Project 4

Your name

Tree-based methods

We study the *German credit data set* from the UC Irvine machine learning repository. A set of 20 covariates (attributes) are available (7 numerical, 13 categorical) for 300 customers with bad credit risk and 700 customers with good credit risk (0 = Good, 1 = Bad).

We aim to classify a customer as *good* or *bad* with respect to credit risk. It is worse to class a customer as good when they are bad, than it is to class a customer as bad when they are good.

```
#read data, divide into train and test
German.credit <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/ge
                            stringsAsFactors = TRUE)
# You can also find a doc file with a brief description of the German credit dataset on the web.
colnames(German.credit) = c("checkaccount", "duration", "credithistory", "purpose",
                            "amount", "saving", "presentjob", "installmentrate",
                            "sexstatus", "otherdebtor", "resident", "property",
                            "age", "otherinstall", "housing", "ncredits", "job",
                            "npeople", "telephone", "foreign", "response")
German.credit$response <- ifelse(German.credit$response==1,0,1)
German.credit$response = as.factor(German.credit$response) # 2 = bad
table(German.credit$response)
```

```
##
##   0   1
## 700 300
```

```
# str(German.credit) # to see factors and integers, numerics

set.seed(1234)

n <- nrow(German.credit)
in.train <- sample(1:n,0.75*n)

train <- German.credit[in.train,]
test <- German.credit[-in.train,]
```

We want to try all 4 tree methods: single classification tree, bagging, random forest, boosting

1. Classification tree

(a) Full classification tree

Modify the setting in the following R chunk for "eval" to be eval=TRUE to see the results.

```
#####################
# construct full tree
#####################
library(tree)
```

```r
library(pROC)

set.seed(100)
fulltree=tree(response~.,train,split="deviance")
summary(fulltree)
par(mfrow=c(1,2))
plot(fulltree)
text(fulltree)
# print(fulltree)
fullpred=predict(fulltree,test,type="class")
testres = table(test$response,fullpred) # confusion matrix, rows=true, columns = predictions
print(testres)
1-sum(diag(testres))/(sum(testres)) # Classification error rate
predfulltree = predict(fulltree,test, type = "vector")
testfullroc=roc(test$response == "1", predfulltree[,2])
auc(testfullroc)
par(pty="s") # "s" generates a square plotting region
plot.roc(testfullroc,xlim=c(1,0),asp=1,print.auc=TRUE)
```

(b) Pruned classification tree

Modify the setting in the following R chunk for "eval" to be eval=TRUE to see the results.

```r
# prune the full tree
set.seed(1234)
fullcv=cv.tree(fulltree,FUN=prune.misclass,K=5)
par(mfrow=c(1,3))

par(pty="s")
plot(fullcv$size,fullcv$dev,type="b", xlab="Terminal nodes",ylab="misclassifications")
# print(fullcv)
prunesize=fullcv$size[which.min(fullcv$dev)]
prunetree=prune.misclass(fulltree,best=prunesize)
plot(prunetree,type="proportional")
text(prunetree,pretty=1)
predprunetree = predict(prunetree,test, type = "class")
prunetest=table(test$response,predprunetree)
print(prunetest)# rows are true; columns are predictions
1-sum(diag(prunetest))/(sum(prunetest))
predprunetree = predict(prunetree,test, type = "vector")
testpruneroc=roc(test$response == "1", predprunetree[,2])
auc(testpruneroc)
par(pty="s")
plot(testpruneroc,xlim=c(1,0),print.auc=TRUE)
```

Question 1: Why do we want to prune the full tree?

Answer:

2. Bagged trees

Modify the setting in the following R chunk for "eval" to be eval=TRUE to see the results.

```r
library(randomForest)
set.seed(1234)
bag=randomForest(response~., data=German.credit,subset=in.train,
```

```
                 mtry=20,ntree=500,importance=TRUE)
bag$confusion # for training data
yhat.bag=predict(bag,newdata=test)
misclass.bag=table(test$response,yhat.bag) # rows are true; columns are predictions
print(misclass.bag)
1-sum(diag(misclass.bag))/(sum(misclass.bag)) # test error rate
predbag = predict(bag,test, type = "prob") # to AUC of ROC curves
testbagroc=roc(test$response == "1", predbag[,2])
auc(testbagroc)

# make plots
layout(matrix(c(1,1,1),  ncol=3, byrow = TRUE),widths = c(1,4))
par(mfrow=c(1,3))
par(pty="s")
plot.roc(testbagroc,xlim=c(1,0),print.auc=TRUE)
varImpPlot(bag,pch=20,type=1)
varImpPlot(bag,pch=20,type=2)
```

Question 2: What is the main motivation behind bagging?

Answer:

3. Random forest

Question 3:

1. Plug in your code in the following R chunk for using random forest method to the train and make predictions for test, calculate ROC curves, etc.
2. The code will be similar to code for bagging method with only one parameter 'mtry' being different (use the number for 'mtry' suggested in the book or notes).
3. Please use your own name for the returned random forest model, predictions, etc.

Question 4: The value of the parameter `mtry` is the only difference between bagging and random forest. What does this parameter mean? What is the good effect of choosing `mtry` to be a value less than the number of covariates?

Answer:

4. Boosting

Modify the setting in the following R chunk for "eval" to be eval=TRUE to see the results.

```
#######################
# Boosting
#######################
library(gbm)
set.seed(1234)

train$response <- as.character(train$response)
boost=gbm(formula= response ~ ., distribution="bernoulli",
          data=train, n.trees=8000,shrinkage=0.001)
summary(boost,plot=FALSE)

par(pty="s")
par(mfrow=c(1,4))
plot(boost,i="purpose")
```

```r
plot(boost,i="amount")
plot(boost,i="checkaccount")
plot(boost,i="credithistory")
```

```r
library(gbm)
# make predictions
test$response <- as.character(test$response)
yhat.boost=predict(boost,newdata=test,n.trees=8000,type="response")
boost.pred <- ifelse(yhat.boost>=0.5,1,0)

misclass.boost <- table(test$response,boost.pred)
print(misclass.boost)
1-sum(diag(misclass.boost))/(sum(misclass.boost))

testrfroc=roc(test$response == "1",yhat.boost)
auc(testrfroc)
par(pty="s")
plot(testrfroc,xlim=c(1,0),print.auc=TRUE)
```

Question 5: What is the main difference between boosting and random forest (or bagging)?

Answer:

Question 6: Compare among the above 4 methods: classification tree, bagging, random forest, boosted trees, using the above results and/or plots, such as the test misclassification error rates, AUC, etc.

Answer: