wineTestAgain8.r

## File: WineTestAgain9.r  
## Author: A. Breitzman  
## Date: 4/2/2017; Revised 6/10/18  
## Description: One v all, one v one

library(e1071)  
library("neuralnet")  
library("plyr")

## Warning: package 'plyr' was built under R version 3.3.3

setwd("C:/Users/Tony/Dropbox/Rowan/DM2/Lecture4")  
whiteWineData <-read.csv("winequality-white.csv",header = TRUE, sep = ";", stringsAsFactors = TRUE)  
  
maxs <- apply(whiteWineData, 2, max)  
mins <- apply(whiteWineData, 2, min)  
scaledWhite <- as.data.frame(scale(whiteWineData, center = mins, scale = maxs - mins))  
names(scaledWhite)

## [1] "fixed.acidity" "volatile.acidity" "citric.acid"   
## [4] "residual.sugar" "chlorides" "free.sulfur.dioxide"   
## [7] "total.sulfur.dioxide" "density" "pH"   
## [10] "sulphates" "alcohol" "quality"

## Unscale quality first  
scaledWhite$quality <- 3+6\*scaledWhite$quality  
count(scaledWhite$quality)

## x freq  
## 1 3 20  
## 2 4 163  
## 3 5 1457  
## 4 6 2198  
## 5 7 880  
## 6 8 175  
## 7 9 5

## Baseline 1 Random Forest on all classes  
  
set.seed(2)  
train <- sample(1:nrow(scaledWhite),nrow(scaledWhite)\*(8/10))  
test<- -train  
  
trainingData<-scaledWhite[train,]  
testingData<-scaledWhite[test,]  
  
n<-names(scaledWhite)  
length(n)

## [1] 12

nSub <- n[1:11]  
nSub

## [1] "fixed.acidity" "volatile.acidity" "citric.acid"   
## [4] "residual.sugar" "chlorides" "free.sulfur.dioxide"   
## [7] "total.sulfur.dioxide" "density" "pH"   
## [10] "sulphates" "alcohol"

formula <- as.formula(paste("as.factor(quality) ~", paste(nSub, collapse = " + ")))  
formula

## as.factor(quality) ~ fixed.acidity + volatile.acidity + citric.acid +   
## residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +   
## density + pH + sulphates + alcohol

trainingData<-scaledWhite[train,]  
testingData<-scaledWhite[test,]  
  
  
  
library(randomForest)

## randomForest 4.6-12

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

formula

## as.factor(quality) ~ fixed.acidity + volatile.acidity + citric.acid +   
## residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +   
## density + pH + sulphates + alcohol

fitB1 <- randomForest(formula,  
 data=trainingData,   
 importance=TRUE,   
 ntree=2000)  
  
  
testingData$result <- predict(fitB1, testingData)  
count(testingData,c("quality","result"))

## quality result freq  
## 1 3 5 4  
## 2 3 6 2  
## 3 4 4 11  
## 4 4 5 14  
## 5 4 6 11  
## 6 5 5 201  
## 7 5 6 89  
## 8 5 7 3  
## 9 6 4 1  
## 10 6 5 54  
## 11 6 6 372  
## 12 6 7 20  
## 13 7 5 2  
## 14 7 6 73  
## 15 7 7 85  
## 16 7 8 1  
## 17 8 6 13  
## 18 8 7 13  
## 19 8 8 9  
## 20 9 6 1  
## 21 9 7 1

dim(testingData)

## [1] 980 13

correct <- 11 + 201 +372 + 85 + 9  
correct

## [1] 678

err <- 980 - correct  
err/980

## [1] 0.3081633

## 69% accurate  
  
## Baseline 2 reduce to 3 classes  
  
## Set B2 to -1 for quality 3-4  
## Set B2 to 0 for quality 5,6,7  
## Set B2 to 1 for quality 8-9  
trainingData$B2 <- sapply(trainingData$quality, function(b) {  
 if (b >=8){  
 return(1)  
 }else{  
 if (b>=5){  
 return(0)  
 }else{  
 return(-1)  
 }  
 }  
})  
  
testingData$B2 <- sapply(testingData$quality, function(b) {  
 if (b >=8){  
 return(1)  
 }else{  
 if (b>=5){  
 return(0)  
 }else{  
 return(-1)  
 }  
 }  
})  
  
set.seed(2)  
  
n<-names(scaledWhite)  
length(n)

## [1] 12

nSub <- n[1:11]  
nSub

## [1] "fixed.acidity" "volatile.acidity" "citric.acid"   
## [4] "residual.sugar" "chlorides" "free.sulfur.dioxide"   
## [7] "total.sulfur.dioxide" "density" "pH"   
## [10] "sulphates" "alcohol"

formula <- as.formula(paste("as.factor(B2)~", paste(nSub, collapse = " + ")))  
formula

