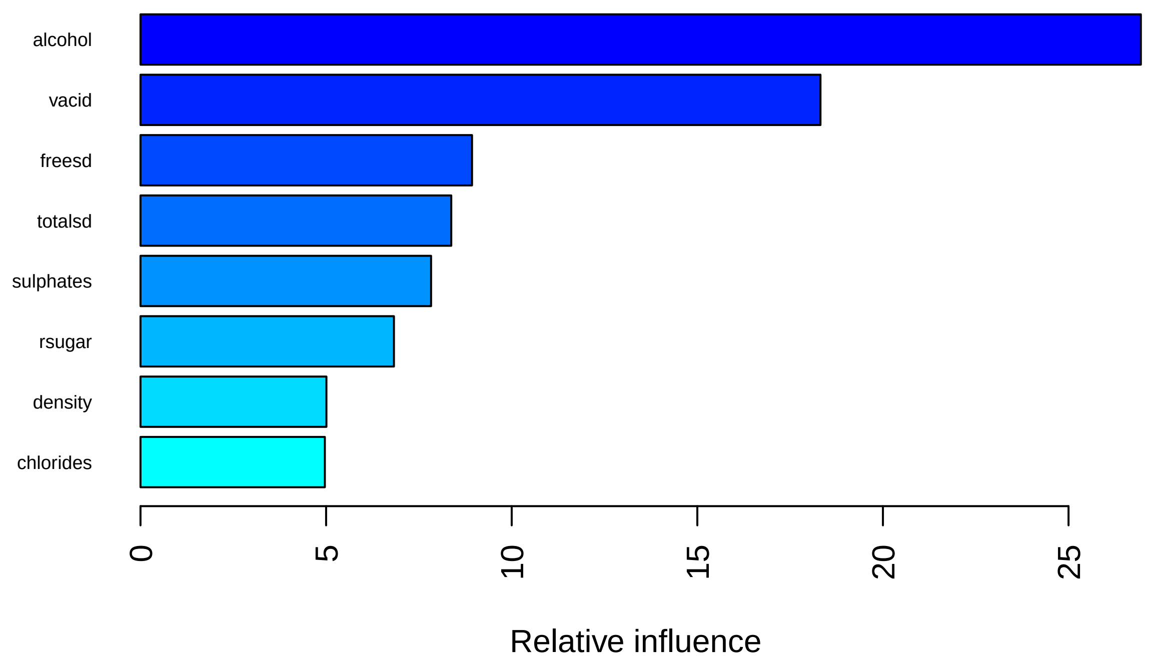
```
## Warning: package 'gbm' was built under R version 3.6.3
```

```
## Loaded gbm 2.1.5
```

```
wine2 = wine
wine2$quality = as.numeric(wine2$quality == "good")
bst = gbm(quality~., wine2[rowtrain,],
          distribution = "adaboost",
          n.trees = 20000,
          interaction.depth = 3,
          shrinkage = 0.001,
          cv.folds = 10)
nt = gbm.perf(bst, method = "cv")
```



```r
summary(bst, las = 2, cBars = 8, cex.names = 0.6)
```



```
##                 var   rel.inf
## alcohol     alcohol 26.948308
## vacid         vacid 18.317838
## freesd       freesd  8.930802
## totalsd     totalsd  8.371572
## sulphates sulphates  7.827669
## rsugar       rsugar  6.827096
## density     density  5.008816
## chlorides chlorides  4.966148
## cacid         cacid  4.876538
## facid         facid  4.002520
## p_h             p_h  3.922693
```

boosting in train

```
## (long time to run )
#gbm.grid_2 <- expand.grid(n.trees = c(2000,3000,4000),
 #                  interaction.depth = 1:6,
#                  shrinkage = c(0.001, 0.003,0.005),
 #                n.minobsinnode = 1)
#gbm.fit_2 <- train(quality~., wine[rowtrain,],
 #                  method = "gbm",
  #                 tuneGrid = gbm.grid_2,
   #               trControl = ctrl,
```

```r
    #                metric = "ROC",
     #             verbose = FALSE)
#ggplot(gbm.fit_2, highlight = TRUE)
```

```r
tree.pred = predict(tree2, newdata = wine[-rowtrain,], type = "prob")[,1]
bag.pred = predict(bagging, newdata = wine[-rowtrain,], type = "prob")[,1]
rf.pred = predict(rf, newdata = wine[-rowtrain,], type = "prob")[,1]

bst.pred = predict(bst, newdata = wine[-rowtrain,], type = "response")
```

```
## Using 20000 trees...
```

```r
roc.tree = roc(wine$quality[-rowtrain], tree.pred)
```

```
## Setting levels: control = bad, case = good
```

```
## Setting direction: controls > cases
```

```r
roc.bag = roc(wine$quality[-rowtrain], bag.pred)
```

```
## Setting levels: control = bad, case = good
## Setting direction: controls > cases
```

```r
roc.rf = roc(wine$quality[-rowtrain], rf.pred)
```

