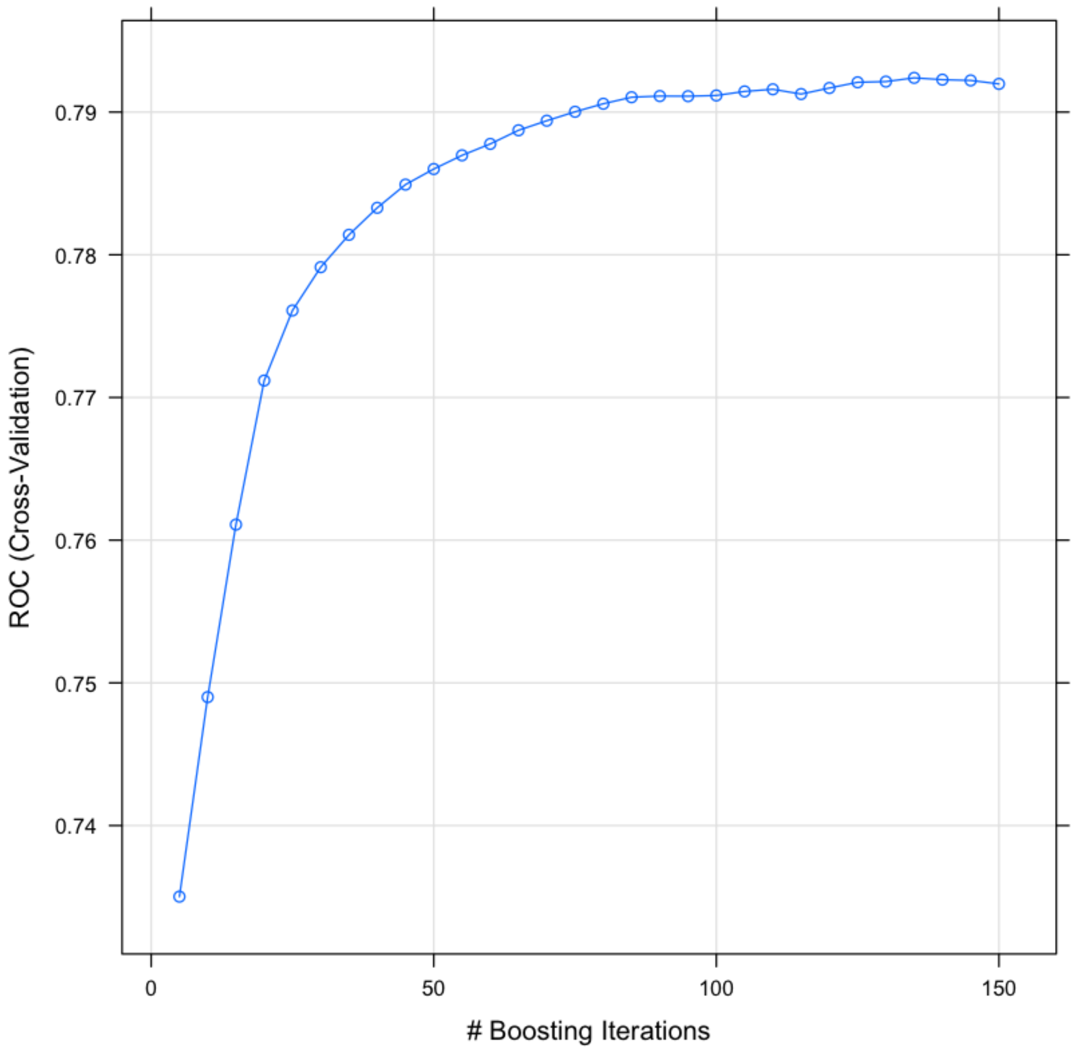
Skyler's Session

June 29, 2016

```
In [2]: #install.packages(c("caret"))
        #install.packages(c("pROC"))
        #install.packages(c("gbm"))
        #install.packages(c("plyr"))

        library(caret)
        library(pROC)
        library(gbm)
        library(plyr)



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

        e <- read.csv("dsss_5.csv")

        #The following lines turn the numeric (0/1) label into a FACTOR
        DFT<- as.factor(e[,1])
        e[,1]<- NULL
        levels(DFT) <- list(good = "0", dft = "1")
        e<-data.frame(e,DFT)

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


        grid <- expand.grid(interaction.depth = c(1),        #what depth of trees should we grow?
        n.trees = seq(5, 150, by=5),     #how many of each depth?
        shrinkage = c(.2),               #smaller steps gives better accuracy but requires more trees
        n.minobsinnode = 10)             #helps prevent trees from making leaves with few observations



        cvCtrl <- trainControl(method = "cv",                      #cross validation
        number =5,                        #how many folds?
        summaryFunction = twoClassSummary,  #allows us to calculate ROC metrics
        classProbs = TRUE,                #keep probabilities
        verboseIter= TRUE)                #watch updating of folds
```

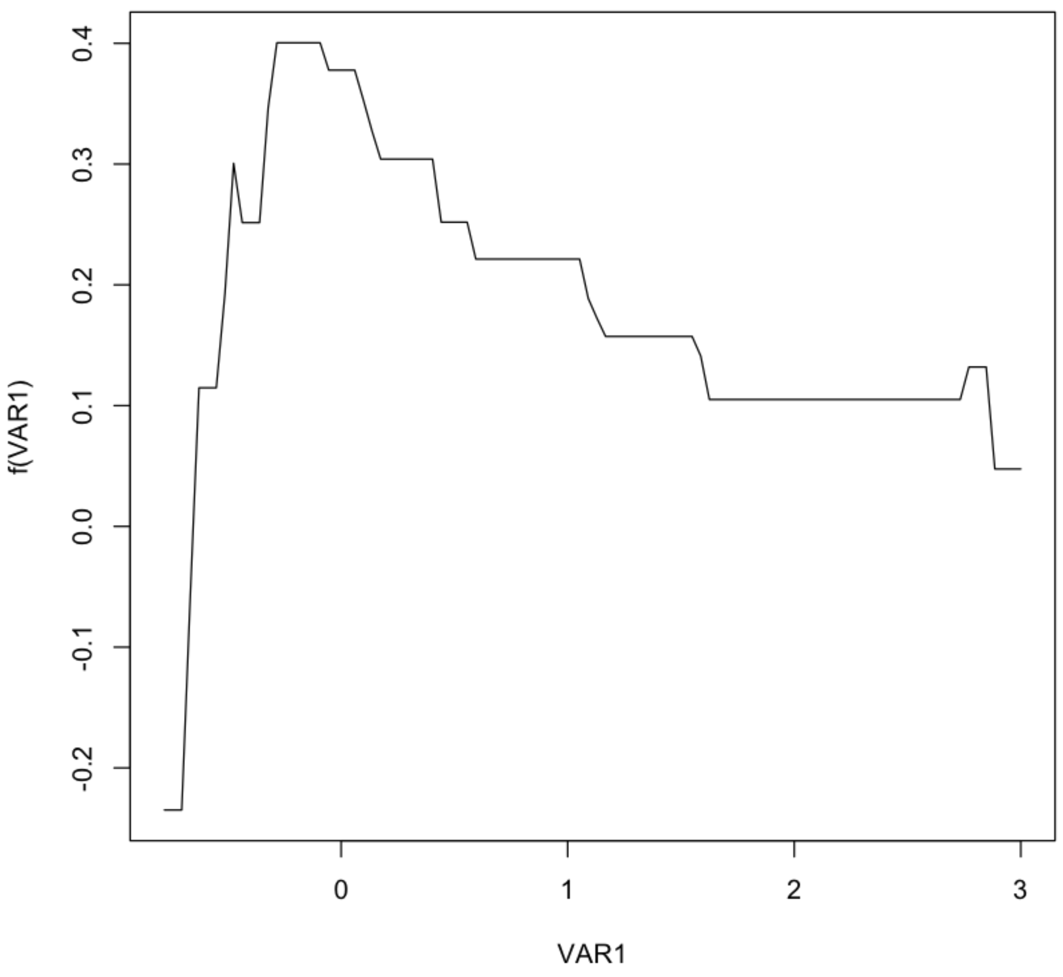
```
gbm_test <- train(DFT ~ .,                  #DFT is the label '.' is short for everything
#Can also try DFT ~ VAR1 + VAR2 to only use 2 predictors
data = e,                  #Our data
method = "gbm",            #use gradient boosted trees, caret has dozens others
distribution = "adaboost",#there are multiple loss functions in GBM, also try "bernoulli"
bag.fraction = .5,         #combine boosting and bagging
metric = "ROC",            #Choose the best model by maximizing ROC (enabled by twoClassSummary)
trControl = cvCtrl,        #perform cross validation according to above values
tuneGrid = grid,           #explore depth and tree numbers according to above values
verbose = FALSE,           #be quiet
na.action = na.pass)       #caret can remove na values, but gbm accepts them.



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

plot(gbm_test)

plot(gbm_test$finalModel, i.var = 1)


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

pretty.gbm.tree(gbm_test$finalModel, i.tree = 1)


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

results <- predict(gbm_test,              #runs the data set 'e' through the gbm_test model
newdata = e,
type = "prob",          #give probabilities instead of assigning a predicted label
na.action = na.pass)    #gbm model can handle na values




b=50                                      #number of bins for our histograms
g<- results[ e$DFT =="good", ]$good       #store probability for all the paying customers
d<- results[ e$DFT =="dft" , ]$good       #store probability for all the defaulting customers

hg <- hist(g, breaks =b, plot = F)        #make 2 histograms of good and bad, don't plot
dg <- hist(d, breaks =b, plot = F)

top <- max(hg$counts, dg$counts)          #makes for a prettier graph, stores the highest bin

hist( g,
col=rgb(0,0,1,0.5),                 #goods are blue-ish with 0.5 transparency
```

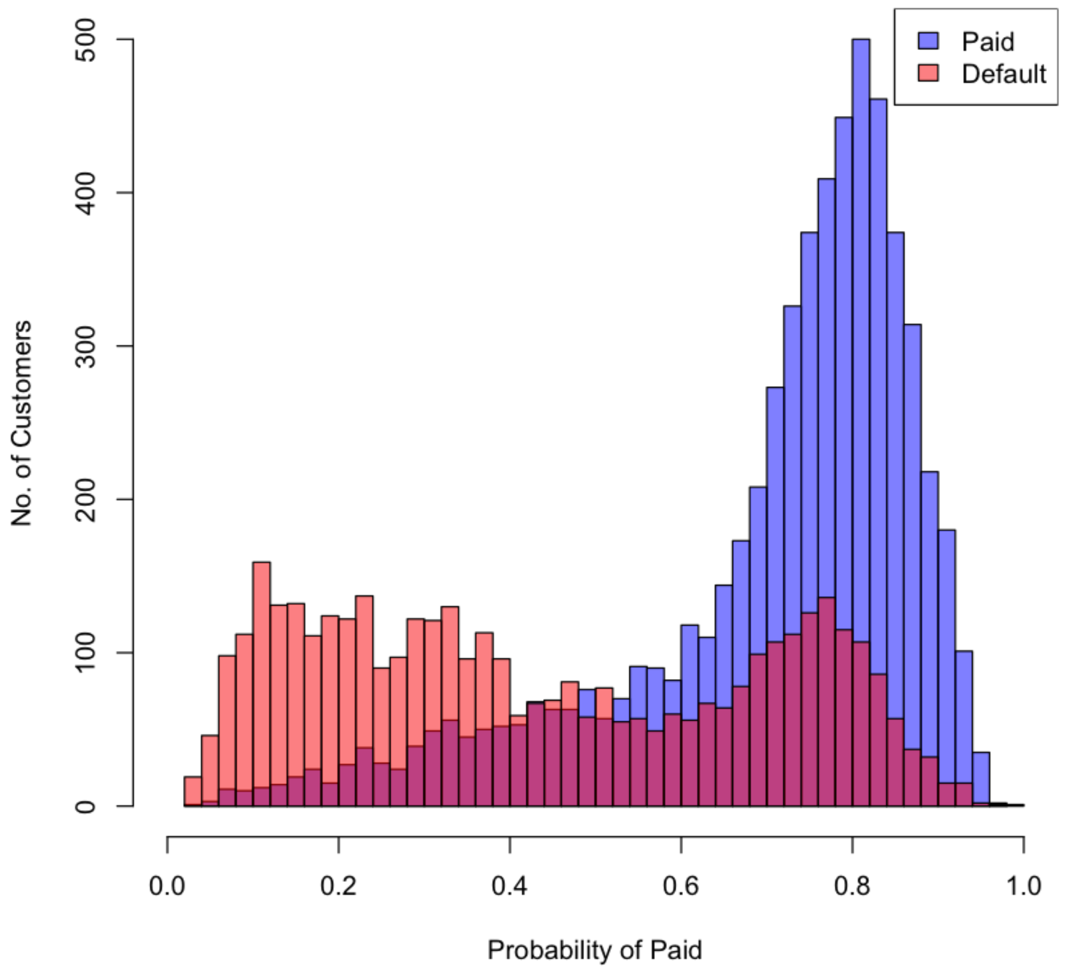
```
        breaks=b,
        xlim = c(0, 1), ylim=c(0,top),      #top of histogram is based on highest of either good or bad
        xlab ="Probability of Paid", main = "",  ylab = "No. of Customers")

        legend("topright", c("Paid","Default"), fill=c( rgb(0,0,1,0.5), rgb(1,0,0,0.5))  )

        hist( d,
        col=rgb(1,0,0,0.5),                 #defaulters are redish with 0.5 transparency
        breaks=b,
        add=T)                              #add this to the previous histogram instead of making a new


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

+ Fold1: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
- Fold1: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
+ Fold2: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
- Fold2: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
+ Fold3: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
- Fold3: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
+ Fold4: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
- Fold4: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
+ Fold5: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
- Fold5: shrinkage=0.2, interaction.depth=1, n.minobsinnode=10, n.trees=150
Aggregating results
Selecting tuning parameters
Fitting n.trees = 135, interaction.depth = 1, shrinkage = 0.2, n.minobsinnode = 10 on full training set
```



Out[2]:

	SplitVar	SplitCodePred	LeftNode	RightNode	MissingNode	ErrorReduction	Weight	Prediction
0	4	-0.195	1	2	3	803.7961	5000	0.004228327
1	-1	-0.1003761	-1	-1	-1	0	1794	-0.1003761
2	-1	0.06276243	-1	-1	-1	0	3206	0.06276243
3	-1	0.004228327	-1	-1	-1	0	5000	0.004228327


f(VAR1)
VAR1
-0.2
-0.1
0.0
0.1
0.2
0.3
0.4
0
1
2
3



Paid
Default
No. of Customers
Probability of Paid
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]: