Homework_3

Lindley Slipetz

7/15/2021

For this homework, I will again be using the Childhood adversity and traumatic stress among inpatients at a psychiatric hospital in the Baltimore area from 1993-1995. The data include diagnoses, psychological symptoms, physical and sexual abuse, post-traumatic stress disorder, self-destructive behavior, and demographic data. I will be predicting the occurence of a substance abuse disorder (both primary and secondary diagnoses) from gender, occurence of anxiety disorder, occurence of dissociative disorder, occurence of mood disorder, age, marital status, SES, social support as a child, and self-destructive behaviors (alcohol/drugs, eating, sexual impulsiveness, self-harm, and suicidality)

Let's load the data and the packages!

```
library(mlbench)
library(parallel)
#install.packages("doParallel")
library(doParallel)
library(foreach)
library(MASS)
library(ggplot2)
library(caret)
#install.packages("ranger")
library(ranger)
library(pROC)
#install.packages("party")
library(party)
library(dplyr)
#install.packages("ggraph")
library(ggraph)
library(igraph)
#install.packages("rpart.plot")
library(rpart.plot)
library(rpart)
library(rpart.plot)
#install.packages("gbm")
library(gbm)
library(dplyr)
library(ggraph)
library(igraph)
library(tidyverse)
full_data <- read.table(file = 'G:\\My Drive\\ICPSR\\ML\\HW_2\\36168-0001-Data.tsv', sep = '\t', header
```

Now, we're going to subset the data to just the variables of interest.

```
subset_data <- full_data %>%
  select(SUBDX, SEX, ANXDX, DISDX, MOODDX, AGE, MAR, SES, SSC, SISDB_TOT)
```

Now we're going to check if there's any missing data.

```
df <- as.data.frame(
  cbind(
    lapply(
      lapply(subset_data, is.na), sum)
    )
  )

rownames(subset(df, df$V1 != 0))
```

```
## [1] "SSC"       "SISDB_TOT"
```

Two columns have missing data. Let's see how much there is.

```
sum(is.na(subset_data$SSC))
```

```
## [1] 11
```

```
sum(is.na(subset_data$SISDB_TOT))
```

```
## [1] 10
```

That isn't too bad. Let's just omit the missing data.

```
complete_data <- na.omit(subset_data)
```

Making factors.

```
complete_data$SUBDX  <- factor(complete_data$SUBDX, labels=c("NSU","SU"))
```

Pre-process data.

```
set.seed(39846)
impute <- preProcess(complete_data, method=c("center","scale"))
complete_data <- predict(impute, complete_data)
```

Let's split the data.

```
set.seed(2964746)
trainIndex <- createDataPartition(complete_data$SUBDX, p=0.2, list = FALSE, times = 1)
train <- complete_data[trainIndex,]
test <- complete_data[-trainIndex,]
```