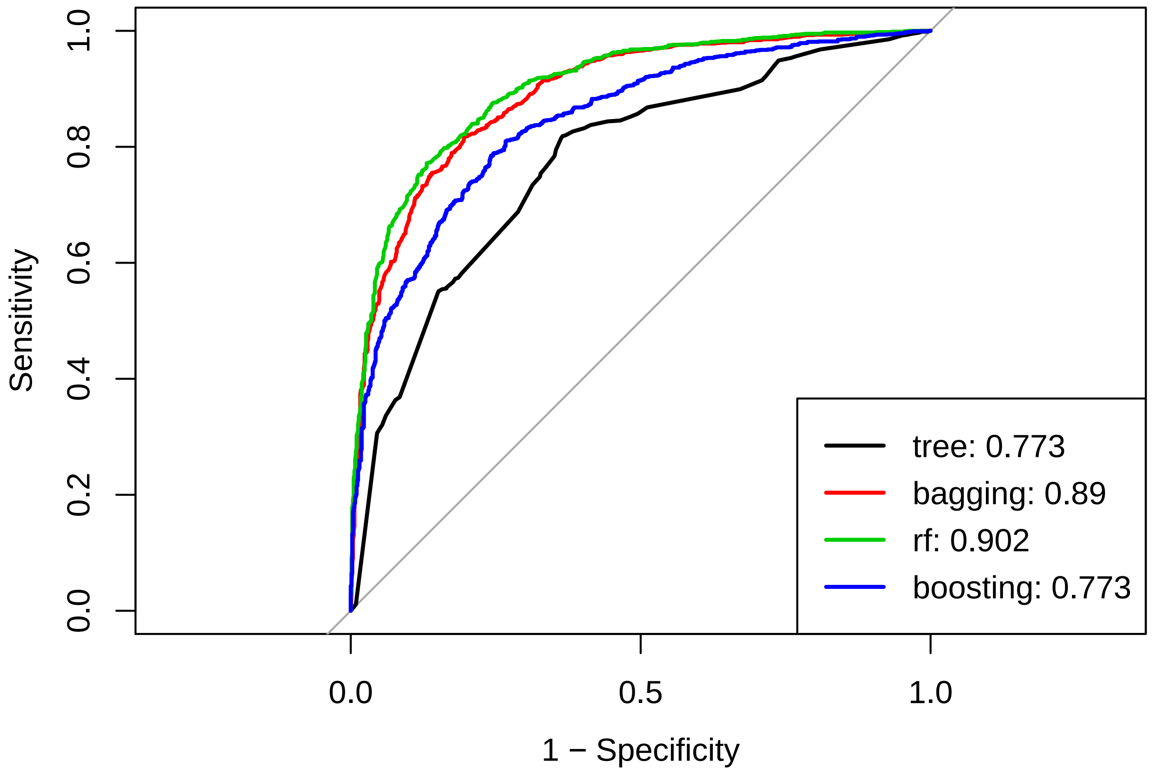
```
## Setting levels: control = bad, case = good
## Setting direction: controls > cases
```

```r
roc.bst = roc(wine$quality[-rowtrain], bst.pred)
```

```
## Setting levels: control = bad, case = good
```

```
## Setting direction: controls < cases
```

```r
auc = c(roc.tree$auc[1], roc.bag$auc[1], roc.rf$auc[1], roc.bst$aur[1])

plot(roc.tree, legacy.axes = TRUE)
plot(roc.bag, col = 2, add = TRUE)
plot(roc.rf, col = 3, add = TRUE)
plot(roc.bst, col = 4, add = TRUE)

modelNames = c("tree", "bagging", "rf", "boosting")

legend("bottomright", legend = paste0(modelNames, ": ", round(auc, 3)),
       col = 1:4, lwd = 2)
```



Linear

```
linear_svm <- tune.svm(quality~.,
                  data = wine[rowtrain,],
                  kernal = "linear",
                  cost = data.frame(cost = exp(seq(-20,-15,len=50))))
linear_svm$best.parameters
```

```
##             cost
## 1 2.061154e-09
```

Radial

```
radi_svm <- tune.svm(quality~.,
                  data = wine[rowtrain,],
                  kernal = "radial",
                  cost = data.frame(cost = exp(seq(-20,-15,len=50))))
radi_svm$best.parameters
```

```
##             cost
## 1 2.061154e-09
```

only error rate can be obtained

```
linea.svm.pred = predict(linear_svm$best.model, newdata = wine[-rowtrain,],type = "prob")
radial.svm.pred = predict(radi_svm$best.model, newdata = wine[-rowtrain,], type = "prob")

error_linear=1-sum(linea.svm.pred==wine[-rowtrain,12])/nrow(wine[-rowtrain,])
error_radial=1-sum(radial.svm.pred==wine[-rowtrain,12])/nrow(wine[-rowtrain,])

tree.pred2 = predict(tree2, newdata = wine[-rowtrain,])
bag.pred2 = predict(bagging, newdata = wine[-rowtrain,])
rf.pred2 = predict(rf, newdata = wine[-rowtrain,])

error_tree=1-sum(tree.pred2 ==wine[-rowtrain,12])/nrow(wine[-rowtrain,])
error_bag=1-sum(bag.pred2==wine[-rowtrain,12])/nrow(wine[-rowtrain,])
error_rf=1-sum(rf.pred2==wine[-rowtrain,12])/nrow(wine[-rowtrain,])

data.frame(model=c("SVM_linear","SVM_radial","Tree","Bagging","Random_forest"),error_rate=c(error_linea
```

model	error_rate
SVM_linear	0.3667436
SVM_radial	0.3667436
Tree	1.0000000
Bagging	0.1852194
Random_forest	0.1704388