## as.factor(B2) ~ fixed.acidity + volatile.acidity + citric.acid +   
## residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +   
## density + pH + sulphates + alcohol

fit0 <- randomForest(formula,  
 data=trainingData,   
 importance=TRUE,   
 ntree=2000)  
  
  
testingData$result0 <- predict(fit0, testingData)  
count(testingData,c("B2","result0"))

## B2 result0 freq  
## 1 -1 -1 10  
## 2 -1 0 32  
## 3 0 -1 1  
## 4 0 0 900  
## 5 1 0 30  
## 6 1 1 7

dim(testingData)

## [1] 980 15

correct <- 10 + 900 + 7  
correct

## [1] 917

err <- 980 - correct  
err/980

## [1] 0.06428571

## 6% error but only predicts 1/4 of the bad wines and less than 1/4 of the good wines  
  
  
## Set quality 8-9 to 1 and rest 0 to see if we can  
## get a 1 vs all setup to work for quality 8-9 wines  
trainingData$Q2 <- ifelse(trainingData$quality>=8,1,0)  
testingData$Q2 <- ifelse(testingData$quality>=8,1,0)  
  
formula <- as.formula(paste("as.factor(Q2) ~", paste(nSub, collapse = " + ")))  
formula

## as.factor(Q2) ~ fixed.acidity + volatile.acidity + citric.acid +   
## residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +   
## density + pH + sulphates + alcohol

fit <- randomForest(formula,  
 data=trainingData,   
 importance=TRUE,   
 ntree=2000)  
  
  
  
  
testingData$result <- predict(fit, testingData)  
count(testingData,c("Q2","result"))

## Q2 result freq  
## 1 0 0 943  
## 2 1 0 30  
## 3 1 1 7

## still only 7 out of 37 good wines correct  
  
count(testingData,c("Q2","quality","result"))

## Q2 quality result freq  
## 1 0 3 0 6  
## 2 0 4 0 36  
## 3 0 5 0 293  
## 4 0 6 0 447  
## 5 0 7 0 161  
## 6 1 8 0 28  
## 7 1 8 1 7  
## 8 1 9 0 2

## This suggests a 1 v all approach is not going to work with the 8 and 9 wines  
  
  
  
## let's try a 1 v 1 approach and see if we can get a random forest  
## to distinguish between 3-4 and 8-9  
  
  
t <- trainingData[which(trainingData$quality >7),]  
dim(t)

## [1] 143 14

s <- trainingData[which(trainingData$quality <5),]  
dim(s)

## [1] 141 14

t$Q2 <-1  
s$Q2 <-0  
temp <- rbind(t,s)  
count(temp,c("quality","Q2"))

## quality Q2 freq  
## 1 3 0 14  
## 2 4 0 127  
## 3 8 1 140  
## 4 9 1 3

fit <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
testingData$result <- predict(fit, testingData)  
count(testingData,c("Q2","quality","result"))

## Q2 quality result freq  
## 1 0 3 0 6  
## 2 0 4 0 35  
## 3 0 4 1 1  
## 4 0 5 0 255  
## 5 0 5 1 38  
## 6 0 6 0 245  
## 7 0 6 1 202  
## 8 0 7 0 38  
## 9 0 7 1 123  
## 10 1 8 0 2  
## 11 1 8 1 33  
## 12 1 9 1 2

## Well now we're getting somewhere  
## The 0 Q2's are all messed up but that's to be expected  
## it's doing a good job of distinguishing between the 3-4 v 8-9  
  
  
## let's see how we do in predicting 3-4 v 7  
t <- trainingData[which(trainingData$quality <=4),]  
s <- trainingData[which(trainingData$quality ==7),]  
  
dim(s)

## [1] 719 14

dim(t)

## [1] 141 14

s2samp <- sample(1:nrow(s),140)  
s2 <- s[s2samp,]  
t$Q2 <-0  
s2$Q2 <-1  
  
  
temp<- rbind(t,s2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 3 14  
## 2 0 4 127  
## 3 1 7 140

fit2 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$result <- predict(fit2, testingData)  
count(testingData,c("Q2","quality","result"))

## Q2 quality result freq  
## 1 0 3 0 5  
## 2 0 3 1 1  
## 3 0 4 0 32  
## 4 0 4 1 4  
## 5 0 5 0 212  
## 6 0 5 1 81  
## 7 0 6 0 172  
## 8 0 6 1 275  
## 9 0 7 0 22  
## 10 0 7 1 139  
## 11 1 8 0 1  
## 12 1 8 1 34  
## 13 1 9 1 2

