STP Project

Nathan Young & Jacob Duchesne

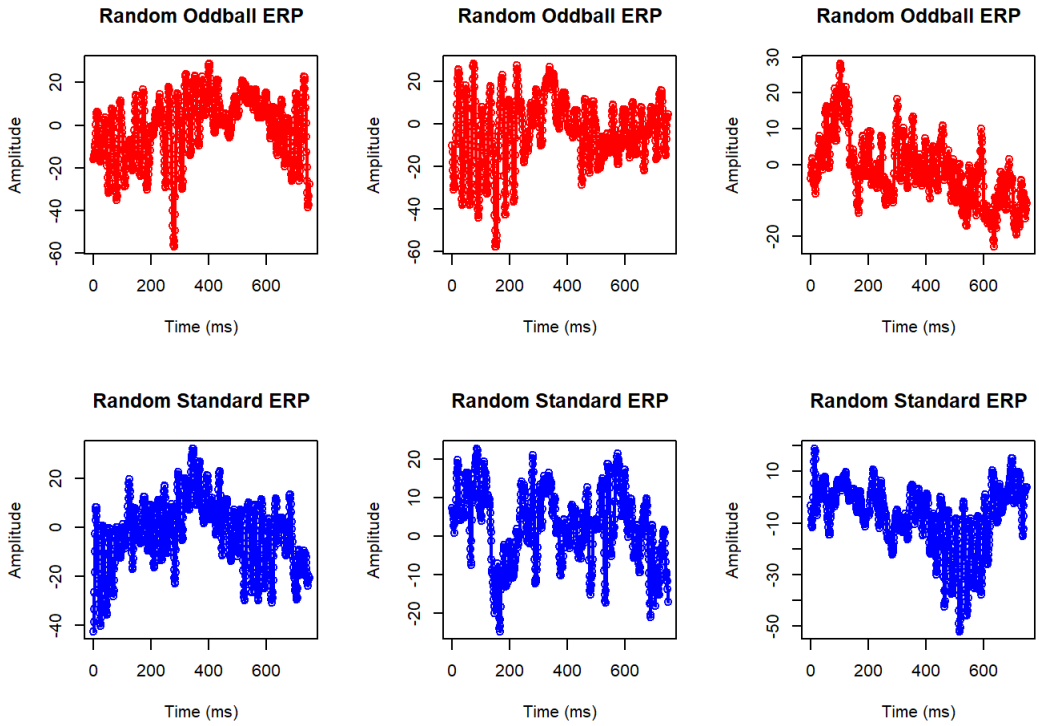
12/11/2019

Random Forest and Boosting on Electroencephalography Data to Predict Tone Type

Objective

The objective of this project was to create a model using random forest and boosting ensemble methods that could successfully predict whether a participant was listening to an “oddball” or a “standard” tone using single-trial electroencephalography data.

Single-trial EEG data is notoriously variable. Here are some randomly selected EEG trials from the data to demonstrate the variability.



Methods

The data used in this project is from an auditory oddball paradigm. Data was recorded from the PZ channel in a 32-channel EEG cap. In each trial of the experiment, one of two tones was presented: a standard tone and an oddball tone. The presentation of a tone constituted one trial, and Each participant (N = 22) participated in 900 trials, and there was a 4:1 ratio for standards to oddballs

All trials were converted into a matrix of time series, and an additional variable was created to represent the tone type (1 = oddball tone, 0 = standard tone). Each trial was 750 milliseconds in length, thereby making each variable in the time series equivalent to a millisecond (e.g., the value at V300 represents the measure of amplitude at 300 milliseconds after the onset of the stimulus). The randomForest and gbm functions in R were used for the random forest and boosting models, respectively.

Because the data were imbalanced, new oddball trials were synthetically generated using the SMOTE function. Given the nature of the dataset, oversampling (rather than undersampling) was used to match the number of oddball trials with the amount of standard-tone observations.

75% of the observations in the dataset were randomly assigned to the training set, and the remaining 25% of the observations were assigned to the test set. A grid search was run on two hyperparameters for the random forest models: mtry (the number of variables randomly selected to create each component tree model in the forest) and ntree (the number of tree models included in the forest). Once the optimal hyperparameter values from the grid search were determined (i.e., after determining the model with the highest AUC value), another grid search on variable importance was run to determine if variable reduction produced better results.

A grid search search was also run for the boosting ensembles, this time on three hyperparameters: tdepth (the number of splits for each component tree model), ntree (the number of tree models included in the boosting ensemble), and shrink (the “learning rate” of the boosting model). Because of the marginal benefits produced by variable reduction with random forest, we decided not to run a grid search on variable reduction for the boosting model. Given unlimited computational power and time, however, we would expect to see similar marginal

benefits.

We then compared the AUC value of the best random forest model with the AUC value of the best boosting model. We used AUC values as our primary measure of accuracy throughout the project.

Results

Despite the high variance in the single-trial EEG data, the random forest model (mtry = 27, ntree = 500) produced an AUC value of ~0.869. The boosting model (tdepth = 2, ntree = 250, shrink = .01) produced an AUC value of ~0.680.

Discussion

The ability to accurately classify states of experience at the single-trial level has profound practical applications. For instance, neurofeedback treatments involve immediate feedback from computer-based analyses of EEG data. This data is dynamic and adaptive in that it adjusts its sound or visual signals in an attempt to “reorganize” or “direct” brain activity. The overarching goal of neurofeedback treatments are for individuals to regulate and improve overall brain function and ameliorate symptoms. These treatments are increasingly used with individuals that suffer from seizure conditions, behavior disorders, attention deficits, developmental delays, acquired brain injuries, anxiety, depression, post-traumatic stress disorder, and insomnia.

The success of this model has promising implications for other fields, such as diagnostic testing. For instance, EEG has been implicated as a tool to diagnose autism. A future project would be to test whether random forest or boosting ensemble model could successfully classify whether an individual has autism by analyzing EEG data. There is also no obvious reason why these models would be limited to EEG data. By incorporating measures that utilize genetic, behavioral, physiological, and other types of information, the diagnostic power provided by these models could have profound use in fields such as medicine.

The variable importance determined by random forest models also provide interesting suggestions for research in cognitive neuroscience, where EEG is one of the most common modalities used to measure cognitive function. Our model found that amplitude values around the 300 millisecond range were highest in variable importance. This is consistent with a processing phenomenon known as the P300, whereby there is increased positive amplitude following an oddball stimulus. The P300 waveform is already established, but using random forest models to assess variable importance in more nuanced experimental paradigms could provide interesting suggestions that we otherwise were not able to detect.

Limitations & Challenges

During the course of the project lifespan we encountered a few limitations that affected our available scope and variable choice. While we used a tolerable and acceptable number of trees for our random forest and boosting we were forced to use less than ideal numbers due to some factors. These factors included processing power of the available computers and time available. If we had stronger processing computers and an infinite amount of time to run the analysis we determined that we would possibly achieve better results. For example, we decided to run our random forest and boosting with the following factors:

Random Forests mtry (Number of Variables Randomly Selected) | 27, 50 ntree (Number of Trees) | 50, 100 **Boosting** idv (Tree Depth) | 1,2 ntv (Number of Trees) | 50, 100 shv (Shrink Rate) | 0.01, 0.0001

One hypothesis we have for why random forest outperformed boosting was that boosting did not have a sufficient amount of decision trees to average over. Our most successful random forest model used ntree = 500, whereas our boosting models were limited to ntree = 250. This was a result of limited computer processing power. However, given additional computing power and time we would have prefered our ensembles to run closer to the values depicted below. We predict that a boosting model with small to intermediate number of splits, high numbers of trees, and a relatively slow learning rate could perform on par with our random forest model, if not better.

Random Forests mtry (Number of Variables Randomly Selected) | 5, 27, 100 ntree (Number of Trees) | 100, 1000, 5000, 7500, 10000 **Boosting** idv (Tree Depth) | 1, 2, 3 ntv (Number of Trees) | 100, 1000, 5000, 7500, 10000 shv (Shrink Rate) | 0.1, 0.01, 0.001, 0.0001, 0.00001

R Script

Import the libraries we’ll be using.

Each participant (N = 22) was presented with 720 standard trials and 180 oddball trials. Import the data for all 22 participants:

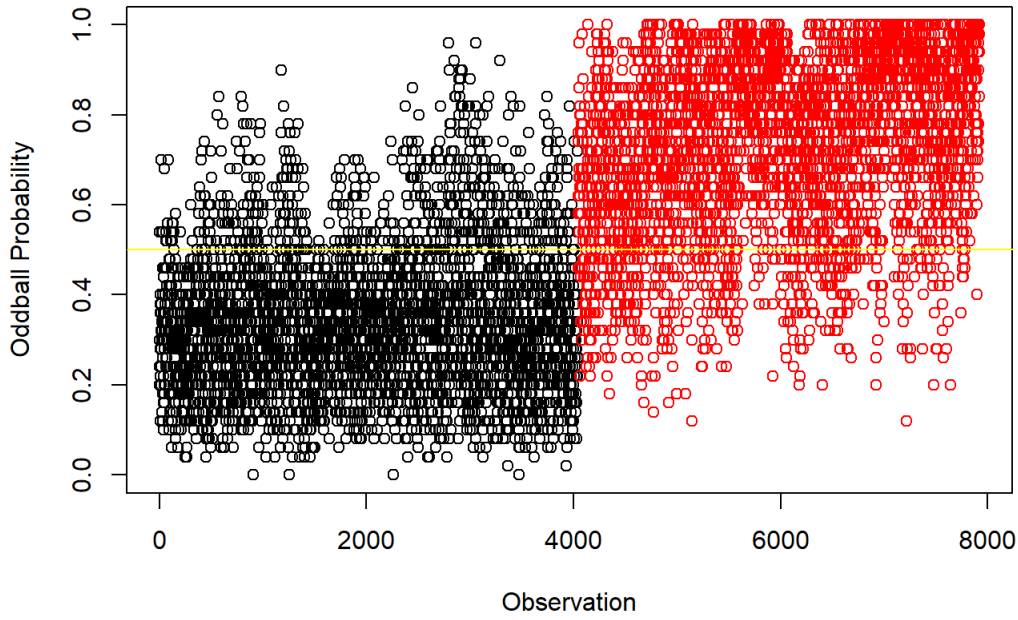
```
## There are 15840 standard trials and 3960 trials.
```

The data is clearly unbalanced. Use the SMOTE function to generate new examples that are representative of the oddballs in the data.

Randomly select 75% of the data for training. The remaining 25% will be used for testing.

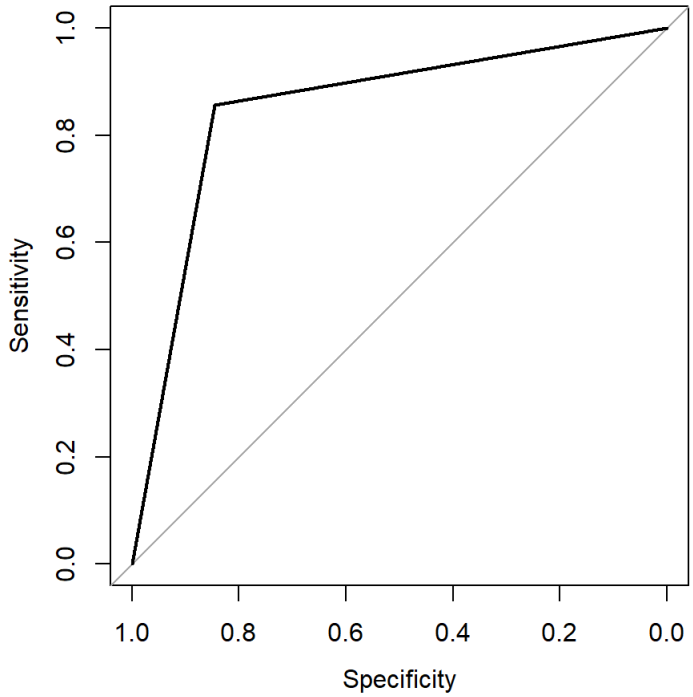
Let’s try a random forest model with ntree = 50.

Let’s plot the predictions. Observations in black are standard tones, and observations in red are oddball tones. If the probability was greater than .5, the model predicted it was an oddball.



Let's print the confusion matrix and plot the ROC curve.

```
##           Actual
## Predicted    0    1
##         0 3419  558
##         1  626 3317
```




```
## Area under the curve: 0.8506
```

Not bad! Let's establish a matrix of values for a grid search. We will be testing different combinations of the hyperparameters mtry (the number of variables randomly selected for each tree model) and ntrees (the number of tree models our random forest will run).

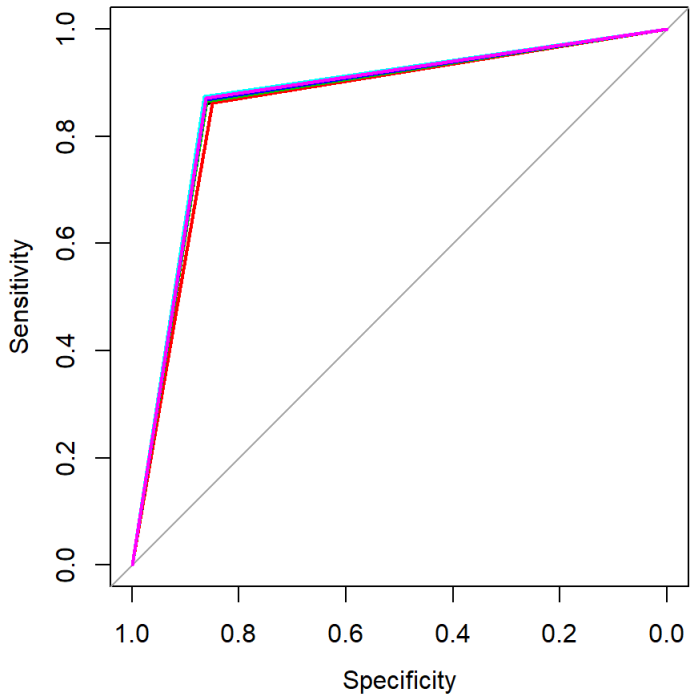
```
##   mtry ntree
## 1   27   100
## 2  100   100
## 3   27   250
## 4  100   250
## 5   27   500
## 6  100   500
```

Time to run the grid search! We will store the predictions and probabilities.

Let's view the AUC's for each of the models in the grid search.

```
## AUC for  1 : 0.8628427
## AUC for  2 : 0.8563239
## AUC for  3 : 0.8634347
## AUC for  4 : 0.8653431
## AUC for  5 : 0.8694288
## AUC for  6 : 0.8666497
```

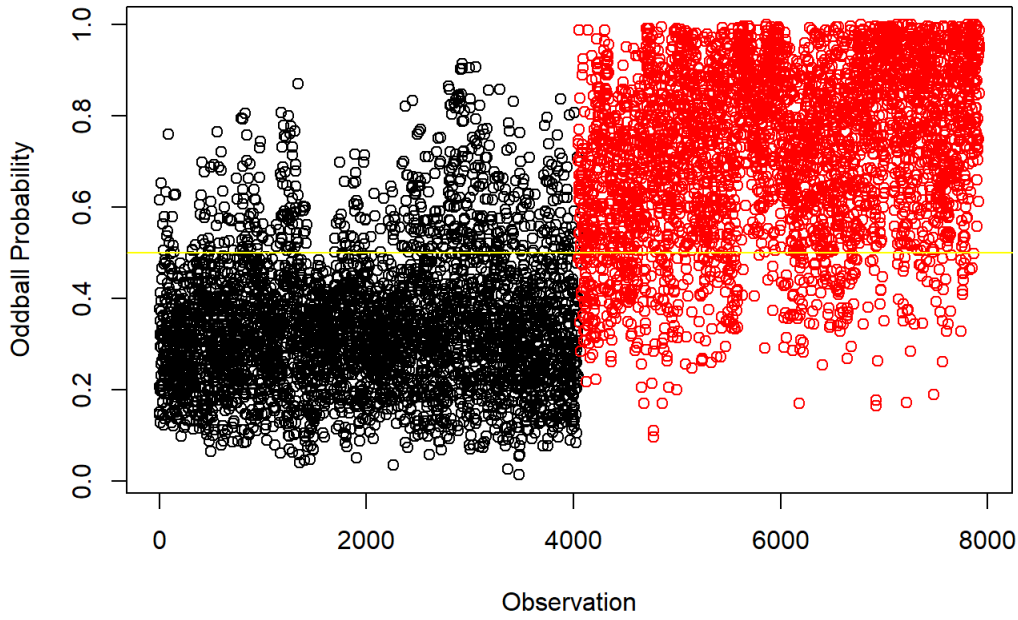
Then plot the ROC curves for each.



Which model has the best AUC?

```
## The best AUC is from model:  5 , with a value of  0.8694288
```

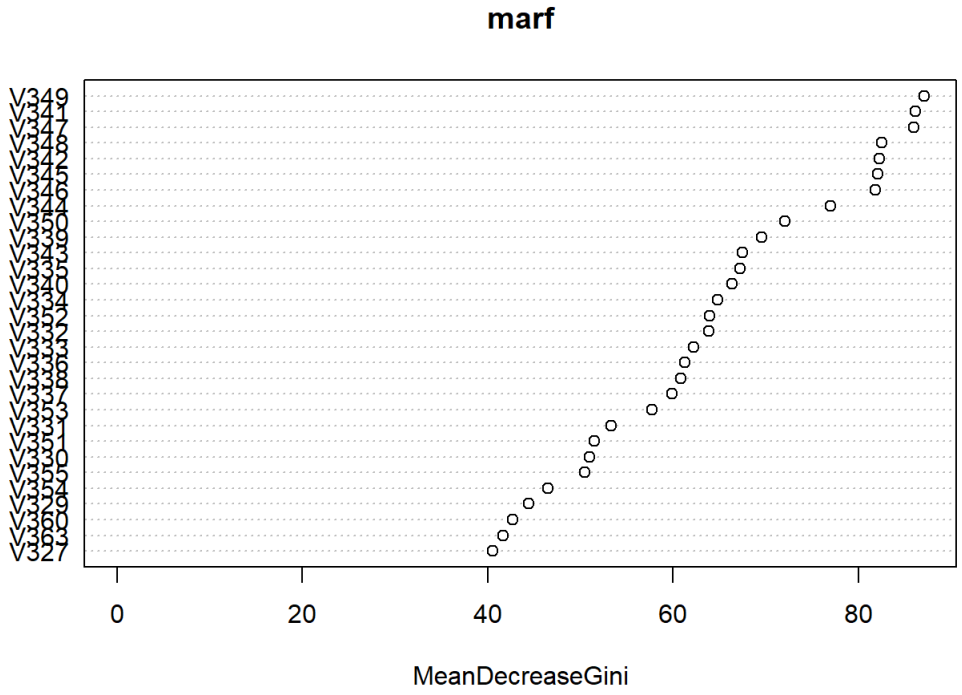
Let's take this model, plot the probabilities, and print a confusion matrix to see how well it did on the testing data.




```
##          Actual
## Predicted    0    1
##         0 3494  501
##         1  551 3374
```

Let's plot the variable importance to see what time points the model considered more "important" for making its prediction.

```
varImpPlot(marf)
```



As we can see from the plot, the time points around 300 milliseconds were considered to be the most important variables for the model. Interestingly, this is consistent with a cognitive phenomenon studied extensively in psychology research. During the processing of an oddball stimulus, there is significantly higher amplitude around 300 milliseconds relative to the ERP waveform in a standard tone. This phenomenon is known in psychology as the P300 waveform. The variable importance plot shows results consistent with this.

It's possible that some of these variables are ultimately making the model worse. Let's do a grid search on different variable importance values to see if eliminating unimportant variables makes the model more accurate. Establish the settings:

Then run models, which progressively sets a higher and higher threshold for variable importance in order to be included in the model.

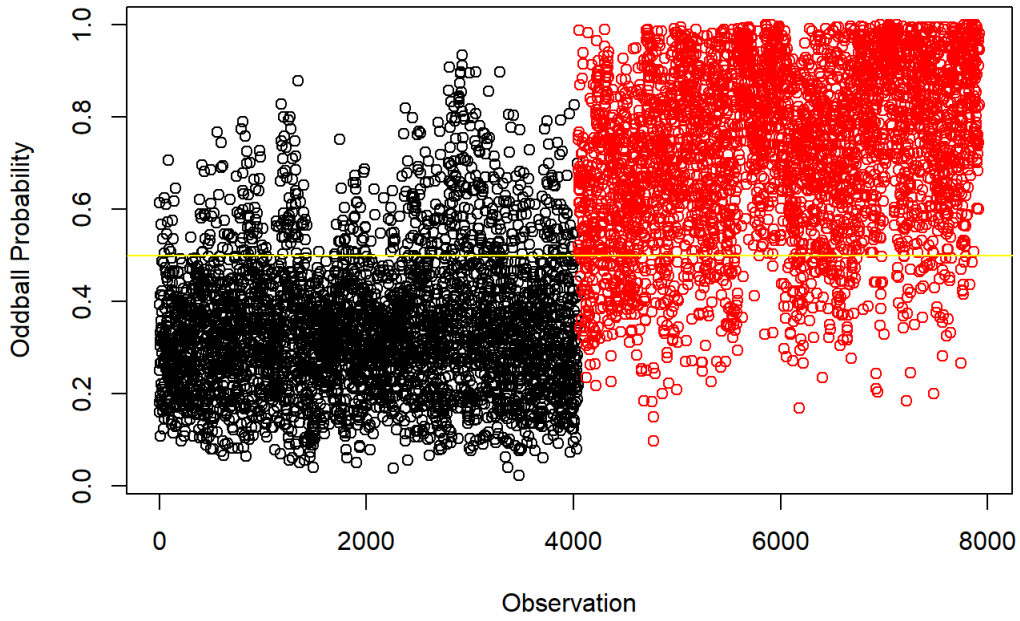
Let's view the AUC's for each of the models in the grid search.

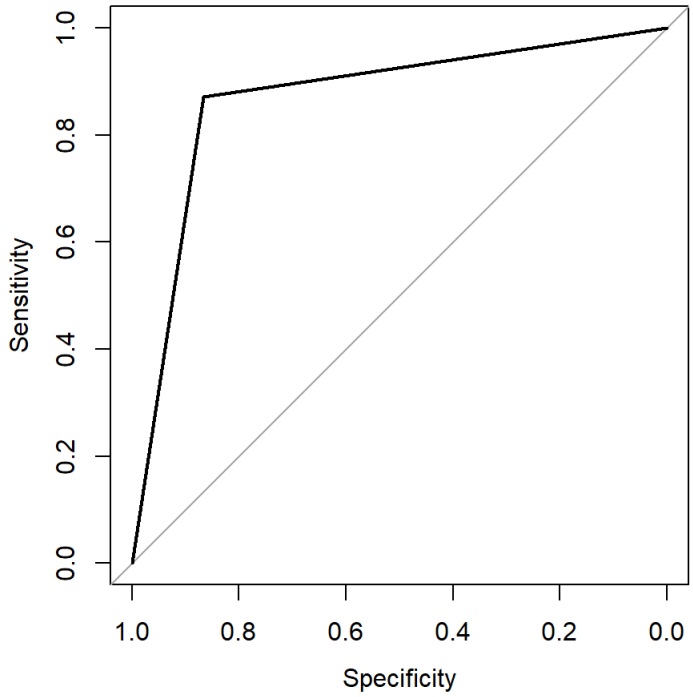
```
## AUC for  1 : 0.8674336
## AUC for  2 : 0.8685147
## AUC for  3 : 0.8256144
## AUC for  4 : 0.7975527
## AUC for  5 : 0.7972307
## AUC for  6 : 0.7932795
## AUC for  7 : 0.798843
## AUC for  8 : 0.7973489
## AUC for  9 : 0.7973163
## AUC for  10 : 0.7957419
## AUC for  11 : 0.7994611
```

Which of the reduced-variable models has the best AUC?

```
## The best AUC is from model:  2 , with a value of  0.8685147
```

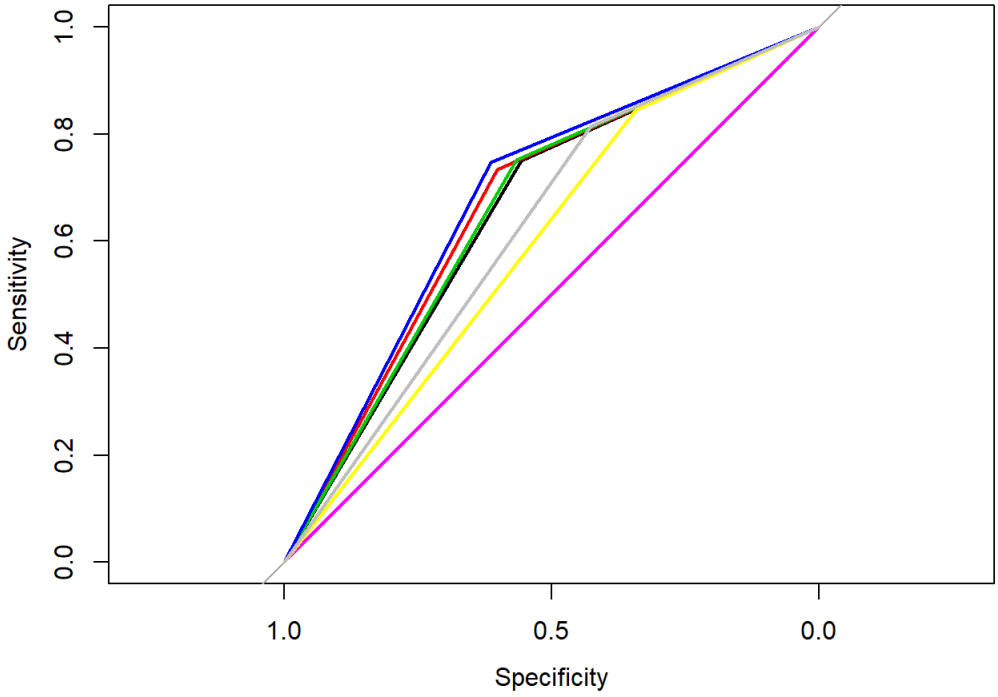
Let's plot the final probabilities, ROC curve, and confusion matrix!






```
##          Actual
## Predicted    0    1
##         0 3502  501
##         1  543 3374
```

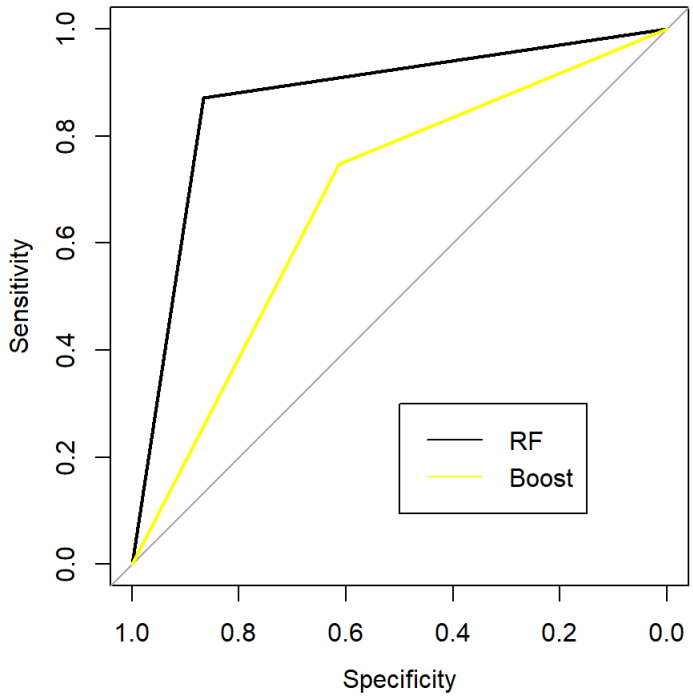
```
##   tdepth ntree shrink
## 1      1   100  1e-02
## 2      2   100  1e-02
## 3      1   250  1e-02
## 4      2   250  1e-02
## 5      1   100  1e-04
## 6      2   100  1e-04
## 7      1   250  1e-04
## 8      2   250  1e-04
```



```
## AUC for  1 : 0.6529653
## AUC for  2 : 0.667704
## AUC for  3 : 0.658673
## AUC for  4 : 0.6801051
## AUC for  5 : 0.5
## AUC for  6 : 0.5
## AUC for  7 : 0.5932581
## AUC for  8 : 0.6203121
```

Which of the reduced-variable models has the best AUC?

```
## The best AUC is from model:  4 , with a value of  0.6801051
```




```
## Random Forest AUC: 0.8682349
##  Boosting AUC:  0.6801051
```

Future Work

We saw in class that neural networks can consistently outperform both random forests and boosting; if this project was taken further, we would like to apply a neural network to the data in order to see it's performance compared to our methods. Additionally, the values for the hyperparameters used in our grid search were limited by time and computational power. With access to these tools and more time, we would like to run a more thorough grid search on the random forest and boosting ensembles.

Acknowledgments

Data was used from the Memory & Attention Control Laboratory at Arizona State University. We would like to thank Dr. McCulloch for his guidance and assistance in this project.

Appendix: R Script

```r
#Import time series data
setwd("D:/Coding/R_Projects/FinalProjectASUMLF19")
standards = read.csv("StandardTSData.csv")
oddballs = read.csv("OddballTSData.csv")

par(mfrow=c(2,3))

r <- sample(1:180,1)
plot(1:750,oddballs[r,1:750], pch=1,col="red",
     lines(1:750,oddballs[r,1:750], col="red",lwd=2),
     xlab = "Time (ms)",
     ylab = "Amplitude",
     main = "Random Oddball ERP")
r <- sample(1:180,1)
plot(1:750,oddballs[r,1:750], pch=1,col="red",
     lines(1:750,oddballs[r,1:750], col="red",lwd=2),
     xlab = "Time (ms)",
     ylab = "Amplitude",
     main = "Random Oddball ERP")
r <- sample(1:180,1)
plot(1:750,oddballs[r,1:750], pch=1,col="red",
     lines(1:750,oddballs[r,1:750], col="red",lwd=2),
     xlab = "Time (ms)",
     ylab = "Amplitude",
     main = "Random Oddball ERP")

r <- sample(1:720,1)
plot(1:750,standards[r,1:750], pch=1,col="blue",
     lines(1:750,standards[r,1:750], col="blue",lwd=2),
     xlab = "Time (ms)",
     ylab = "Amplitude",
     main = "Random Standard ERP")
r <- sample(1:720,1)
plot(1:750,standards[r,1:750], pch=1,col="blue",
     lines(1:750,standards[r,1:750], col="blue",lwd=2),
     xlab = "Time (ms)",
     ylab = "Amplitude",
     main = "Random Standard ERP")
r <- sample(1:720,1)
plot(1:750,standards[r,1:750], pch=1,col="blue",
     lines(1:750,standards[r,1:750], col="blue",lwd=2),
     xlab = "Time (ms)",
     ylab = "Amplitude",
     main = "Random Standard ERP")
```

```r
library(DMwR)
library(randomForest)
library(caret)
library(plyr)
library(dplyr)
library(pROC)
library(gbm)
library(e1071)
```

```r
setwd("D:/Coding/R_Projects/FinalProjectASUMLF19")

#Import data
allstandards = read.csv("StandardsDF.csv")
alloddballs = read.csv("OddballDFnormalizedrows.csv")
#COMBINE
alldffr <- rbind(alloddballs,allstandards)
alldffr$class = as.factor(alldffr$class)
cat("There are",nrow(allstandards),"standard trials and",nrow(alloddballs),"trials.")
```

```
#For each case in the orginal data set belonging to the minority class, perc.over/100 new examples of that c
lass will be created.
sm.da <- SMOTE(form = class ~ .,
               data = alldffr, perc.over = 300,perc.under = 0)
#How many oddballs are in the new dataframe?
obs <- sm.da$class
sum(sm.da$class == 1)
#Select all the oddballs from new data frame
odind <- which(obs == 1)
smote.odd <- sm.da[odind,]
#Combine all oddballs and original standards
sm.da <- rbind(allstandards,smote.odd)
```

```
write.csv(sm.da,"D:/Coding/R_Projects/FinalProjectASUMLF19/DataFrameSMOTEd.csv", row.names = FALSE)
```

```
setwd("D:/Coding/R_Projects/FinalProjectASUMLF19")
sm.da <- read.csv("DataFrameSMOTEd.csv")
```

```
sm.da$class <- as.factor(sm.da$class)
set.seed(100)
n.samp <- ceiling(nrow(sm.da)*.75)
nn <- sample(1:nrow(sm.da),n.samp,replace = F)
sm.train <- sm.da[nn,]
sm.test <- sm.da[-nn,]
```

```
sm.lfg = randomForest(formula = sm.train$class ~ .,
                      data = sm.train,
                      ntree = 50)
sm.ypred <- predict(sm.lfg, newdata = sm.test, type ='class')
sm.yprob <- predict(sm.lfg, newdata = sm.test, type ='prob')
```

```
xint = which(sm.yprob[,2] >= 0)
plot(x = xint,y = sm.yprob[,2],col = sm.test$class,
     xlab = "Observation",ylab = "Oddball Probability")
abline(h=0.5, col="yellow")
```

```
#Plot confusion matrix
sm.cm <- confusionMatrix(data = sm.ypred,
                         reference = sm.test$class,
                         dnn = c("Predicted","Actual"))
print(sm.cm$table)
```

```
#Plot ROC Curves
par(pty = "s")
roc <- roc(sm.test$class, as.numeric(sm.ypred), plot=TRUE)
print(roc$auc)
```

```
p=ncol(sm.train)-1
mtryv = c(27, 100)
ntreev = c(10,20,30)
setrf = expand.grid(mtryv,ntreev)
colnames(setrf)=c("mtry","ntree")
print(setrf)
```

```
rfr <- matrix(0.0,nrow(setrf),nrow(sm.test)) #Create an empty matrix to store predictions for each iteration
of our grid search
rfp <- matrix(0.0,nrow(setrf),nrow(sm.test))
auc = matrix(0.0,1,nrow(setrf))
###fit random forests
for(k in 1:nrow(setrf)) {
  #cat("on randomForest fit ",k,"\n")
  #print(setrf[k,])
  frf = randomForest(formula = sm.train$class ~ .,
                     data=sm.train,
                     mtry=setrf[k,1],
                     ntree=setrf[k,2])
  frfpred = predict(frf,newdata=sm.test)
  rfr[k,]=frfpred
  frfprob = predict(frf,newdata=sm.test,type = 'prob')
  rfp[k,] = frfprob[,2]
  rf.cm <- confusionMatrix(data = frfpred,
                           reference = sm.test$class,
                           dnn = c("Predicted","Actual"))
  #print(rf.cm$table)
  rfroc <-roc(sm.test$class,rfr[k,])
  auc[,k] = rfroc$auc
}
```

```
for(i in 1:ncol(auc)){
  cat("AUC for ",i,":",auc[,i],"\n")
}
```

```
#Plot ROC curves for grid search
par(pty = "s")
roc(sm.test$class,rfr[1,], plot=TRUE)
for(z in 1:nrow(rfr)){
  print(z)
  zz = roc(sm.test$class,rfr[z,])
  lines(zz,col = z)
}
```

```
ma = which.max(auc)
cat("The best AUC is from model: ",ma,", with a value of ",auc[,ma],"\n")
```

```
#Need to re-run model; can't pull from matrix because you lost the factor characteristics
param = setrf[ma,]
marf = randomForest(formula = sm.train$class ~ .,
                      data = sm.train,
                      ntree = param[1,2],
                    mtry = param[1,1])
marfpred = predict(marf, newdata = sm.test, type ='class')

plot(x = 1:ncol(rfp),y = rfp[ma,],
     xlab = "Observation",ylab = "Oddball Probability",col = sm.test$class)
abline(h=0.5, col="yellow")

orf = confusionMatrix(data = marfpred,
                           reference = sm.test$class,
                           dnn = c("Predicted","Actual"))
print(orf$table)
```

```
varImpPlot(marf)
```

```
#Grid Search variable selection
vi.fr <- varImp(marf)
vi.vec <- as.numeric(unlist(varImp(marf)))
max.ind <- which.max(vi.vec)
max.val <- vi.fr[max.ind,]
thr.val <- floor(max.val)
res <- thr.val/10

maxvarimp <- thr.val
tmat <- as.data.frame(seq(0,floor(max.val),res))

varimppred <- matrix(0.0,nrow(tmat),nrow(sm.test))
marfauc = matrix(0.0,1,nrow(varimppred))
pused = matrix(1:p)
pused = t(pused)
```

```
varimp = as.data.frame(varImp(marf)) #Orders variables by importance
for(z in 1:nrow(tmat)){
  cat("Model",z,", threshold at",tmat[z,],"\n")
  new = matrix(0.0,1,nrow(varimp)) #Matrix to store certain variables
  threshold = tmat[z,] #Set variable importance threshold
  for(p in 1:nrow(varimp)){
    if(varimp[p,]>threshold){ #If variable importance is above threshold
      new[,p] = p
    }
  }
  tm=which(new[1,] > 0)
  varimp = as.data.frame(varimp[tm,])
  if(nrow(varimp)>=param[1,1]){
  pos <- which(new > 0)
  wc <- c(pos,751)
  newtrain <- sm.train[,wc]

  marf2 = randomForest(newtrain$class~.,data=newtrain,
                       ntree=param[1,2],mtry=param[1,1])
  phat = predict(marf2,newdata=sm.test)
  varimppred[z,]=phat
  varimp = varImp(marf2)

  cm <- confusionMatrix(data = phat,
                        reference = sm.test$class,
                        dnn = c("Predicted","Actual"))
  #print(cm$table)
  marfroc <-roc(sm.test$class,varimppred[z,])
  marfauc[,z] = marfroc$auc
  #print(marfauc[,z]) #just to make sure model is actually changing
  tm = as.data.frame(tm)
  tm = t(tm)
  pused = rbind.fill.matrix(pused,tm)
  } else{
    print("not enough variables")
  }
}
```

```
for(i in 1:ncol(marfauc)){
  cat("AUC for ",i,":",marfauc[,i],"\n")
}
```

```
vma = which.max(marfauc)
ps = which(pused[vma,] > 0)
ps = pused[vma,ps]
nps = matrix(1:750,750,1)
nps = nps[-ps,]
cat("The best AUC is from model: ",vma,", with a value of ",marfauc[,vma],"\n")
```

```
#Need to re-run model; can't pull from matrix because you lost the factor characteristics
bv = c(ps,751)
btr = sm.train[,bv]
brf = randomForest(formula = btr$class ~ .,
                     data = btr,
                     ntree = param[1,2],
                    mtry = param[1,1])
brfpred = predict(brf, newdata = sm.test, type ='class')
brfprob = predict(brf, newdata = sm.test, type ='prob')

xint = which(brfprob[,2] >= 0)
plot(x = xint,y = brfprob[,2],col = sm.test$class,
     xlab = "Observation",ylab = "Oddball Probability")
abline(h=0.5, col="yellow")

par(pty = "s")
roc <- roc(sm.test$class, as.numeric(brfpred), plot=TRUE)
rfauc = roc$auc

obrf = confusionMatrix(data = brfpred,
                           reference = sm.test$class,
                           dnn = c("Predicted","Actual"))
print(obrf$table)
```

```
sm.tr.b = sm.train; sm.tr.b$class = as.numeric(sm.tr.b$class)-1
sm.te.b = sm.test; sm.te.b$class = as.numeric(sm.te.b$class)-1

##NEW settings for boosting
#Tree depth; number of boosting iterations; shrinkage

idv = c(1,2); ntv = c(10,20); shv = c(.01,.001)
setboost = expand.grid(idv,ntv,shv)
colnames(setboost) = c("tdepth","ntree","shrink")
print(setboost)

#storage for fits
boost.fits = matrix(0.0,nrow(setboost),nrow(sm.test))
```

```
##fit boosting
for(i in 1:nrow(setboost)) {
  #cat("on boosting fit ",i,"\n")
  print(setboost[i,])
  ##fit and predict
  fboost = gbm(sm.tr.b$class ~ .,data=sm.tr.b,distribution="bernoulli",
               n.trees=setboost[i,2],interaction.depth=setboost[i,1],
               shrinkage=setboost[i,3])
  phat = predict(fboost,newdata=sm.te.b,n.trees=setboost[i,2],interaction.depth=setboost[i,1],
                 shrinkage=setboost[i,3],type="response")
  boost.fits[i,] = phat
}
```

```
#Then run the code below:
for(n in 1:nrow(boost.fits)){
  for(q in 1:ncol(boost.fits)){
    if(boost.fits[n,q] >= .5){
      boost.fits[n,q]= 1} else{
        boost.fits[n,q] = 0}
  }
}
```

```
#Plot ROC curves for grid search
roc(sm.test$class, as.numeric(boost.fits[1,]), plot=TRUE)
auc = matrix(0.0,1,nrow(boost.fits))
for(z in 1:nrow(boost.fits)){
  print(z)
  cmx = confusionMatrix(data = as.factor(boost.fits[z,]),
                        reference = sm.test$class,
                        dnn = c("Predicted","Actual"))
  #print(cmx$table)
  par(pty = "s")
  kk <-roc(sm.te.b$class, as.numeric(boost.fits[z,]))
  lines(kk,col = z)
  auc[,z] = kk$auc
}
```

```
for(i in 1:ncol(auc)){
  cat("AUC for ",i,":",auc[,i],"\n")
}
```

```
vma = which.max(auc)
cat("The best AUC is from model: ",vma,", with a value of ",auc[,vma],"\n")
```

```
par(pty = "s")
roc <- roc(sm.test$class, as.numeric(brfpred), plot=TRUE)
kk <-roc(sm.te.b$class, as.numeric(boost.fits[vma,]))
btauc = kk$auc
lines(kk,col = "yellow")
legend(x = .5,y=.3,legend = c("RF","Boost"),col = c("black","yellow"),lty = c(1))
cat("Random Forest AUC:",rfauc,"\n","Boosting AUC: ",btauc)
```