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 = red %>%
  mutate(quality = as.factor(ifelse(quality > 6.5, "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

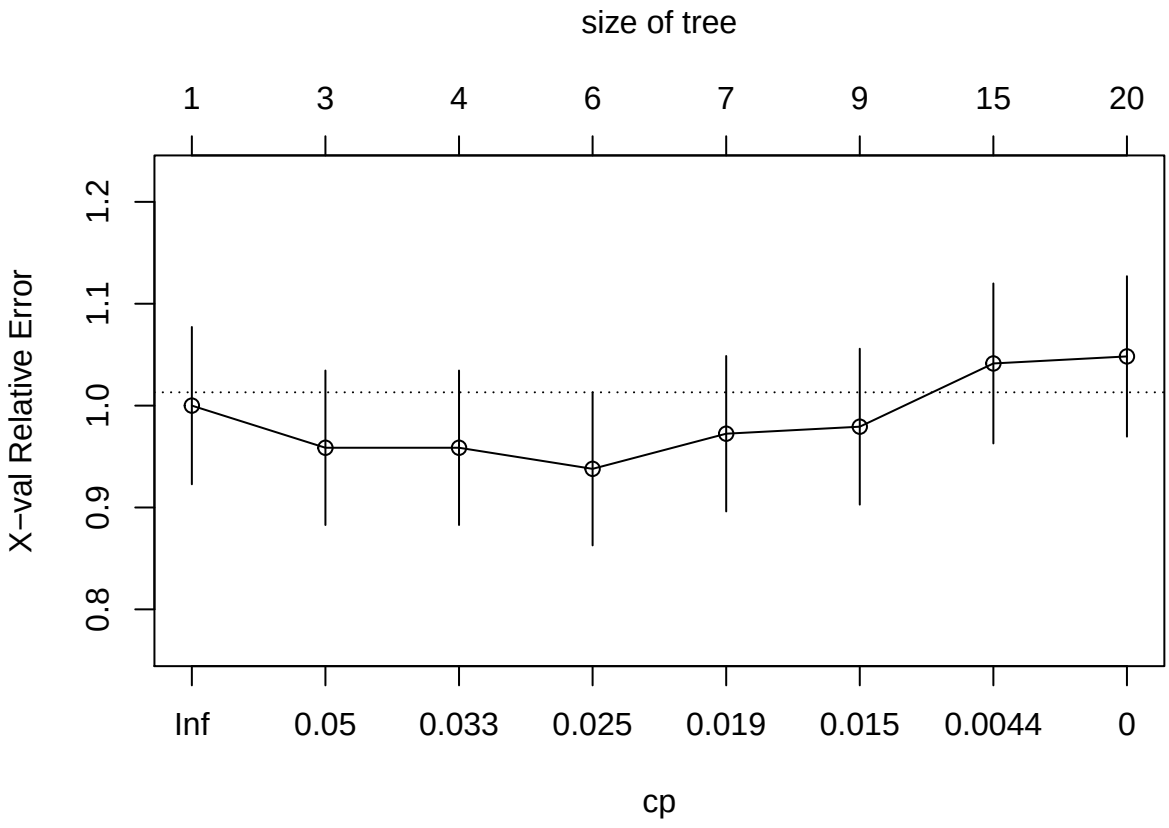
```
## using caret
#rpart.fit = train(quality~., wine,
 #                 subset = rowtrain,
  #                method = "rpart",
   #               tuneGrid = data.frame(cp = exp(seq(-6, -2, len = 20))),
    #              trControl = ctrl,
```

```
    #             metric = "ROC")
#ggplot(rpart.fit, highlight = TRUE)
#rpart.plot(rpart.fit$finalModel)
#rpart.fit$bestTune
```

```
set.seed(666)
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   chlorides facid     freesd    p_h       rsugar    sulphates
## [8] totalsd   vacid
##
## Root node error: 145/1067 = 0.1359
##
## n= 1067
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0724138      0   1.00000 1.00000 0.077197
## 2 0.0344828      2   0.85517 0.95862 0.075828
## 3 0.0310345      3   0.82069 0.95862 0.075828
## 4 0.0206897      5   0.75862 0.93793 0.075127
## 5 0.0172414      6   0.73793 0.97241 0.076290
## 6 0.0137931      8   0.70345 0.97931 0.076518
## 7 0.0013793     14   0.60000 1.04138 0.078521
## 8 0.0000000     19   0.59310 1.04828 0.078738
```

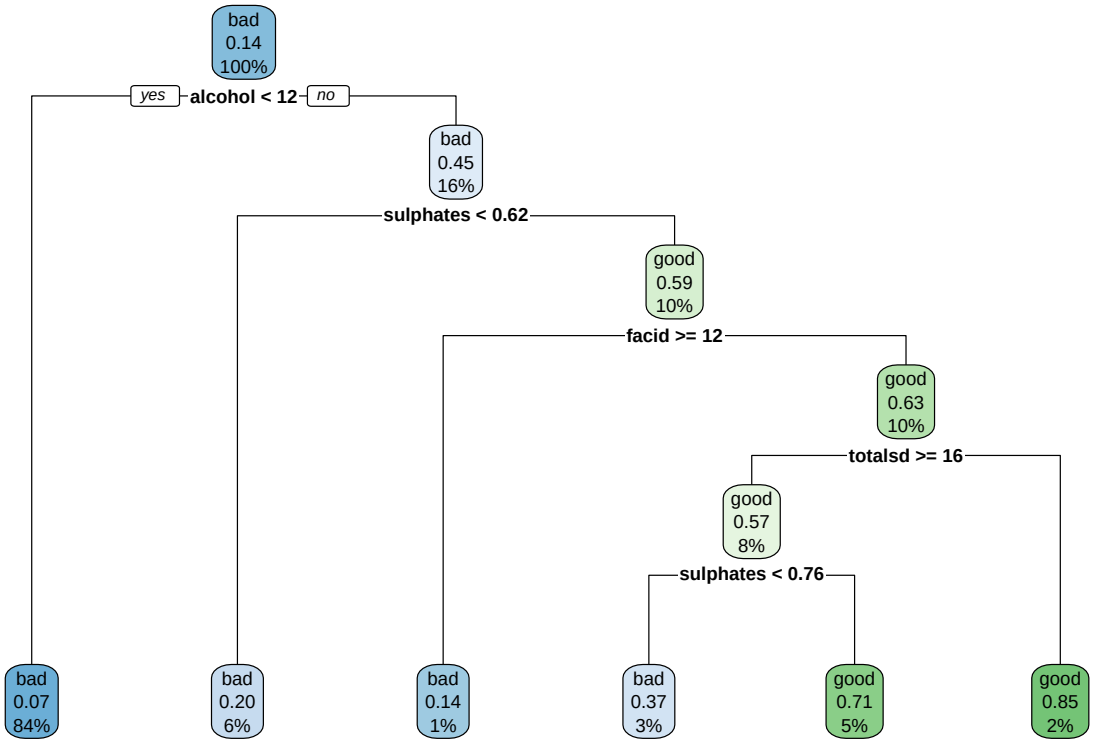
```
plotcp(tree1)
```




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

# optimal cp value
cpTable[minErr, 1]
```

```
## [1] 0.02068966
```

```
tree2 = prune(tree1, cp = cpTable[minErr, 1])# mse
tree3=prune(tree1,cp=cpTable[cpTable[,4]<cpTable[minErr,4]+cpTable[minErr,5],1][1])
# mse
rpart.plot(tree2)
```




```
# 1se
rpart.plot(tree3)
```

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)
```

```
library(randomForest)
```

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

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

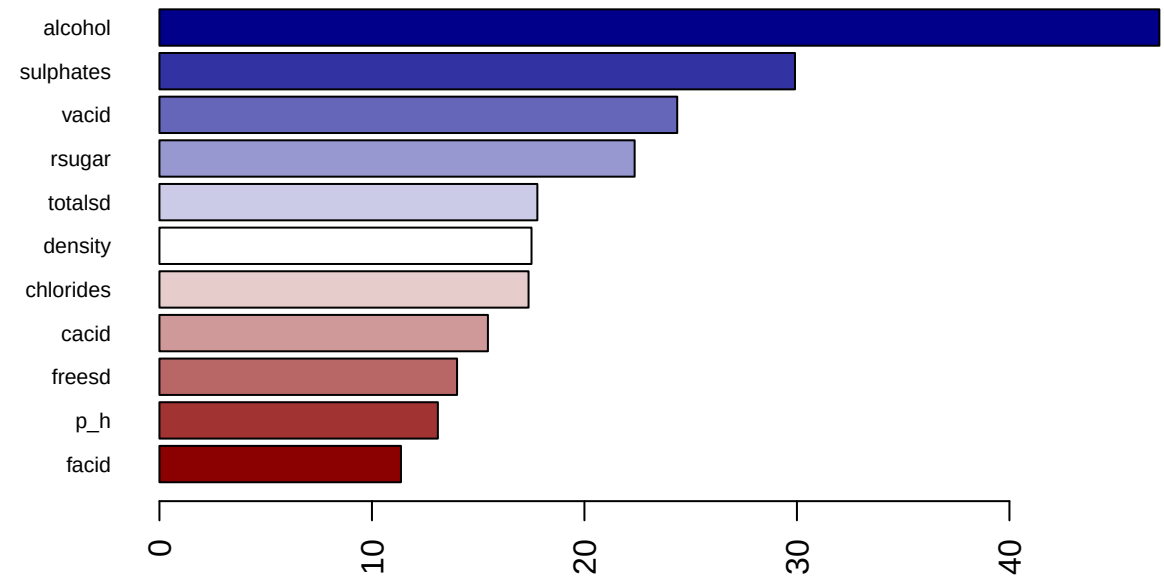
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## 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
```

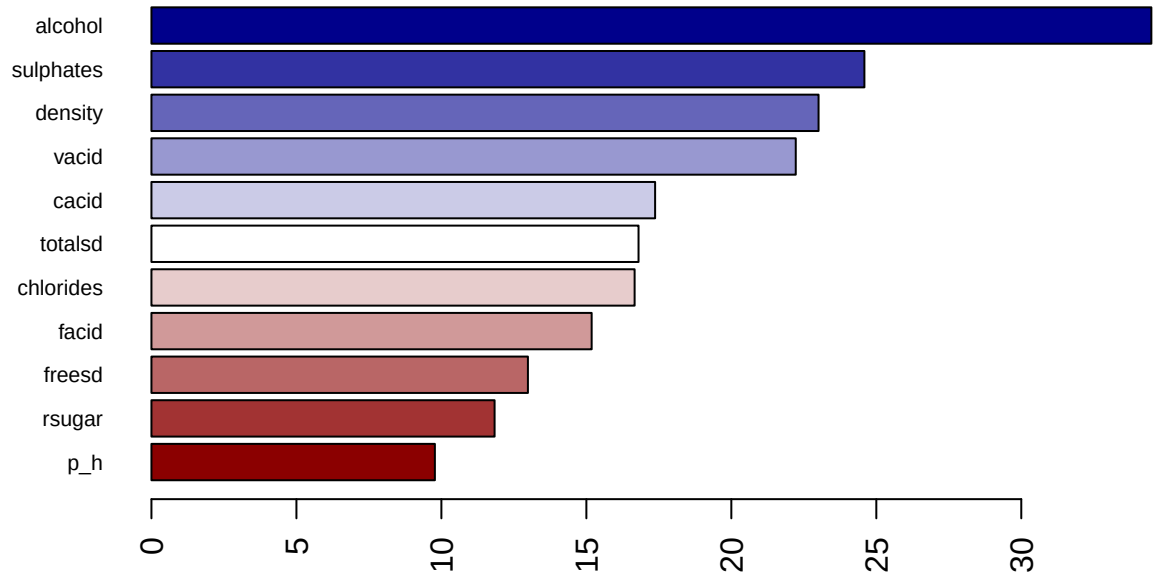
```
# bagging
bagging = randomForest(quality~., wine[rowtrain,],mtry = 11)
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
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))
```