## good job on the 3-4, ok on the 7  
  
s <- trainingData[which(trainingData$quality ==6),]  
  
dim(s)

## [1] 1751 14

dim(t)

## [1] 141 14

s2samp <- sample(1:nrow(s),140)  
s2 <- s[s2samp,]  
t$Q2 <-0  
s2$Q2 <-1  
  
  
temp<- rbind(t,s2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 3 14  
## 2 0 4 127  
## 3 1 6 140

fit3 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$result <- predict(fit3, testingData)  
count(testingData,c("Q2","quality","result"))

## Q2 quality result freq  
## 1 0 3 0 5  
## 2 0 3 1 1  
## 3 0 4 0 31  
## 4 0 4 1 5  
## 5 0 5 0 133  
## 6 0 5 1 160  
## 7 0 6 0 84  
## 8 0 6 1 363  
## 9 0 7 0 14  
## 10 0 7 1 147  
## 11 1 8 1 35  
## 12 1 9 1 2

## About a 25% error on the 6's much better on the 3-4  
  
  
  
s <- trainingData[which(trainingData$quality ==5),]  
t <- trainingData[which(trainingData$quality >=8),]  
dim(s)

## [1] 1164 14

dim(t)

## [1] 143 14

s2samp <- sample(1:nrow(s),145)  
s2 <- s[s2samp,]  
t$Q2 <-1  
s2$Q2 <-0  
  
  
  
temp<- rbind(t,s2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 5 145  
## 2 1 8 140  
## 3 1 9 3

fit4 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$result <- predict(fit4, testingData)  
count(testingData,c("Q2","quality","result"))

## Q2 quality result freq  
## 1 0 3 0 5  
## 2 0 3 1 1  
## 3 0 4 0 33  
## 4 0 4 1 3  
## 5 0 5 0 272  
## 6 0 5 1 21  
## 7 0 6 0 276  
## 8 0 6 1 171  
## 9 0 7 0 51  
## 10 0 7 1 110  
## 11 1 8 0 3  
## 12 1 8 1 32  
## 13 1 9 1 2

## not too bad  
  
s <- trainingData[which(trainingData$quality ==6),]  
t <- trainingData[which(trainingData$quality >=8),]  
dim(s)

## [1] 1751 14

dim(t)

## [1] 143 14

s2samp <- sample(1:nrow(s),145)  
s2 <- s[s2samp,]  
t$Q2 <-1  
s2$Q2 <-0  
  
  
temp<- rbind(t,s2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 6 145  
## 2 1 8 140  
## 3 1 9 3

fit5 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$result <- predict(fit5, testingData)  
count(testingData,c("Q2","quality","result"))

## Q2 quality result freq  
## 1 0 3 0 5  
## 2 0 3 1 1  
## 3 0 4 0 34  
## 4 0 4 1 2  
## 5 0 5 0 277  
## 6 0 5 1 16  
## 7 0 6 0 344  
## 8 0 6 1 103  
## 9 0 7 0 81  
## 10 0 7 1 80  
## 11 1 8 0 8  
## 12 1 8 1 27  
## 13 1 9 1 2

## Not great, but not awful  
  
s <- trainingData[-which(trainingData$quality ==7),]  
t <- trainingData[which(trainingData$quality ==7),]  
dim(s)

## [1] 3199 14

dim(t)

## [1] 719 14

s2samp <- sample(1:nrow(s),719)  
s2 <- s[s2samp,]  
t$Q2 <-0  
s2$Q2 <-1  
  
  
temp<- rbind(t,s2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 7 719  
## 2 1 3 5  
## 3 1 4 29  
## 4 1 5 263  
## 5 1 6 401  
## 6 1 8 20  
## 7 1 9 1

fit6 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$result <- predict(fit6, testingData)  
count(testingData,c("Q2","quality","result"))

## Q2 quality result freq  
## 1 0 3 0 1  
## 2 0 3 1 5  
## 3 0 4 0 2  
## 4 0 4 1 34  
## 5 0 5 0 17  
## 6 0 5 1 276  
## 7 0 6 0 149  
## 8 0 6 1 298  
## 9 0 7 0 130  
## 10 0 7 1 31  
## 11 1 8 0 27  
## 12 1 8 1 8  
## 13 1 9 0 2

## Not great, but not awful  
  
  
s <- trainingData[which(trainingData$quality ==7),]  
t <- trainingData[which(trainingData$quality >=8),]  
dim(s)

## [1] 719 14

dim(t)

## [1] 143 14

s2samp <- sample(1:nrow(s),145)  
s2 <- s[s2samp,]  
t$Q2 <-0  
s2$Q2 <-1  
  
  
temp<- rbind(t,s2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 8 140  
## 2 0 9 3  
## 3 1 7 145

fit6 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$result <- predict(fit6, testingData)  
count(testingData,c("Q2","quality","result"))

## Q2 quality result freq  
## 1 0 3 0 2  
## 2 0 3 1 4  
## 3 0 4 0 3  
## 4 0 4 1 33  
## 5 0 5 0 25  
## 6 0 5 1 268  
## 7 0 6 0 78  
## 8 0 6 1 369  
## 9 0 7 0 38  
## 10 0 7 1 123  
## 11 1 8 0 20  
## 12 1 8 1 15  
## 13 1 9 0 1  
## 14 1 9 1 1

## Not great, but not awful  
  
  
  
  
#####################################################  
## Let's do the final prediction  
## p1 will be 345 v 6789  
## p2 will be 34 v 56789  
## p3 will be 34567 v 89  
## p4 will be 3456 v 789  
  
set.seed(2)  
s <- trainingData[which(trainingData$quality <=5),]  
t <- trainingData[which(trainingData$quality >=6),]  
dim(s)

## [1] 1305 14

dim(t)

## [1] 2613 14

tsamp <-sample(1:nrow(t),1305)  
t2 <- t[tsamp,]  
t2$Q2 <-1  
s$Q2 <-0  
  
  
temp<- rbind(s,t2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 3 14  
## 2 0 4 127  
## 3 0 5 1164  
## 4 1 6 870  
## 5 1 7 363  
## 6 1 8 70  
## 7 1 9 2

p1 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$p1 <-predict(p1, testingData)  
count(testingData,c("Q2","quality","p1"))

## Q2 quality p1 freq  
## 1 0 3 0 4  
## 2 0 3 1 2  
## 3 0 4 0 30  
## 4 0 4 1 6  
## 5 0 5 0 241  
## 6 0 5 1 52  
## 7 0 6 0 121  
## 8 0 6 1 326  
## 9 0 7 0 7  
## 10 0 7 1 154  
## 11 1 8 1 35  
## 12 1 9 1 2

## now do p2  
s <- trainingData[which(trainingData$quality <=4),]  
t <- trainingData[which(trainingData$quality >=5),]  
dim(s)

## [1] 141 14

dim(t)

## [1] 3777 14

tsamp <-sample(1:nrow(t),145)  
t2 <- t[tsamp,]  
t2$Q2 <-1  
s$Q2 <-0  
  
  
temp<- rbind(s,t2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 3 14  
## 2 0 4 127  
## 3 1 5 37  
## 4 1 6 78  
## 5 1 7 28  
## 6 1 8 2

p2 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$p2 <- predict(p2, testingData)  
count(testingData,c("Q2","quality","p2"))

## Q2 quality p2 freq  
## 1 0 3 0 5  
## 2 0 3 1 1  
## 3 0 4 0 31  
## 4 0 4 1 5  
## 5 0 5 0 126  
## 6 0 5 1 167  
## 7 0 6 0 91  
## 8 0 6 1 356  
## 9 0 7 0 10  
## 10 0 7 1 151  
## 11 1 8 0 2  
## 12 1 8 1 33  
## 13 1 9 1 2

## now do p3  
s <- trainingData[which(trainingData$quality <=7),]  
t <- trainingData[which(trainingData$quality >=8),]  
dim(s)

## [1] 3775 14

dim(t)

## [1] 143 14

ssamp <-sample(1:nrow(s),145)  
s2 <- s[ssamp,]  
t$Q2 <-1  
s2$Q2 <-0  
  
  
temp<- rbind(s2,t)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 4 6  
## 2 0 5 40  
## 3 0 6 75  
## 4 0 7 24  
## 5 1 8 140  
## 6 1 9 3

p3 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$p3 <- predict(p3, testingData)  
count(testingData,c("Q2","quality","p3"))

## Q2 quality p3 freq  
## 1 0 3 0 5  
## 2 0 3 1 1  
## 3 0 4 0 35  
## 4 0 4 1 1  
## 5 0 5 0 278  
## 6 0 5 1 15  
## 7 0 6 0 338  
## 8 0 6 1 109  
## 9 0 7 0 81  
## 10 0 7 1 80  
## 11 1 8 0 8  
## 12 1 8 1 27  
## 13 1 9 1 2

## now do p4  
s <- trainingData[which(trainingData$quality <=6),]  
t <- trainingData[which(trainingData$quality >=7),]  
dim(s)