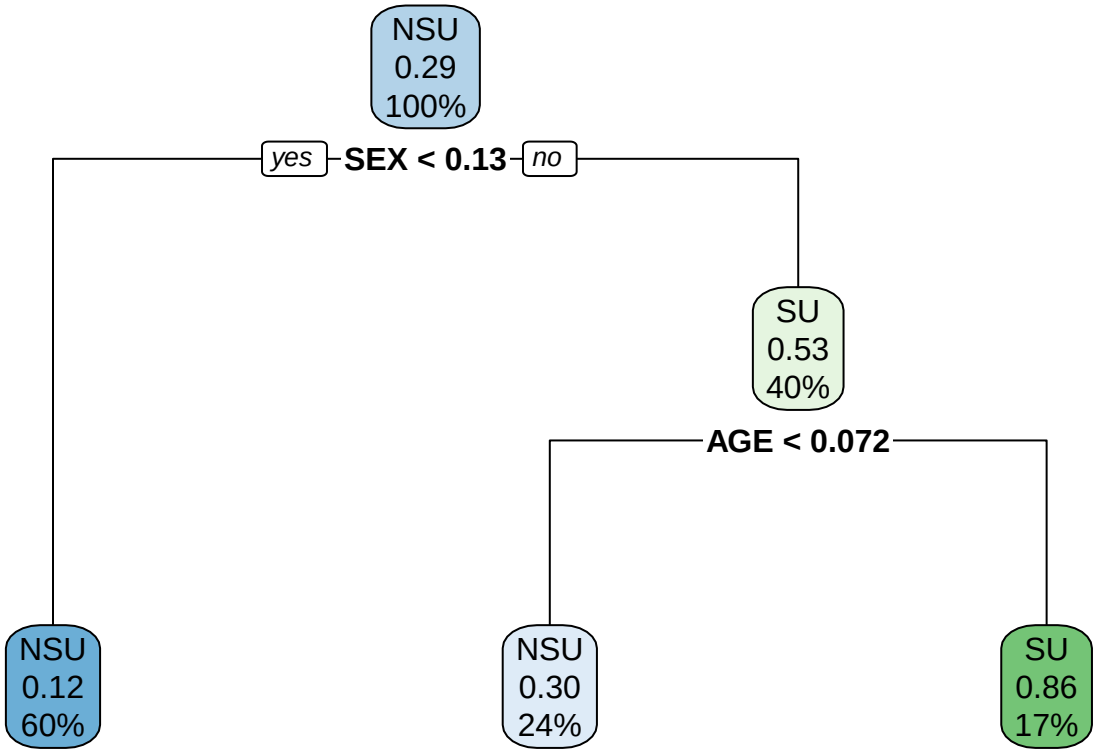
Single tree

Fitting classification tree.

```
set.seed(1985)
fit <- rpart(SUBDX ~ ., data = train,
  control = rpart.control(minsplit = 10, minbucket = 5))
```

Plotting tree.

```
rpart.plot(fit)
```



So it starts by splitting males to right and females to the left. Then it splits the males into ages centered on standardized at 0.072. The highest proportion of substance use is males older than the standardized score of 0.072.

```
train.pred <- predict(fit, train, type="class")
test.pred <- predict(fit, test, type="class")
#
confusionMatrix(as.factor(train$SUBDX), train.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NSU SU
##        NSU  29  1
##        SU    6  6
##
##                Accuracy : 0.8333
##                  95% CI : (0.6864, 0.9303)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 0.5991
##
##                   Kappa : 0.5333
##
##  Mcnemar's Test P-Value : 0.1306
##
##             Sensitivity : 0.8286
##             Specificity : 0.8571
```

```
##          Pos Pred Value : 0.9667
##          Neg Pred Value : 0.5000
##              Prevalence : 0.8333
##          Detection Rate : 0.6905
##    Detection Prevalence : 0.7143
##       Balanced Accuracy : 0.8429
##
##        'Positive' Class : NSU
##
```

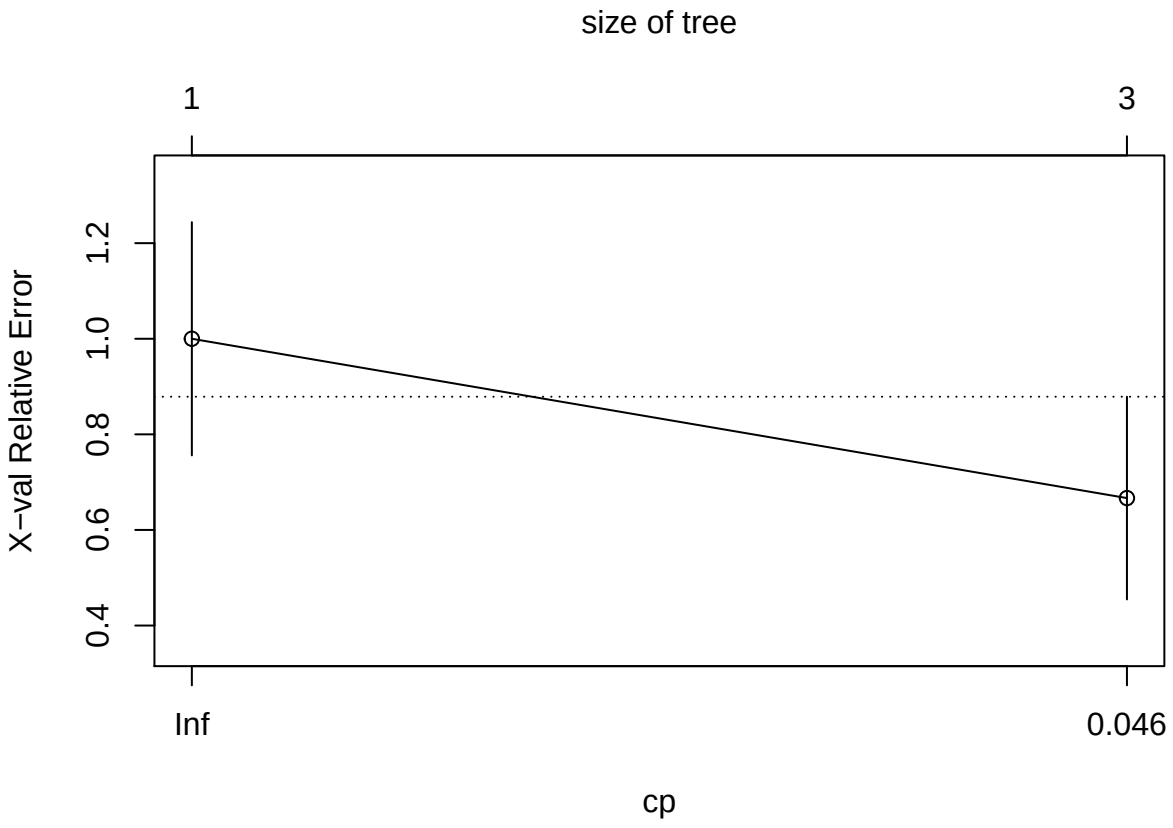
```
confusionMatrix(as.factor(test$SUBDX), test.pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NSU SU
##        NSU  96 22
##        SU   31 15
##
##                Accuracy : 0.6768
##                  95% CI : (0.5995, 0.7477)
##     No Information Rate : 0.7744
##     P-Value [Acc > NIR] : 0.9985
##
##                   Kappa : 0.1485
##
##  Mcnemar's Test P-Value : 0.2718
##
##             Sensitivity : 0.7559
##             Specificity : 0.4054
##          Pos Pred Value : 0.8136
##          Neg Pred Value : 0.3261
##              Prevalence : 0.7744
##          Detection Rate : 0.5854
##    Detection Prevalence : 0.7195
##       Balanced Accuracy : 0.5807
##
##        'Positive' Class : NSU
##
```

Accuracy is 83% for the training data and 68% for the testing data. That's not too good.

Pruning the tree.

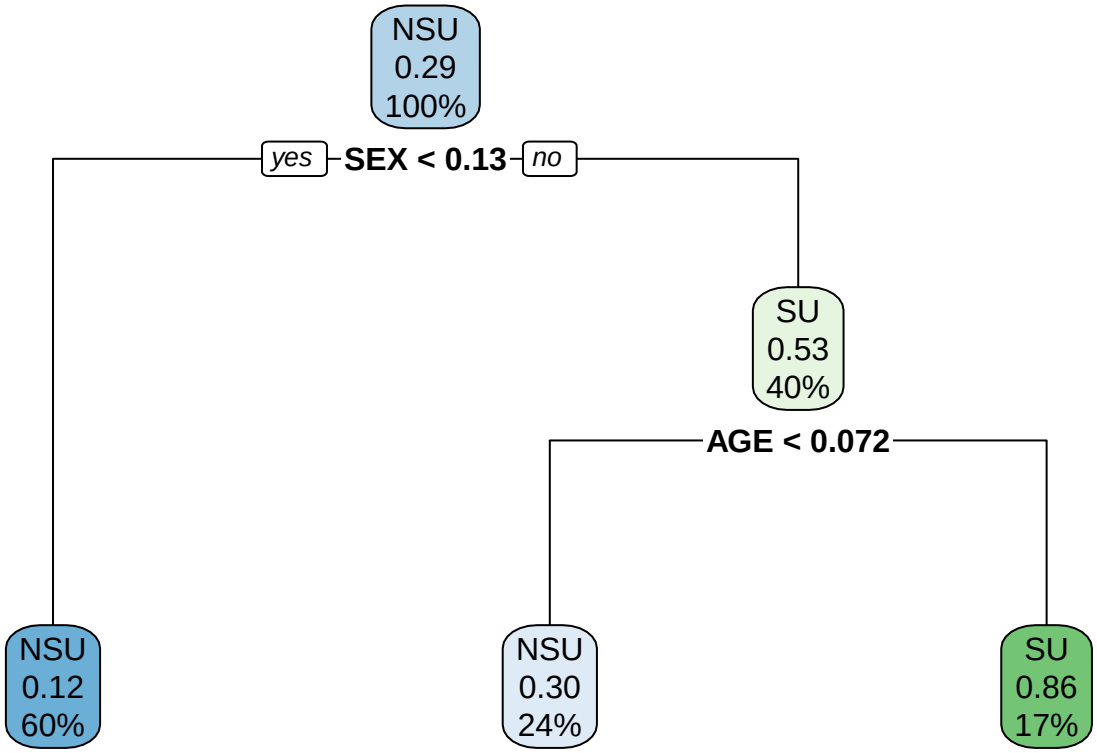
```
plotcp(fit)
```




```
pfit <- prune(fit, cp=fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])
```

Pruned tree plot.

```
rpart.plot(pfit)
```



Random forest

Parallel processing

```
cl <- makeCluster(detectCores() - 1, setup_timeout = 0.5)
registerDoParallel(cl)
```

Set control parameters

```
fitCtrl <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 2,
                        summaryFunction=twoClassSummary,
                        ## Estimate class probabilities
                        classProbs = TRUE,
                        ## Search "grid" or "random"
                        search = "random",
                        ## Use cluster
                        allowParallel = TRUE)
```

Testing grid.

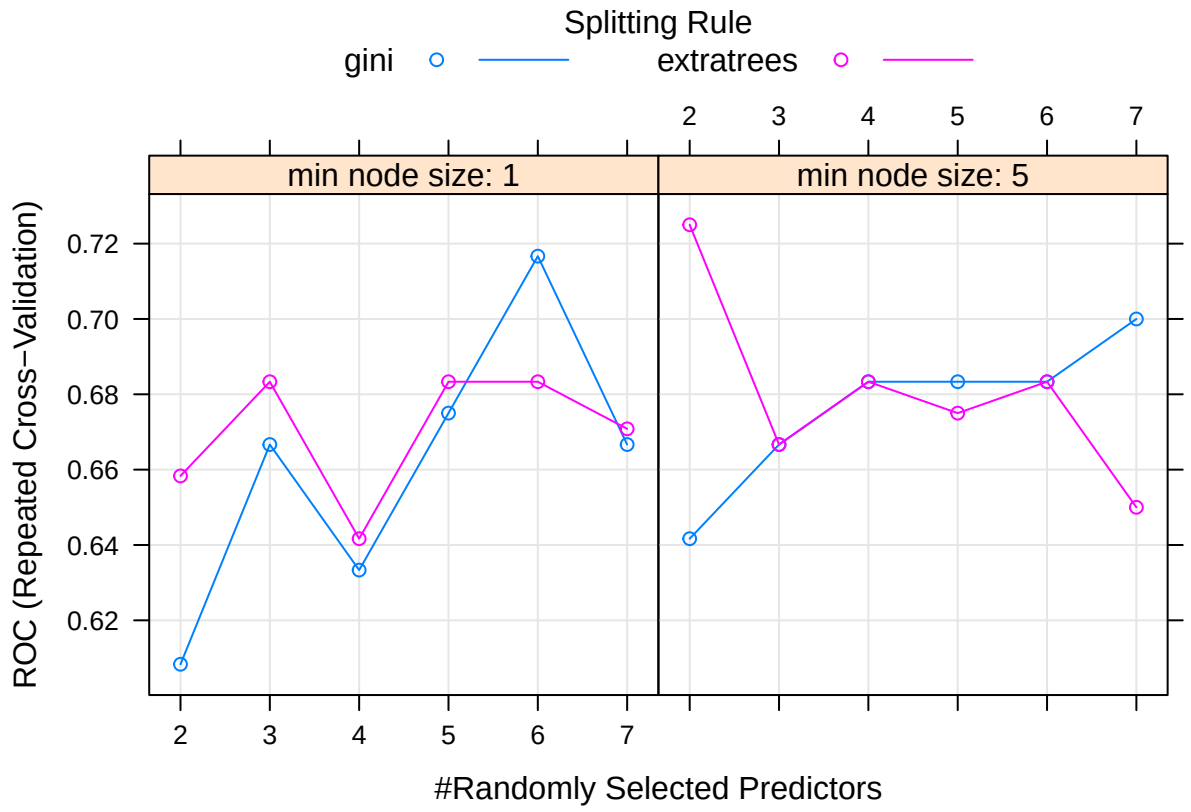
```
rfGrid <- expand.grid(mtry=c(2:7),
    min.node.size=c(1,5),
    splitrule=c("gini","extratrees"))
```

Forest.

```r
rf.res <- train(SUBDX ~ .,
    data=train,
    method="ranger",
    trControl=fitCtrl,
    tuneGrid=rfGrid,
    importance="impurity",
    num.trees=500,
    metric="ROC",
    verbose=FALSE)
rf.res
```

```
## Random Forest
##
## 42 samples
##  9 predictor
##  2 classes: 'NSU', 'SU'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 2 times)
## Summary of sample sizes: 38, 38, 38, 38, 38, 37, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  splitrule   ROC        Sens       Spec
##   2     1              gini        0.6083333  0.9500000  0.100
##   2     1              extratrees  0.6583333  0.9000000  0.050
##   2     5              gini        0.6416667  0.9333333  0.100
##   2     5              extratrees  0.7250000  0.9000000  0.000
##   3     1              gini        0.6666667  0.8833333  0.100
##   3     1              extratrees  0.6833333  0.8666667  0.275
##   3     5              gini        0.6666667  0.9000000  0.100
##   3     5              extratrees  0.6666667  0.8833333  0.125
##   4     1              gini        0.6333333  0.8833333  0.150
##   4     1              extratrees  0.6416667  0.8333333  0.275
##   4     5              gini        0.6833333  0.8833333  0.150
##   4     5              extratrees  0.6833333  0.8500000  0.225
##   5     1              gini        0.6750000  0.8666667  0.150
##   5     1              extratrees  0.6833333  0.8333333  0.325
##   5     5              gini        0.6833333  0.8666667  0.175
##   5     5              extratrees  0.6750000  0.8333333  0.325
##   6     1              gini        0.7166667  0.8666667  0.275
##   6     1              extratrees  0.6833333  0.8333333  0.325
##   6     5              gini        0.6833333  0.8833333  0.200
##   6     5              extratrees  0.6833333  0.8166667  0.325
##   7     1              gini        0.6666667  0.8833333  0.150
##   7     1              extratrees  0.6708333  0.8333333  0.325
##   7     5              gini        0.7000000  0.8833333  0.225
##   7     5              extratrees  0.6500000  0.8166667  0.325
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 2, splitrule = extratrees
##  and min.node.size = 5.
```

```
plot(rf.res)
```



ROC is maximized for 5 splits, and minimum node size of 5.

