Final Report

Chris Obermeier

04/21/2020

## Section 1 - Executive Summary

This report analyzes and evaluates historical loan data in order to see if this can be used to significantly predict whether or not someone is going to have a loan that is in a good or bad status, and which loans will generate the most profit. Methods of analysis included finding outliers and useful variables with graphs and preparing and cleansing the dataset for outliers and blanks. The dataset was explored with tables and boxplots and then used to create a single variable to enter into the dataset indicating whether a loan status was “good” or “bad.” Training and testing models were generated using the variables found to have the highest prediction percentages based on correlation values. Finally, confusion matrix and profit analysis was done by evaluating different classification thresholds for maximum profit generation. All calculations can be found attached in the report. The result of the analysis and creation of the models show that the model could be used to generate more profit (than even a “perfect” model that denied all loans that had a “bad” loan status) by $1,451,587.

The report shows that the loans are not currently being accepted and denied in a way that maximizes its loan profits, additional money could be made if certain changes are made. The recommended changes are: • Use historical data to focus on the profitability instead of whether a loan is “Good” or “Bad” • Use the created model that best predicts profitability. • Use thresholds to fine tune how many loans are accepted to maximize profitability.

This report was generated using the best data that it was given, but there were limitations of the analysis. Some of the limitations are: • Outside factors that may effect this data like large-scale economic are unknown. • There is no limitation as to how many loans or total amount of loans the bank can accept or any other outside reasons the loans cannot be accepted. • The granularity of the data is limited to how it was collected and cannot be further drilled into. • The data entered is assumed to be correct, and outliers and assumed to be true outliers.

## Section 2 - Introductions

As a bank, predicting which customers are more likely to maintain a good loan status, and who will likely have a loan in a bad status is extremely important. This information can be extremely helpful when deciding whether or not to give out that loan, and what terms are tied to it.

In this particular part of the project I will analyze a group of data points that may or may not be indicators of whether or not a potential loan customer will keep their loan in a good status or put it into a bad status, based on historical data that will be used to create a model that can predict future customers loan statuses.

I will begin by preparing and cleaning the data by breaking up the data into two sections for analysis. Data that is numerical in nature, and data that is factorial, or group based, and compare each set separately using the appropriate statistical testing.

Based on those results I will select a few key indicators that have been identified as either equaling the same real mean (numeric) or are proven to be dependent (factor) on the good and bad loan status.

From there I will analyze and choose to transform the data to help it create an even stronge indicator by removing skewness and tranform the data into having less extreme values. That data will then be used to finalize the data that will be used in order to predict a good or bad loan status as well as possible with the data given.

## Section 3 - Preparing and Cleaning the data

Your response variable is a new version of the status variable and will be a factor variable that has two levels: “Good” and “Bad”.

Loans that are late, current (being paid), or in grace period should be removed from the data.

library(data.table)  
  
df <- read.csv("loans50k.csv")  
  
df <- df[!(df$status %like% "Late" | df$status=="In Grace Period" | df$status=="Current"),]

Good loans are all those that are fully paid. Bad loans are loans that have a status of charged off or default (there may not be any “default” in this data).

library(data.table)  
  
#Load Data  
df <- read.csv("loans50k.csv")  
  
#Remove Bad States  
df <- df[!(df$status %like% "Late" | df$status=="In Grace Period" | df$status=="Current" | df$status=="Default"),]  
  
#Assign "Good" and "Bad" by if the loan is fully paid or not, minus the other bad states that removed rows  
df$statusGoodBad <- ifelse(df$status == "Fully Paid", "Good", "Bad")  
df$statusGoodBad <- factor(df$statusGoodBad)

Eliminate variables that you think are clearly not useful as predictors and explain your choices.

#good or bad column of data  
  
#removing data that should not logically affect good or bad credit standing (just loanID)  
df <- subset(df, select = -c(loanID))

From a logical perspective there was only one I felt comfortable removing using basic logic which was the loanID. There are some other columns like state that although may not SEEM important, without testing it’s hard to say if some states have higher rates than others.

Separting numerics from factors for analysis

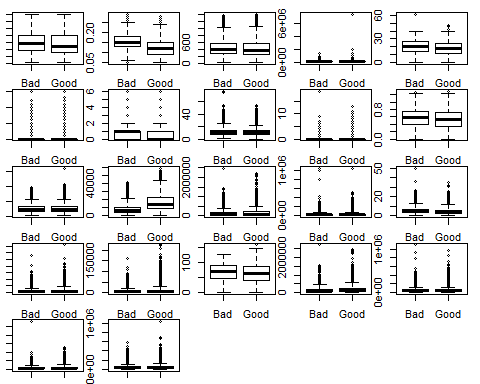
#Tried to run, but it won't since some of the other variables I want to compare it to are factors.  
  
#creating subset of factor columns  
term <- df$term  
grade <- df$grade  
employment <- df$employment  
length <- df$length  
home <- df$home  
verified <- df$home  
status <- df$status  
reason <- df$reason  
state <- df$state  
statusGoodBad <- df$statusGoodBad  
  
#Saving vector of columns that are factors  
factor\_df <- data.frame(term, grade, employment, length, home, verified, status, reason, state, statusGoodBad)  
  
#removing factor type columns from df  
df = subset(df, select = -c(term, grade, employment, length, home, verified, status, reason, state))

Creating a plot for the numerics so I can see how they look

attach(df)

## The following object is masked \_by\_ .GlobalEnv:  
##   
## statusGoodBad

par(mar = c(1,1,1,1), mfrow=c(5,5))  
l <- lapply(1:22, function(x) plot(df[,x] ~ df[,23]))



I’m looking to see if the values are significantly different for the y axis for the two variables bad and good to see if $statusGoodBad is dependent on any of the numerical columns. Lots of these plots have large amounts of outliers in their box plots which likely means this data is not going to be very useful unless they are transformed.

Correlation analysis:

#Pearson correlation tests  
with(df, cor.test(as.numeric(statusGoodBad), amount))$estimate

