breastCancer4.r

setwd("C:/Users/Tony/Dropbox/Rowan/DM2/Lecture1/WisconsinBreastCancer")  
f=read.csv(file="wdbc\_data\_abclean.csv",header=F,stringsAsFactors = TRUE)  
  
dim(f)

## [1] 569 32

f[1,]

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## 1 842302 1 17.99 10.38 122.8 1001 0.1184 0.2776 0.3001 0.1471 0.2419  
## V12 V13 V14 V15 V16 V17 V18 V19 V20  
## 1 0.07871 1.095 0.9053 8.589 153.4 0.006399 0.04904 0.05373 0.01587  
## V21 V22 V23 V24 V25 V26 V27 V28 V29 V30  
## 1 0.03003 0.006193 25.38 17.33 184.6 2019 0.1622 0.6656 0.7119 0.2654  
## V31 V32  
## 1 0.4601 0.1189

## We don't need the first column, it's just an id  
ff <-subset(f,select=-V1)  
dim(ff)

## [1] 569 31

library(plyr)

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

count(ff,c("V2"))

## V2 freq  
## 1 0 357  
## 2 1 212

## 357 benign, 212 malignant  
  
  
set.seed(2)  
train=sample(1:nrow(ff),nrow(ff)\*(8/10))  
test=-train  
  
trainingData=ff[train,]  
testingData=ff[test,]  
dim(trainingData)

## [1] 455 31

dim(testingData)

## [1] 114 31

## We make V2 a factor because if it's a factor the randomforest call  
## solves as classification. Otherwise it attempts a regression  
## in other words it will return a 0 or 1 in this case rather than a range between 0 and 1  
formula = as.factor(V2)~V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14+V15+V16+V17+V18+V19+V20+V21+V22+V23+V24+V25+V26+V27+V28+V29+V30+V31+V32  
  
##install.packages("randomForest")  
library(randomForest)

## randomForest 4.6-12

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

fit <- randomForest(formula,  
 data=trainingData,   
 importance=TRUE,   
 ntree=2000)  
  
  
  
testingData$result <- predict(fit, testingData)  
count(testingData,c("V2","result"))

## V2 result freq  
## 1 0 0 77  
## 2 0 1 1  
## 3 1 0 2  
## 4 1 1 34

err = 3/(77+3+34)  
err

## [1] 0.02631579

## Let's compare this to a single tree using   
library(rpart)  
treeFit <- rpart(formula,  
 data=trainingData,method="class")  
  
testingData$result2 <- predict(treeFit, testingData,type="class")  
  
  
count(testingData,c("V2","result2"))

## V2 result2 freq  
## 1 0 0 76  
## 2 0 1 2  
## 3 1 0 2  
## 4 1 1 34

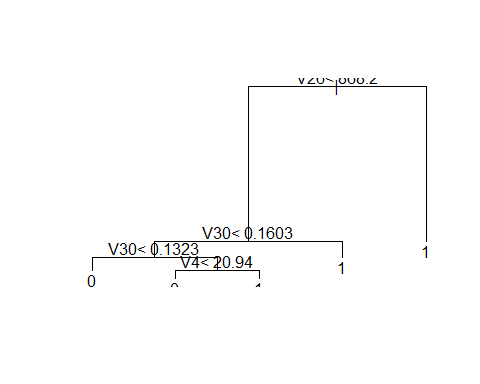
err = 4/(76+4+34)  
err

## [1] 0.03508772

## One more error than forest  
  
  
count(testingData,c("V2","result","result2"))

## V2 result result2 freq  
## 1 0 0 0 76  
## 2 0 0 1 1  
## 3 0 1 1 1  
## 4 1 0 0 2  
## 5 1 1 1 34

## single tree gives one more false positive than forest  
  
plot(treeFit); text(treeFit)



y <- sapply(trainingData$V2, function(x){ifelse(x == 0, -1, 1)})  
trainingData[1,]

## V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12  
## 106 1 13.11 15.56 87.21 530.2 0.1398 0.1765 0.2071 0.09601 0.1925 0.07692  
## V13 V14 V15 V16 V17 V18 V19 V20 V21  
## 106 0.3908 0.9238 2.41 34.66 0.007162 0.02912 0.05473 0.01388 0.01547  
## V22 V23 V24 V25 V26 V27 V28 V29 V30 V31  
## 106 0.007098 16.31 22.4 106.4 827.2 0.1862 0.4099 0.6376 0.1986 0.3147  
## V32  
## 106 0.1405

dim(trainingData)

## [1] 455 31

x<-trainingData[,2:31]  
w<-rep(1/455,455)  
  
#install.packages("freestats")  
library(freestats)

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

weak <- decisionStump(X=x,w=w,y=y)  
weak

## $j  
## [1] 21  
##   
## $theta  
## [1] 16.76  
##   
## $m  
## [1] 1  
##   
## attr(,"class")  
## [1] "ds"

x[1:5,21]

## [1] 16.31 12.68 13.71 24.22 23.23

pred <- sapply(x[,21], function(s){ifelse(s <= 16.76,0,1)})  
  
pred[1:10]

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

results<-data.frame(y,pred)  
count(results,c("pred","y"))

## pred y freq  
## 1 0 -1 269  
## 2 0 1 27  
## 3 1 -1 10  
## 4 1 1 149

37/455

## [1] 0.08131868

## 8% error. Pretty strong for a weak learner!  
  
## Could also check my error like this  
result5 <- classify(weak,x)  
results<-data.frame(y,result5)  
count(results,c("y","result5"))

## y result5 freq  
## 1 -1 -1 269  
## 2 -1 1 10  
## 3 1 -1 27  
## 4 1 1 149

## same result of course  
  
  
  
## We can build our own boosting algorithm using the decisionStump above, but ## we'll just use the one that's already built.  
  
## We'll start with 10 trees since we know our data is highly predictive  
res <- adaBoost(dat.train=x,y.train=y,B=10)  
result6<-classify(res,testingData[,2:31])  
result6[1:10]

## [1] 1 1 1 1 1 1 -1 -1 -1 -1

result7<-data.frame(testingData$V2,result6)  
result7[1:5,]

## testingData.V2 result6  
## 1 1 1  
## 2 1 1  
## 3 1 1  
## 4 1 1  
## 5 1 1

count(result7,c("testingData.V2","result6"))

## testingData.V2 result6 freq  
## 1 0 -1 78  
## 2 1 -1 2  
## 3 1 1 34

## 2 errors (1 fewer than with bagging; still 2 false negatives though which ## is bad)  
err<-2/(78+2+34)  
err

## [1] 0.01754386

1-err

## [1] 0.9824561