Confusion matrix.

```
confusionMatrix(predict(rf.res, train, type="raw"), train$SUBDX)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NSU SU
##        NSU  30  3
##        SU    0  9
##
##                Accuracy : 0.9286
##                  95% CI : (0.8052, 0.985)
##     No Information Rate : 0.7143
##     P-Value [Acc > NIR] : 0.0006489
##
##                   Kappa : 0.8108
##
##  Mcnemar's Test P-Value : 0.2482131
##
##             Sensitivity : 1.0000
##             Specificity : 0.7500
##          Pos Pred Value : 0.9091
```

```
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7143
##          Detection Rate : 0.7143
##    Detection Prevalence : 0.7857
##       Balanced Accuracy : 0.8750
##
##        'Positive' Class : NSU
##
```

```
confusionMatrix(predict(rf.res, test, type="raw"), test$SUBDX)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NSU  SU
##        NSU 115  38
##        SU    3   8
##
##                Accuracy : 0.75
##                  95% CI : (0.6765, 0.8142)
##     No Information Rate : 0.7195
##     P-Value [Acc > NIR] : 0.2185
##
##                   Kappa : 0.1934
##
##  Mcnemar's Test P-Value : 1.097e-07
##
##             Sensitivity : 0.9746
##             Specificity : 0.1739
##          Pos Pred Value : 0.7516
##          Neg Pred Value : 0.7273
##              Prevalence : 0.7195
##          Detection Rate : 0.7012
##    Detection Prevalence : 0.9329
##       Balanced Accuracy : 0.5742
##
##        'Positive' Class : NSU
##
```

Training data is predicted with 95% accuracy and the testing data is predicted with 76% accuracy.

ROC graphs

```
pred.train <- predict(rf.res, train, type="prob")[,"SU"]
roc(train$SUBDX ~ pred.train)
```

```
## Setting levels: control = NSU, case = SU
```

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

```
##
## Call:
## roc.formula(formula = train$SUBDX ~ pred.train)
##
## Data: pred.train in 30 controls (train$SUBDX NSU) < 12 cases (train$SUBDX SU).
## Area under the curve: 1
```

```
#
pred.test <- predict(rf.res, test, type="prob")[,"SU"]
roc(test$SUBDX ~ pred.test)
```

```
## Setting levels: control = NSU, case = SU
## Setting direction: controls < cases
```

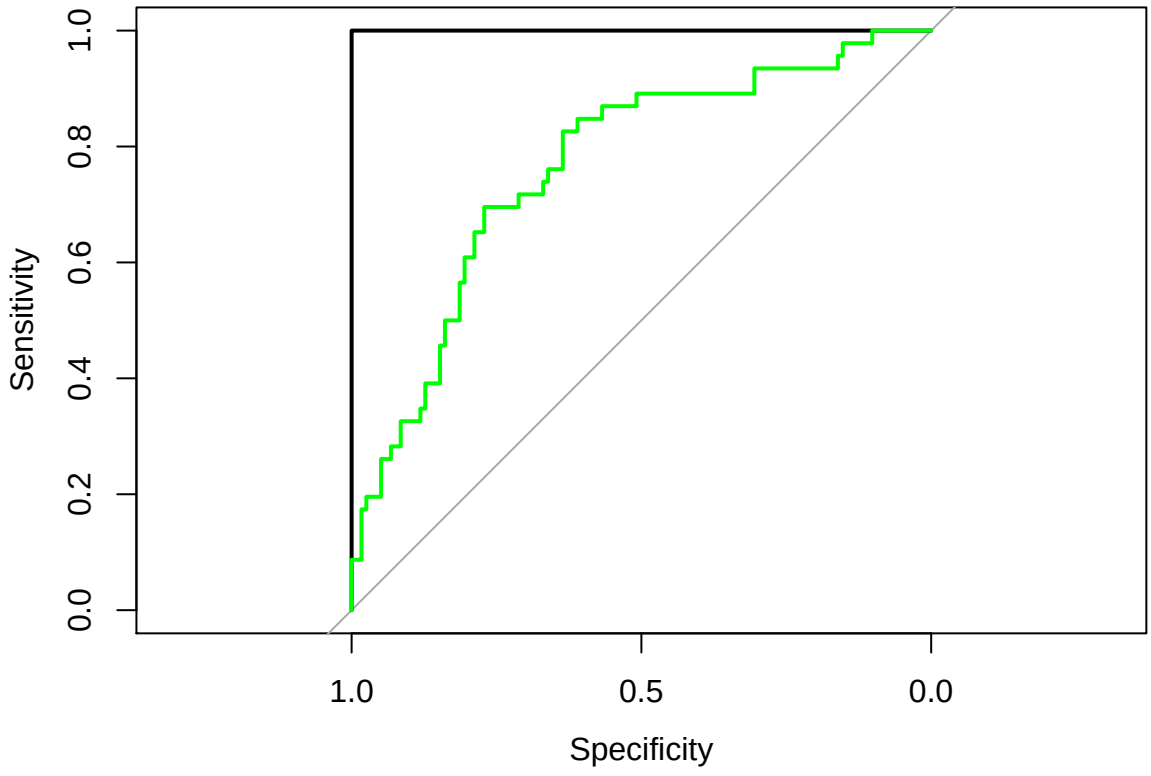
```
##
## Call:
## roc.formula(formula = test$SUBDX ~ pred.test)
##
## Data: pred.test in 118 controls (test$SUBDX NSU) < 46 cases (test$SUBDX SU).
## Area under the curve: 0.7651
```

```
#
plot.roc(train$SUBDX, pred.train)
```

```
## Setting levels: control = NSU, case = SU
## Setting direction: controls < cases
```

```
plot.roc(test$SUBDX, pred.test, add=TRUE, col="green")
```

```
## Setting levels: control = NSU, case = SU
## Setting direction: controls < cases
```



#GBM‘

Control.

```
fitCtrl_1 <- trainControl(method = "repeatedcv",
                          number = 5,
                          repeats = 2,
                          summaryFunction=twoClassSummary,
                          classProbs = TRUE,
                          search = "random",
                          ## Down-sampling
                          sampling = "smote",
                          allowParallel = TRUE)
```

Testing grid.

```
gbmGrid <-  expand.grid(n.trees = c(1:20)*100,
                          interaction.depth=c(2,3),
                          shrinkage = c(0.01, 0.05),
                          n.minobsinnode=5)
```

GBM

gbm.res <- train(SUBDX ~ .,

```
data=train,

method="gbm",

trControl=fitCtrl_1,

tuneGrid=gbmGrid,

bag.fraction=0.5,

metric="ROC",

verbose=FALSE)
```

gbm.res

plot(gbm.res)

It says I need "DMwR" to run this, but it's not compatible with my version of R (which is completely up to date). From here on out, I'll include the code; but, unfortunately, I can't run it.

Predictions

confusionMatrix(predict(gbm.res, train, type="raw"), train$SUBDX)

confusionMatrix(predict(gbm.res, test, type="raw"), test$SUBDX)

pred.train <- predict(gbm.res, train, type="prob")[,"SU"]

roc(train$SUBDX ~ pred.train)

pred.test <- predict(gbm.res, test, type="prob")[,"SU"]

roc(test$SUBDX ~ pred.test)

plot.roc(train$SUBDX, pred.train)

plot.roc(test$SUBDX, pred.test, add=TRUE, col="green")

Variable importance

gbmImp <- varImp(gbm.res)

```
plot(gbmImp)
```