```
# 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))
```



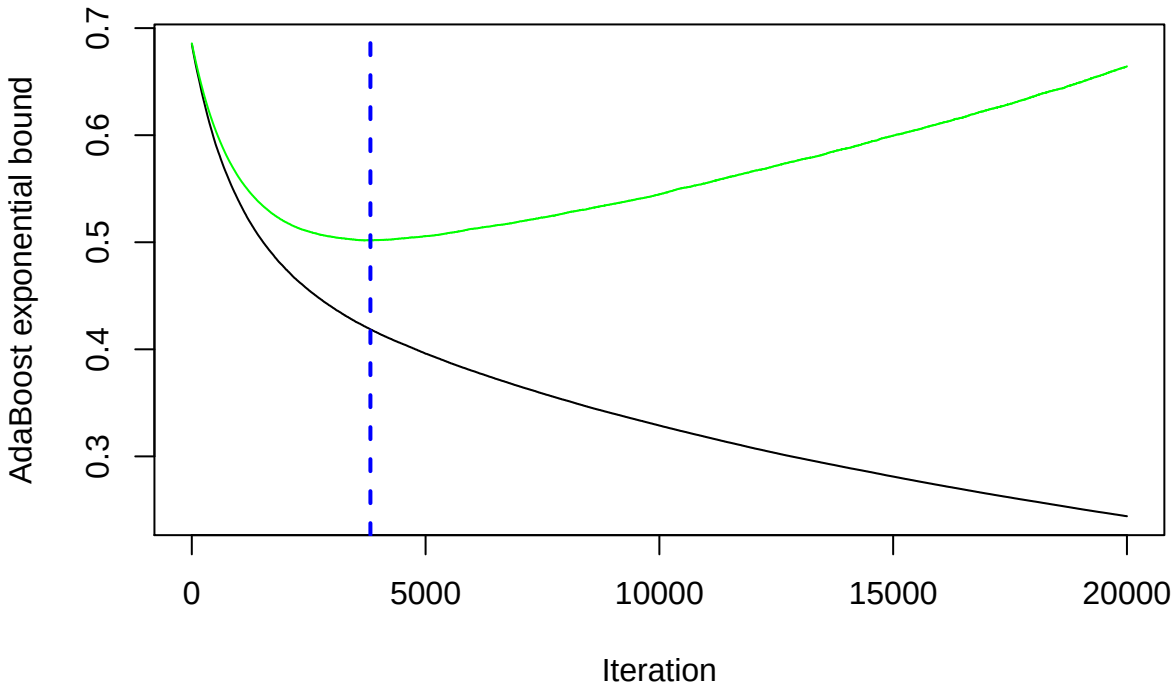
boosting

```
library(gbm)
```

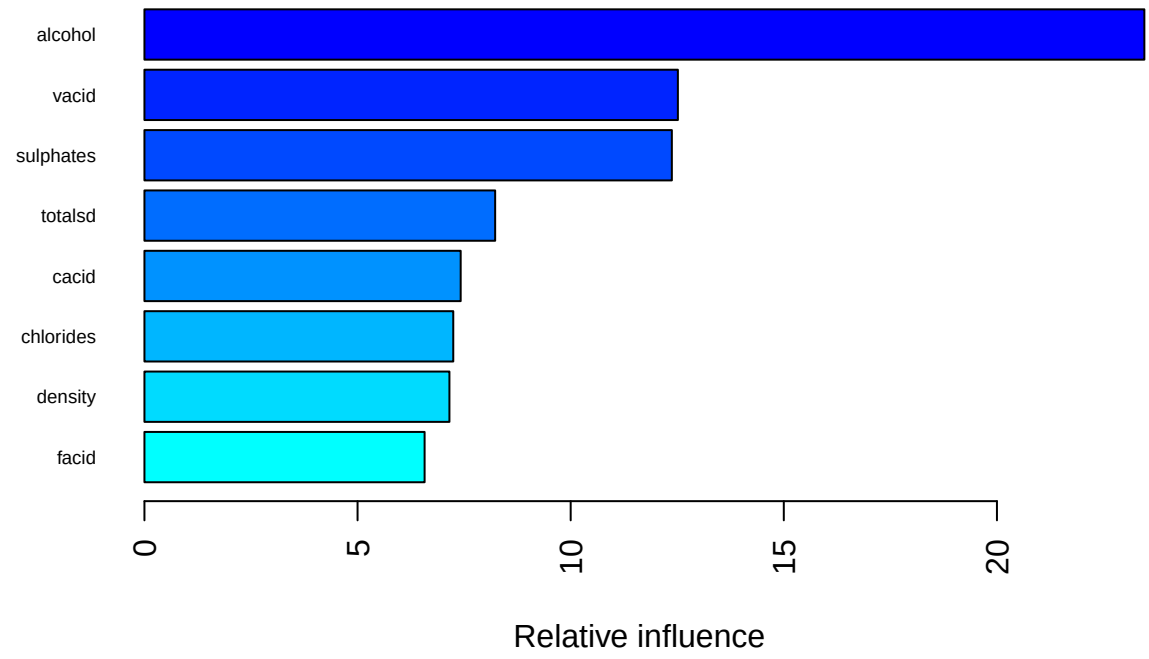
```
## Warning: package 'gbm' was built under R version 3.6.3
```

```
## 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")
```




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




```
##                 var   rel.inf
## alcohol     alcohol 23.458682
## vacid         vacid 12.514496
## sulphates sulphates 12.373228
## totalsd     totalsd  8.228050
## cacid         cacid  7.419293
## chlorides chlorides  7.245257
## density     density  7.154903
## facid         facid  6.572538
## rsugar       rsugar  5.684333
## freesd       freesd  4.967191
## p_h            p_h  4.382029
```

boosting in train

```
## (long time to run )
#gbm.grid_2 <- expand.grid(n.trees = 4586,
 #                  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,
```

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

```
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 3820 trees...
```

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

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

## Setting direction: controls < cases
```

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

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

## Setting direction: controls > cases
```

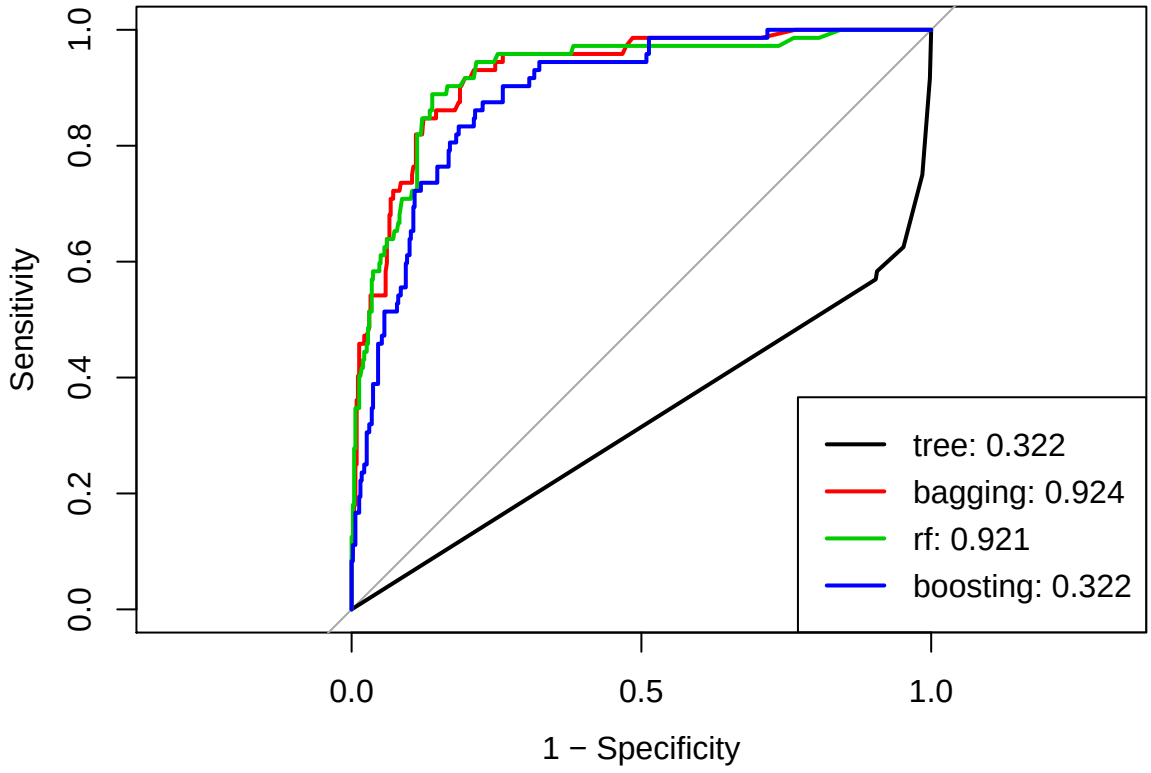
```
roc.rf = roc(wine$quality[-rowtrain], rf.pred)
```

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

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

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

## Setting direction: controls < cases
```

```
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$best.model, newdata = wine[-rowtrain,],type = "prob")%>% as.data.fra
radial.svm.pred = predict(radi_svm$best.model, newdata = wine[-rowtrain,], type = "prob")%>% as.data.fra

error_linear=1-sum(linea.svm.pred==as.vector(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,],type="class") %>% as.data.frame()
bag.pred2 = predict(bagging, newdata = wine[-rowtrain,],type="class")%>% as.data.frame()
rf.pred2 = predict(rf, newdata = wine[-rowtrain,],type="class")%>% as.data.frame()
bst.pred2 = predict(bst, newdata = wine[-rowtrain,], type = "response") %>% as.data.frame() %>% mutate(
  .<0.5~"bad",
  .>0.5~"good"
))
```

```
## Using 3820 trees...
```

```
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,])
error_bst=1-sum(bst.pred2[,2]==wine[-rowtrain,12])/nrow(wine[-rowtrain,])

# test misclassification
data.frame(model=c("SVM_linear","SVM_radial","Tree","Bagging","Random_forest","Boosting"),error_rate=c(
```

model	error_rate
SVM_linear	0.135
SVM_radial	0.135
Tree	0.115
Bagging	0.094
Random_forest	0.096
Boosting	0.118

```
# train data misclassification rate
linea.svm.pred3 = predict(linear_svm$best.model, newdata = wine[rowtrain,],type = "prob")%>% as.data.fra
radial.svm.pred3 = predict(radi_svm$best.model, newdata = wine[rowtrain,], type = "prob")%>% as.data.fra

error_linear3=1-sum(linea.svm.pred3==wine[rowtrain,12])/nrow(wine[rowtrain,])
error_radial3=1-sum(radial.svm.pred3==wine[rowtrain,12])/nrow(wine[rowtrain,])

tree.pred3 = predict(tree2, newdata = wine[rowtrain,],type="class") %>% as.data.frame()
bag.pred3 = predict(bagging, newdata = wine[rowtrain,],type="class")%>% as.data.frame()
rf.pred3 = predict(rf, newdata = wine[rowtrain,],type="class")%>% as.data.frame()
bst.pred3 = predict(bst, newdata = wine[rowtrain,], type = "response") %>% as.data.frame() %>% mutate(la
  .<0.5~"bad",
  .>0.5~"good"
))
```

```
## Using 3820 trees...
```

```r
error_tree3=1-sum(tree.pred3 ==wine[rowtrain,12])/nrow(wine[rowtrain,])
error_bag3=1-sum(bag.pred3==wine[rowtrain,12])/nrow(wine[rowtrain,])
error_rf3=1-sum(rf.pred3==wine[rowtrain,12])/nrow(wine[rowtrain,])
error_bst3=1-sum(bst.pred3[,2]==wine[rowtrain,12])/nrow(wine[rowtrain,])

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

model	error_rate
SVM_linear	0.1358950
SVM_radial	0.1358950
Tree	0.1030928
Bagging	0.0000000
Random_forest	0.0000000
Boosting	0.1040300