## [1] 3056 14

dim(t)

## [1] 862 14

ssamp <-sample(1:nrow(s),863)  
s2 <- s[ssamp,]  
t$Q2 <-1  
s2$Q2 <-0  
  
  
temp<- rbind(s2,t)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 3 3  
## 2 0 4 43  
## 3 0 5 325  
## 4 0 6 492  
## 5 1 7 719  
## 6 1 8 140  
## 7 1 9 3

p4 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$p4 <- predict(p4, testingData)  
count(testingData,c("Q2","quality","p4"))

## Q2 quality p4 freq  
## 1 0 3 0 6  
## 2 0 4 0 34  
## 3 0 4 1 2  
## 4 0 5 0 278  
## 5 0 5 1 15  
## 6 0 6 0 296  
## 7 0 6 1 151  
## 8 0 7 0 29  
## 9 0 7 1 132  
## 10 1 8 0 2  
## 11 1 8 1 33  
## 12 1 9 1 2

testingData$result <- as.numeric(testingData$p1)-1 +as.numeric(testingData$p2)-1+as.numeric(testingData$p3)-1+as.numeric(testingData$p4)-1  
count(testingData,c("quality","result"))

## quality result freq  
## 1 3 0 3  
## 2 3 1 2  
## 3 3 2 1  
## 4 4 0 29  
## 5 4 1 2  
## 6 4 2 4  
## 7 4 4 1  
## 8 5 0 120  
## 9 5 1 115  
## 10 5 2 45  
## 11 5 3 8  
## 12 5 4 5  
## 13 6 0 59  
## 14 6 1 80  
## 15 6 2 147  
## 16 6 3 76  
## 17 6 4 85  
## 18 7 0 1  
## 19 7 1 7  
## 20 7 2 27  
## 21 7 3 48  
## 22 7 4 78  
## 23 8 2 2  
## 24 8 3 8  
## 25 8 4 25  
## 26 9 4 2

count(testingData,c("B2","result"))

## B2 result freq  
## 1 -1 0 32  
## 2 -1 1 4  
## 3 -1 2 5  
## 4 -1 4 1  
## 5 0 0 180  
## 6 0 1 202  
## 7 0 2 219  
## 8 0 3 132  
## 9 0 4 168  
## 10 1 2 2  
## 11 1 3 8  
## 12 1 4 27

testingData$predQ2 <- sapply(testingData$result, function(b) {  
 if (b ==0){  
 return(-1)  
 }else{  
 if (b==4){  
 return(1)  
 }else{  
 return(0)  
 }  
 }  
})  
  
count(testingData,c("B2","predQ2"))

## B2 predQ2 freq  
## 1 -1 -1 32  
## 2 -1 0 9  
## 3 -1 1 1  
## 4 0 -1 180  
## 5 0 0 553  
## 6 0 1 168  
## 7 1 0 10  
## 8 1 1 27

dim(testingData)

## [1] 980 21

correct <- 32 + 553 + 27  
err <- 980 - correct  
err/980

## [1] 0.3755102

## only 62% correct but we get most of the 3-4 and 8-9 picked but also a lot of false positives  
  
  
  
################################################################  
## One last try  
## a1 345 v 6789  
## a2 34 v 5  
## a3 3456 v 789  
## a4 7 v 89  
## a1 and a2 == 0 --> 3-4 bad (-1)  
## a3 and a4 == 1 --> 8-9 good (1)  
## all else goes to ok (0)  
  
set.seed(2)  
s <- trainingData[which(trainingData$quality <=5),]  
t <- trainingData[which(trainingData$quality >=6),]  
dim(s)

## [1] 1305 14

dim(t)

## [1] 2613 14

tsamp <-sample(1:nrow(t),1305)  
t2 <- t[tsamp,]  
t2$Q2 <-1  
s$Q2 <-0  
  
  
temp<- rbind(s,t2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 3 14  
## 2 0 4 127  
## 3 0 5 1164  
## 4 1 6 870  
## 5 1 7 363  
## 6 1 8 70  
## 7 1 9 2

a1 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$a1 <-predict(a1, testingData)  
count(testingData,c("Q2","quality","a1"))

## Q2 quality a1 freq  
## 1 0 3 0 4  
## 2 0 3 1 2  
## 3 0 4 0 30  
## 4 0 4 1 6  
## 5 0 5 0 241  
## 6 0 5 1 52  
## 7 0 6 0 121  
## 8 0 6 1 326  
## 9 0 7 0 7  
## 10 0 7 1 154  
## 11 1 8 1 35  
## 12 1 9 1 2

## a2 34 v 5  
s <- trainingData[which(trainingData$quality <=4),]  
t <- trainingData[which(trainingData$quality ==5),]  
dim(s)

## [1] 141 14

dim(t)

## [1] 1164 14

tsamp <-sample(1:nrow(t),141)  
t2 <- t[tsamp,]  
t2$Q2 <-1  
s$Q2 <-0  
  
  
temp<- rbind(s,t2)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 3 14  
## 2 0 4 127  
## 3 1 5 141

a2 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$a2 <-predict(a2, testingData)  
count(testingData,c("Q2","quality","a1","a2"))

## Q2 quality a1 a2 freq  
## 1 0 3 0 0 3  
## 2 0 3 0 1 1  
## 3 0 3 1 0 1  
## 4 0 3 1 1 1  
## 5 0 4 0 0 25  
## 6 0 4 0 1 5  
## 7 0 4 1 0 3  
## 8 0 4 1 1 3  
## 9 0 5 0 0 63  
## 10 0 5 0 1 178  
## 11 0 5 1 0 11  
## 12 0 5 1 1 41  
## 13 0 6 0 0 32  
## 14 0 6 0 1 89  
## 15 0 6 1 0 78  
## 16 0 6 1 1 248  
## 17 0 7 0 0 3  
## 18 0 7 0 1 4  
## 19 0 7 1 0 45  
## 20 0 7 1 1 109  
## 21 1 8 1 0 12  
## 22 1 8 1 1 23  
## 23 1 9 1 0 2

## a3 3456 v 789  
s <- trainingData[which(trainingData$quality <=6),]  
t <- trainingData[which(trainingData$quality >=7),]  
dim(s)

## [1] 3056 14

dim(t)

## [1] 862 14

ssamp <-sample(1:nrow(s),862)  
s2 <- s[ssamp,]  
t$Q2 <-1  
s2$Q2 <-0  
  
  
temp<- rbind(s2,t)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 3 1  
## 2 0 4 38  
## 3 0 5 331  
## 4 0 6 492  
## 5 1 7 719  
## 6 1 8 140  
## 7 1 9 3

a3 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$a3 <-predict(a3, testingData)  
count(testingData,c("Q2","quality","a1","a2","a3"))

## Q2 quality a1 a2 a3 freq  
## 1 0 3 0 0 0 3  
## 2 0 3 0 1 0 1  
## 3 0 3 1 0 0 1  
## 4 0 3 1 1 0 1  
## 5 0 4 0 0 0 25  
## 6 0 4 0 1 0 5  
## 7 0 4 1 0 0 1  
## 8 0 4 1 0 1 2  
## 9 0 4 1 1 0 2  
## 10 0 4 1 1 1 1  
## 11 0 5 0 0 0 60  
## 12 0 5 0 0 1 3  
## 13 0 5 0 1 0 175  
## 14 0 5 0 1 1 3  
## 15 0 5 1 0 0 9  
## 16 0 5 1 0 1 2  
## 17 0 5 1 1 0 35  
## 18 0 5 1 1 1 6  
## 19 0 6 0 0 0 31  
## 20 0 6 0 0 1 1  
## 21 0 6 0 1 0 87  
## 22 0 6 0 1 1 2  
## 23 0 6 1 0 0 37  
## 24 0 6 1 0 1 41  
## 25 0 6 1 1 0 146  
## 26 0 6 1 1 1 102  
## 27 0 7 0 0 0 2  
## 28 0 7 0 0 1 1  
## 29 0 7 0 1 0 3  
## 30 0 7 0 1 1 1  
## 31 0 7 1 0 0 8  
## 32 0 7 1 0 1 37  
## 33 0 7 1 1 0 19  
## 34 0 7 1 1 1 90  
## 35 1 8 1 0 0 1  
## 36 1 8 1 0 1 11  
## 37 1 8 1 1 0 1  
## 38 1 8 1 1 1 22  
## 39 1 9 1 0 1 2

## a4 7 v 89  
s <- trainingData[which(trainingData$quality ==7),]  
t <- trainingData[which(trainingData$quality >=8),]  
dim(s)

## [1] 719 14

dim(t)

## [1] 143 14

ssamp <-sample(1:nrow(s),143)  
s2 <- s[ssamp,]  
t$Q2 <-1  
s2$Q2 <-0  
  
  
temp<- rbind(s2,t)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 7 143  
## 2 1 8 140  
## 3 1 9 3

a4 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$a4 <-predict(a4, testingData)  
count(testingData,c("Q2","quality","a1","a2","a3","a4"))

## Q2 quality a1 a2 a3 a4 freq  
## 1 0 3 0 0 0 0 2  
## 2 0 3 0 0 0 1 1  
## 3 0 3 0 1 0 0 1  
## 4 0 3 1 0 0 0 1  
## 5 0 3 1 1 0 0 1  
## 6 0 4 0 0 0 0 22  
## 7 0 4 0 0 0 1 3  
## 8 0 4 0 1 0 0 3  
## 9 0 4 0 1 0 1 2  
## 10 0 4 1 0 0 1 1  
## 11 0 4 1 0 1 0 2  
## 12 0 4 1 1 0 0 2  
## 13 0 4 1 1 1 0 1  
## 14 0 5 0 0 0 0 57  
## 15 0 5 0 0 0 1 3  
## 16 0 5 0 0 1 0 1  
## 17 0 5 0 0 1 1 2  
## 18 0 5 0 1 0 0 146  
## 19 0 5 0 1 0 1 29  
## 20 0 5 0 1 1 0 3  
## 21 0 5 1 0 0 0 7  
## 22 0 5 1 0 0 1 2  
## 23 0 5 1 0 1 0 2  
## 24 0 5 1 1 0 0 28  
## 25 0 5 1 1 0 1 7  
## 26 0 5 1 1 1 0 3  
## 27 0 5 1 1 1 1 3  
## 28 0 6 0 0 0 0 28  
## 29 0 6 0 0 0 1 3  
## 30 0 6 0 0 1 1 1  
## 31 0 6 0 1 0 0 68  
## 32 0 6 0 1 0 1 19  
## 33 0 6 0 1 1 0 2  
## 34 0 6 1 0 0 0 32  
## 35 0 6 1 0 0 1 5  
## 36 0 6 1 0 1 0 29  
## 37 0 6 1 0 1 1 12  
## 38 0 6 1 1 0 0 116  
## 39 0 6 1 1 0 1 30  
## 40 0 6 1 1 1 0 67  
## 41 0 6 1 1 1 1 35  
## 42 0 7 0 0 0 0 2  
## 43 0 7 0 0 1 0 1  
## 44 0 7 0 1 0 0 2  
## 45 0 7 0 1 0 1 1  
## 46 0 7 0 1 1 0 1  
## 47 0 7 1 0 0 0 5  
## 48 0 7 1 0 0 1 3  
## 49 0 7 1 0 1 0 24  
## 50 0 7 1 0 1 1 13  
## 51 0 7 1 1 0 0 17  
## 52 0 7 1 1 0 1 2  
## 53 0 7 1 1 1 0 56  
## 54 0 7 1 1 1 1 34  
## 55 1 8 1 0 0 0 1  
## 56 1 8 1 0 1 0 6  
## 57 1 8 1 0 1 1 5  
## 58 1 8 1 1 0 0 1  
## 59 1 8 1 1 1 0 8  
## 60 1 8 1 1 1 1 14  
## 61 1 9 1 0 1 0 1  
## 62 1 9 1 0 1 1 1

testingData$aresult <- as.numeric(testingData$a1)-1 +as.numeric(testingData$a2)-1+as.numeric(testingData$a3)-1+as.numeric(testingData$a4)-1  
count(testingData,c("Q2","aresult","result"))

## Q2 aresult result freq  
## 1 0 0 0 102  
## 2 0 0 1 9  
## 3 0 1 0 88  
## 4 0 1 1 158  
## 5 0 1 2 22  
## 6 0 1 3 8  
## 7 0 1 4 1  
## 8 0 2 0 22  
## 9 0 2 1 37  
## 10 0 2 2 167  
## 11 0 2 3 34  
## 12 0 2 4 32  
## 13 0 3 1 2  
## 14 0 3 2 33  
## 15 0 3 3 83  
## 16 0 3 4 73  
## 17 0 4 2 2  
## 18 0 4 3 7  
## 19 0 4 4 63  
## 20 1 1 2 1  
## 21 1 2 2 1  
## 22 1 2 3 4  
## 23 1 2 4 3  
## 24 1 3 3 4  
## 25 1 3 4 10  
## 26 1 4 4 14

testingData$pred2Q2 <- sapply(testingData$aresult, function(b) {  
 if (b ==0){  
 return(-1)  
 }else{  
 if (b==4){  
 return(1)  
 }else{  
 return(0)  
 }  
 }  
})  
  
count(testingData,c("B2","pred2Q2"))

## B2 pred2Q2 freq  
## 1 -1 -1 24  
## 2 -1 0 18  
## 3 0 -1 87  
## 4 0 0 742  
## 5 0 1 72  
## 6 1 0 23  
## 7 1 1 14

dim(testingData)

