CPTS315Report

Shane Sweeney

November 30, 2021

```
# install package e1071 if it is not installed and attach it, if it is
already installed then attach it
if(!require(e1071)){
    install.packages("e1071")
    library(e1071)
}
# repeat for ROCR
if(!require(ROCR)){
    install.packages("ROCR")
    library(ROCR)
}
# repeat for performaceEstimation
if(!require(performanceEstimation)){
    install.packages("performanceEstimation")
    library(performanceEstimation)
}
# repeat for randomForest
if(!require(randomForest)){
    install.packages("randomForest")
    library(randomForest)
}



# read in the csv file from kaggle. Highly correlated attributes have been
removed. There are 50 attributes and 1,163 observations.
sampleSLIdf <- read.csv(file = "all_data_R_high_cor_removed.csv")

# remove the column filename as it is unnecessary. I will also remove sex
because not every row had it filled in and I will remove the column corpus as
this just shows which study the sample came from. The rest of the columns are
numeric or integer type so SVM will function.
sampleSLIdf <- sampleSLIdf[,-c(1,2,4)]
# make group column a factor type so SMOTE and SVM will work properly
sampleSLIdf$group <- as.factor(sampleSLIdf$group)
#View(sampleSLIdf)
#str(sampleSLIdf$group)
#levels(sampleSLIdf$group)
#nrow(sampleSLIdf[sampleSLIdf$group=="SLI",])

####### Before I used the smote() method. You can see I was worried about how
to get an even division between SLI and TD ###############
```

```
# according to the Kaggle dataset description, only 23% of the samples of
children were SLI. I want to split the SLI and typically developing (TD)
children samples and then get an 80/20 ratio for training/test and combine
the two types of children back together. So I'll have a training group with
80% sampled My fear with there being much less SLI samples is that if I
randomly sample the whole data set to make a training set, the training set
may not have a lot of SLI children in it to make a good prediction model.

#onlySLIdf <- sampleSLIdf[sampleSLIdf$group=="SLI",]
#onlyTDdf <- sampleSLIdf[sampleSLIdf$group=="TD",]

# randomly select 133 rows in onlySLIdf
#set.seed(123)
#trainSLI <- sample(1:nrow(onlySLIdf), size=133, replace=FALSE)

# use the vector to select 133 rows for training data
#trainSLIdf <- onlySLIdf[trainSLI, ]
# select other rows for test data
#testSLIdf <- onlySLIdf[-trainSLI,]

# randomly select 448 rows in onlyTDdf
#set.seed(123)
#trainTD <- sample(1:nrow(onlyTDdf), size=448, replace=FALSE)

# use the vector to select 448 rows for training data
#trainTDdf <- onlyTDdf[trainTD, ]
# select other rows for test data
#testTDdf <- onlyTDdf[-trainTD,]

#trainingDF <- rbind(trainSLIdf, trainTDdf)
#testDF <- rbind(testSLIdf, testTDdf)

#############################################################################
#########

# 267 SLI samples and 896 TD samples before using smote()
nrow(sampleSLIdf[sampleSLIdf$group=="SLI",])

## [1] 267

nrow(sampleSLIdf[sampleSLIdf$group=="TD",])

## [1] 896

# Perform smote, tried to have the TD samples be undersampled and the SLI
samples oversampled.
# perc.over=3 and perc.under=1 was the best combination I could get to get
the SLI and TD samples closer to 50/50
SmotesampleSLIdf <- smote(group~., sampleSLIdf, perc.over = 3, k=5,
perc.under=1)
# After using smote, 1068 SLI samples and 801 TD samles with perc.over=3 and
perc.under=1
# When perc.over=9 and perc.under=1, 2670 SLI samples and 2403 TD samples,
which though closer to 50/50 did not give as accurate models
nrow(SmotesampleSLIdf[SmotesampleSLIdf$group=="SLI",])
```

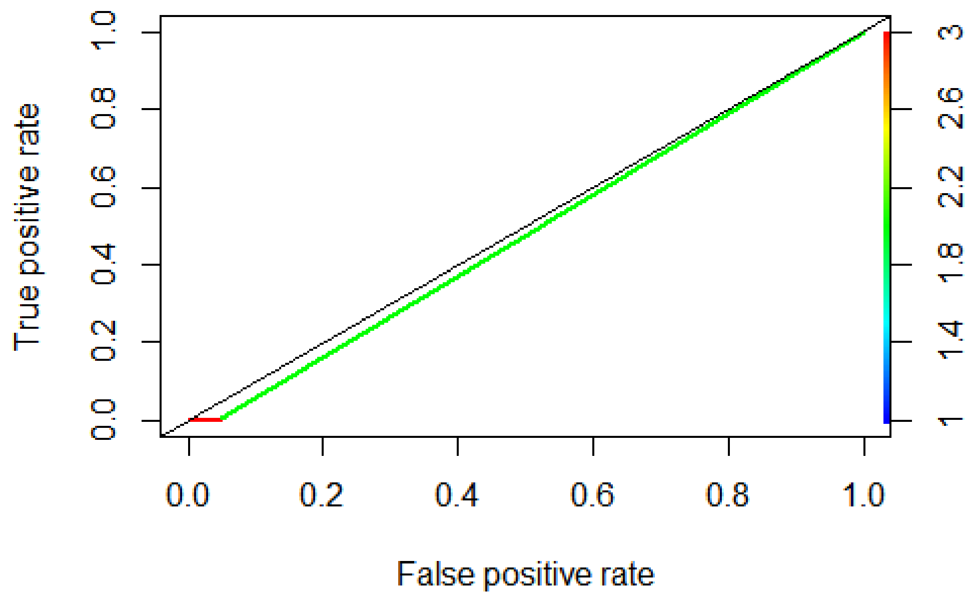
```
## [1] 1068
```

```
nrow(SmotesampleSLIdf[SmotesampleSLIdf$group=="TD",])
```

```
## [1] 801
```

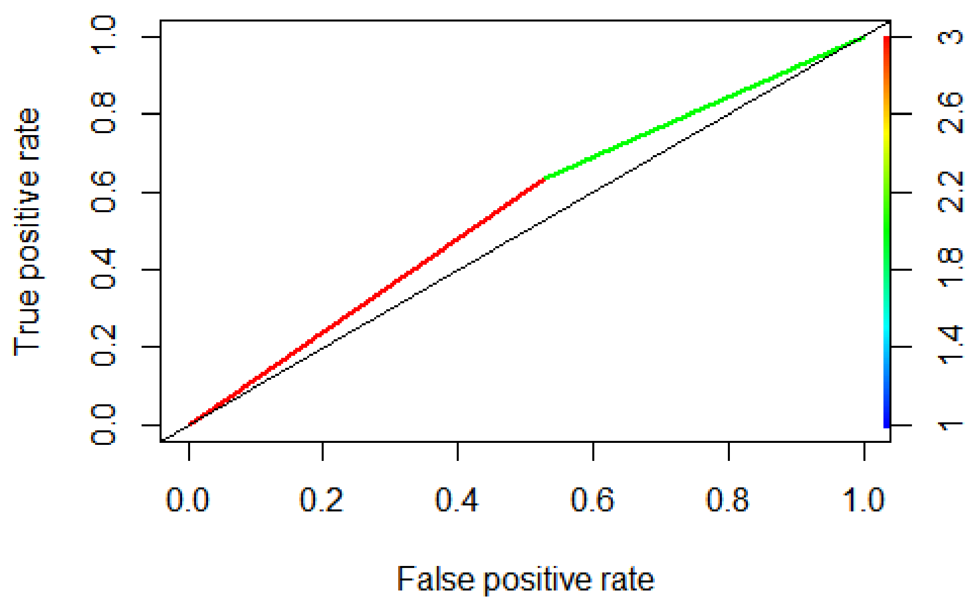
```
#nrow(SmotesampleSLIdf[SmotesampleSLIdf$group=="NA",])

# randomly select 1495 rows for training data (this will stick with the
generally suggested 80/20 training/testing ratio)
set.seed(123)
trainingVector <- sample(1:nrow(SmotesampleSLIdf), size=1495, replace=FALSE)

# create a training dataframe
trainingDF <- SmotesampleSLIdf[trainingVector,]
#nrow(trainingDF)
# create a test dataframe
testDF <- SmotesampleSLIdf[-trainingVector,]
#nrow(testDF)

#### Before I used the smote() method
################################################
# The training and testing data samples seem to hold around the original
ratio of SLI to TD children which is 23% to 77% from the Kaggle dataset
#nrow(trainingDF) #930 rows total
#nrow(trainingDF[trainingDF$group=="SLI",]) #221 rows are SLI, 24%

#nrow(testDF) #233 rows total
#nrow(testDF[testDF$group=="SLI",]) #46 rows are SLI, 20%
##############################################################################
#########

# standardize training data as SVM can be affected by the scale of features.
Also remove the group column which holds the labels SLI and TD
standardized.trainingDF = scale(trainingDF[,-2],center=TRUE,scale=TRUE)
# set standardized.trainingDF to data frame
standardized.trainingDF <- as.data.frame(standardized.trainingDF)

# add group column back to standardized.trainingDF
standardized.trainingDF <- cbind(standardized.trainingDF, group =
trainingDF$group)
#str(standardized.trainingDF$group)
```

```r
# perform 10-fold cross validation to find best cost with linear kernel, best
cost is 5
# Commented out to save on run time
#set.seed(123)
#linearTuneResult=tune(svm,group~.,
data=standardized.trainingDF,kernel="linear", ranges=list(cost=c(0.01, 0.1,
1,5,10,50)))
#summary(linearTuneResult)

# perform linear svm (best cost came out to 5) (commented out above as it
takes a couple minutes to run)
svmlinear=svm(group~.,
data=standardized.trainingDF,kernel="linear",cost=5,scale=FALSE)
#summary(svmlinear)
# get training accuracy
svmPredlinearTA <- predict(svmlinear, standardized.trainingDF, type="class")
# 81.6% training accuracy
table(svmPredlinearTA, standardized.trainingDF$group)

##
## svmPredlinearTA SLI  TD
##             SLI 762 146
##             TD  101 486

# use svm model to predict for test
svmPredlinear <- predict(svmlinear, testDF, type="class")
#is.vector(svmPredlinear)
# confusion matrix, testing accuracy of 61.5%
table(svmPredlinear, testDF$group)

##
## svmPredlinear SLI  TD
##           SLI 195 168
##           TD   10   1

# get ROC curve
predLinear <- prediction(as.numeric(svmPredlinear), as.numeric(testDF$group))
roc = performance(predLinear,"tpr","fpr")
plot(roc, colorize = T, lwd = 2)
abline(a = 0, b = 1)
```



```
# AUC = 0.49
auc = performance(predLinear, measure = "auc")
print(auc@y.values)

## [[1]]
## [1] 0.4785683

# perform 10-fold cross validation to find best cost with radial kernel, best
cost is 10 and gamma 2
# For some of the runs this tune() function was giving me an infinite time
wheel so I've commented it out
#set.seed(123)
#radialTuneResult=tune(svm,group~.,
data=standardized.trainingDF,kernel="radial", ranges=list(cost=c(0.01, 0.1,
1,5,10,50),gamma=c(0.5,1,2,3,4)))
#summary(radialTuneResult)

# perform radial svm
svmradial=svm(group~.,
data=standardized.trainingDF,kernel="radial",gamma=2,cost=10,scale=FALSE)
#summary(svmradial)
# get training accuracy
svmPredradialTA <- predict(svmradial, standardized.trainingDF, type="class")
#  training accuracy 100% (most likely overfitting)
table(svmPredradialTA, standardized.trainingDF$group)
```

```
##
## svmPredradialTA SLI  TD
##             SLI 863   0
##             TD    0 632

# use svm model to predict
svmPredradial <- predict(svmradial, testDF, type="class")
#View(svmPredradial)
# confusion matrix, testing accuracy of 61.2% (really I am getting all
predictions as SLI is all)
table(svmPredradial, testDF$group)

##
## svmPredradial SLI  TD
##           SLI 205 169
##           TD    0   0

predRadial <- prediction(as.numeric(svmPredradial), as.numeric(testDF$group))
# ROC curve can't be created here because there are no TD predictions
#roc = performance(svmPredradial,"tpr","fpr")
#plot(roc, colorize = T, lwd = 2)
#abline(a = 0, b = 1)
# AUC = 0.5
auc = performance(predRadial, measure = "auc")
print(auc@y.values)

## [[1]]
## [1] 0.5

# without tuning number of trees and maxnodes AUC = 0.58
set.seed(125)
rfResult <- randomForest(group~., data=standardized.trainingDF,
importance=TRUE)
# I trialed different ntree and maxnodes combinations and ntree=450 and
maxnodes=40 gave me the highest AUC of 0.61
# even though I set the seed for the below randomForest model, it's results
were not coming back consistently
# So I am leaving the randomForest model above which has no tuning metrics
#set.seed(123)
#rfResult1 <- randomForest(group~., data=standardized.trainingDF,
importance=TRUE, ntree=500, maxnodes=50)
# 89.36% training accuracy when perc.over=3 and perc.under=1
# 90.3% training accuracy when perc.over=9 and perc.under=1
# 85% training accuracy without SMOTE
rfResult

##
## Call:
##  randomForest(formula = group ~ ., data = standardized.trainingDF,
importance = TRUE)
##                Type of random forest: classification
##                       Number of trees: 500
## No. of variables tried at each split: 6
##
##          OOB estimate of  error rate: 5.28%
## Confusion matrix:
##     SLI  TD class.error
## SLI 836  27  0.03128621
## TD   52 580  0.08227848

rfPred <- predict(rfResult, testDF, type="class")

# 53% test accuracy when perc.over=3 and perc.under=1
# 52.22% test accuracy when perc.over=9 and perc.under=1
# 52% test accuracy without SMOTE
table(rfPred, testDF$group)

##
## rfPred SLI  TD
##    SLI  97  62
##    TD  108 107

predrfForROC <- prediction(as.numeric(rfPred), as.numeric(testDF$group))
roc = performance(predrfForROC,"tpr","fpr")
plot(roc, colorize = T, lwd = 2)
abline(a = 0, b = 1)
```



```
# AUC = 0.58 when perc.over=3 and perc.under=1, this is generalized as a
"poor" classifier score for a model but was the highest AUC I could get
auc = performance(predrfForROC, measure = "auc")
print(auc@y.values)

## [[1]]
## [1] 0.5531534

#NNresult <- neuralnet(group~., data=standardized.trainingDF)
#NNresult

#NNpred <- predict(NNresult, testDF)
#NNpred

#table(testDF$group=="SLI", NNpred[, 1] > 0.89)

#set.seed(123)
#NNresult <- nnet(group~., data=standardized.trainingDF, size=1)

#NNpred <- predict(NNresult, testDF, type="class")

#table(NNpred, testDF$group)
```