## [1] 980 27

correct <- 24 + 742 + 14  
err <- 980 - correct  
err/980

## [1] 0.2040816

## only 80% correct but we only get some 8-9 and 3-4 correct  
  
testingData$result0 <- predict(fit0, testingData)  
count(testingData,c("B2","predQ2","pred2Q2","result0"))

## B2 predQ2 pred2Q2 result0 freq  
## 1 -1 -1 -1 -1 8  
## 2 -1 -1 -1 0 16  
## 3 -1 -1 0 -1 2  
## 4 -1 -1 0 0 6  
## 5 -1 0 0 0 9  
## 6 -1 1 0 0 1  
## 7 0 -1 -1 -1 1  
## 8 0 -1 -1 0 77  
## 9 0 -1 0 0 102  
## 10 0 0 -1 0 9  
## 11 0 0 0 0 535  
## 12 0 0 1 0 9  
## 13 0 1 0 0 105  
## 14 0 1 1 0 63  
## 15 1 0 0 0 10  
## 16 1 1 0 0 12  
## 17 1 1 0 1 1  
## 18 1 1 1 0 8  
## 19 1 1 1 1 6

bad <- 26  
ok <- 102 + 9 + 535 + 9 + 105  
good <- 1 + 8 + 6  
  
ok

## [1] 760

correct <- bad + ok + good  
correct

## [1] 801

err <- 980 - correct  
err

## [1] 179

err/980

## [1] 0.1826531

## Blend is 72% correct with 26 3-4 predicted, 760 5-6-7 predicted, 15 good predicted

#############################################################################  
## Let's see how adaBoost does compared with random forest.  
## we'll just do 7 v 89  
## a4 7 v 89  
s <- trainingData[which(trainingData$quality ==7),]  
t <- trainingData[which(trainingData$quality >=8),]  
dim(s)

## [1] 719 14

dim(t)

## [1] 143 14

ssamp <-sample(1:nrow(s),143)  
s2 <- s[ssamp,]  
t$Q2 <-1  
s2$Q2 <-0  
  
  
temp<- rbind(s2,t)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 0 7 143  
## 2 1 8 140  
## 3 1 9 3

a4 <- randomForest(formula,  
 data=temp,   
 importance=TRUE,   
 ntree=2000)  
  
testingData$a4 <-predict(a4, testingData)  
count(testingData,c("quality","a4"))

## quality a4 freq  
## 1 3 0 2  
## 2 3 1 4  
## 3 4 0 32  
## 4 4 1 4  
## 5 5 0 252  
## 6 5 1 41  
## 7 6 0 349  
## 8 6 1 98  
## 9 7 0 100  
## 10 7 1 61  
## 11 8 0 13  
## 12 8 1 22  
## 13 9 0 2

correct <- 100+22  
err <- 61+15  
err/(err+correct)

## [1] 0.3838384

library(freestats)

## Warning: package 'freestats' was built under R version 3.3.3

names(temp)

## [1] "fixed.acidity" "volatile.acidity" "citric.acid"   
## [4] "residual.sugar" "chlorides" "free.sulfur.dioxide"   
## [7] "total.sulfur.dioxide" "density" "pH"   
## [10] "sulphates" "alcohol" "quality"   
## [13] "B2" "Q2"

## We'll start with 50 trees since we know our data is highly predictive  
temp[1:10,14]

## [1] 0 0 0 0 0 0 0 0 0 0

t$Q2 <-1  
s2$Q2 <- -1  
  
  
temp<- rbind(s2,t)  
count(temp,c("Q2","quality"))

## Q2 quality freq  
## 1 -1 7 143  
## 2 1 8 140  
## 3 1 9 3

names(temp)

## [1] "fixed.acidity" "volatile.acidity" "citric.acid"   
## [4] "residual.sugar" "chlorides" "free.sulfur.dioxide"   
## [7] "total.sulfur.dioxide" "density" "pH"   
## [10] "sulphates" "alcohol" "quality"   
## [13] "B2" "Q2"

res <- adaBoost(dat.train=temp[1:11],y.train=temp[13],B=50)  
result6<-classify(res,testingData[,1:11])  
testingData$a4b <- result6  
count(testingData,c("quality","a4b"))

## quality a4b freq  
## 1 3 1 6  
## 2 4 1 36  
## 3 5 1 293  
## 4 6 -1 3  
## 5 6 1 444  
## 6 7 -1 4  
## 7 7 1 157  
## 8 8 1 35  
## 9 9 1 2

correct <- 4+ 37  
err <- 157  
err/(err+correct)

## [1] 0.7929293