## cor   
## -0.05256144

with(df, cor.test(as.numeric(statusGoodBad), rate))$estimate

## cor   
## -0.2741917

with(df, cor.test(as.numeric(statusGoodBad), payment))$estimate

## cor   
## -0.02805173

with(df, cor.test(as.numeric(statusGoodBad), income))$estimate

## cor   
## 0.0520191

with(df, cor.test(as.numeric(statusGoodBad), debtIncRat))$estimate

## cor   
## -0.1258594

with(df, cor.test(as.numeric(statusGoodBad), delinq2yr))$estimate

## cor   
## -0.01705693

with(df, cor.test(as.numeric(statusGoodBad), inq6mth))$estimate

## cor   
## -0.06921251

with(df, cor.test(as.numeric(statusGoodBad), openAcc))$estimate

## cor   
## -0.03450904

with(df, cor.test(as.numeric(statusGoodBad), pubRec))$estimate

## cor   
## -0.01926367

with(df, cor.test(as.numeric(statusGoodBad), revolRatio))$estimate

## cor   
## -0.05969023

with(df, cor.test(as.numeric(statusGoodBad), totalAcc))$estimate

## cor   
## 0.007127911

with(df, cor.test(as.numeric(statusGoodBad), totalPaid))$estimate

## cor   
## 0.3547454

with(df, cor.test(as.numeric(statusGoodBad), totalBal))$estimate

## cor   
## 0.0715052

with(df, cor.test(as.numeric(statusGoodBad), totalRevLim))$estimate

## cor   
## 0.05705426

with(df, cor.test(as.numeric(statusGoodBad), accOpen24))$estimate

## cor   
## -0.1262231

with(df, cor.test(as.numeric(statusGoodBad), avgBal))$estimate

## cor   
## 0.0819526

with(df, cor.test(as.numeric(statusGoodBad), bcOpen))$estimate

## cor   
## 0.08674829

with(df, cor.test(as.numeric(statusGoodBad), bcRatio))$estimate

## cor   
## -0.06432804

with(df, cor.test(as.numeric(statusGoodBad), totalLim))$estimate

## cor   
## 0.07988231

with(df, cor.test(as.numeric(statusGoodBad), totalBcLim))$estimate

## cor   
## 0.07568071

with(df, cor.test(as.numeric(statusGoodBad), accOpen24))$estimate

## cor   
## -0.1262231

with(df, cor.test(as.numeric(statusGoodBad), totalIlLim))$estimate

## cor   
## 0.001750597

with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(term)))$estimate

## cor   
## -0.1994995

with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(grade)))$estimate

## cor   
## -0.2784533

with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(employment)))$estimate

## cor   
## 0.01445512

with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(length)))$estimate

## cor   
## -0.0322163

with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(home)))$estimate

## cor   
## -0.06742023

with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(verified)))$estimate

## cor   
## -0.06742023

#Not needed since this is the status value that was used to great $statusGoodBad  
#with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(status)))$estimate  
with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(reason)))$estimate

## cor   
## -0.0310365

with(factor\_df, cor.test(as.numeric(statusGoodBad), as.numeric(state)))$estimate

## cor   
## -0.007140792

The highest correlation value (closest to 1) to show statusGoodBad was correlated to any of the numeric values was $totalPaid - cor = 0.35

*Note: This is not very high at all, so I would not use this value personally- but since it’s the highest value we have, I will go forward with this*

#Checking p-value of the $totalPaid value  
with(df, cor.test(as.numeric(statusGoodBad), amount))$p.value

## [1] 1.232019e-22

Kept: $totalPaid

creating new data frame with good numeric indicator (and statusGoodBad in column 1):

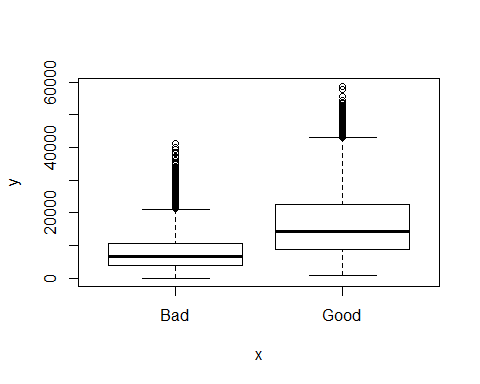
#Creating data frame with kept variables  
df\_kept <- data.frame(df$statusGoodBad, df$totalPaid)

#Missing values: I did not find missing values in any of the columns that I kept- I did see zeros, but I am unable to say that those are missing for sure. Additionally I probably fixed some of the missing data when I assigned the good or bad values by creating an if-else statement and removing data that was not one way or the other.

Variables kept as indicators: $statusGoodBad (indicator), $totalpaid df\_kept

## Section 4 - Exploring and Transforming the data

#plotting the value I kept as a model indicator  
plot(df\_kept$df.statusGoodBad, df\_kept$df.totalPaid)

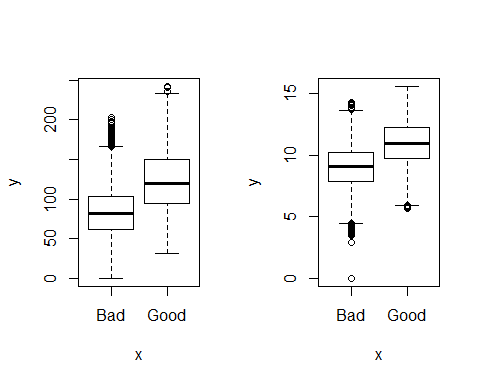
 I can see that there are a lot of outliers in the data that are larger y values, so I’m going to see if taking the square root can help the data fit into the model better.

attach(df)

## The following object is masked \_by\_ .GlobalEnv:  
##   
## statusGoodBad

## The following objects are masked from df (pos = 3):  
##   
## accOpen24, amount, avgBal, bcOpen, bcRatio, debtIncRat, delinq2yr,  
## income, inq6mth, openAcc, payment, pubRec, rate, revolRatio,  
## statusGoodBad, totalAcc, totalBal, totalBcLim, totalIlLim,  
## totalLim, totalPaid, totalRevBal, totalRevLim

par(mfrow=c(1,2))  
  
#Take Square root of totalPAid to lessen outliers  
df\_kept$sqrtTotalPaid <- sqrt(df\_kept$df.totalPaid)  
plot(df\_kept$df.statusGoodBad, df\_kept$sqrtTotalPaid)  
  
#still high amounts of outliers so trying to take another square root to see results  
df\_kept$sqrtTotalPaid2 <- sqrt(df\_kept$sqrtTotalPaid)  
plot(df\_kept$df.statusGoodBad, df\_kept$sqrtTotalPaid2)

 After the initial square root of total pay the outliers are lessened by a good amount. After the second round of taking a square root it looks as if it’s about equal to the amount so I will re-take the Pearson product-moment correlation tests.

cor.test(as.numeric(df\_kept$df.statusGoodBad), df\_kept$sqrtTotalPaid)

##   
## Pearson's product-moment correlation  
##   
## data: as.numeric(df\_kept$df.statusGoodBad) and df\_kept$sqrtTotalPaid  
## t = 79.793, df = 34651, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.3850515 0.4028409  
## sample estimates:  
## cor   
## 0.3939831

cor.test(as.numeric(df\_kept$df.statusGoodBad), df\_kept$sqrtTotalPaid2)

##   
## Pearson's product-moment correlation  
##   
## data: as.numeric(df\_kept$df.statusGoodBad) and df\_kept$sqrtTotalPaid2  
## t = 83.623, df = 34651, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.4009809 0.4185029  
## sample estimates:  
## cor   
## 0.4097797

Since the second one has a higher correlation value I will stick with the second sqrt value taken $sqrtTotalPaid2 in order to reflect a better model with a higher correlation value.

## Section 5 - The Logistic Model

This section will be used to create two different sets of data, one for training and one for testing. This datasets will be used to create create models, and ultimately those models will be used to make predictions.

#Assign training and testing datasets

#Set Seed  
set.seed(0)  
  
#Assign training index at 80% of data frame  
train\_index <- sample(1:nrow(df), 0.8 \* nrow(df))  
  
#Assign testing index at 80% of data frame  
test\_index <- setdiff(1:nrow(df), train\_index)  
  
#Create 80% training dataset  
df\_train <- df[train\_index, -15]  
df\_train <- na.omit(df\_train)  
  
#Create 20% testing index  
df\_test <- df[test\_index, -15]  
df\_test <- na.omit(df\_test)

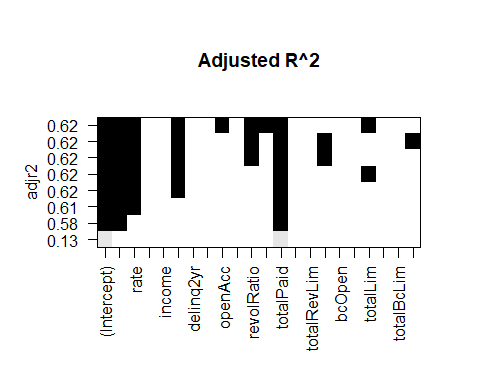
# Creating training model

This code uses the regsubsetes method to gather the summaries, takes the best model’s summary, plots it, finds the best 8 variables, and creates a training model with it.

library(leaps)

## Warning: package 'leaps' was built under R version 3.6.3

#create best single model  
regsubsets.out <- regsubsets(statusGoodBad~., data=df\_train, nvmax=8)  
  
#summary of best single model  
summary.out <- summary(regsubsets.out)  
  
#plot best single model  
plot(regsubsets.out, scale = "adjr2", main = "Adjusted R^2")



#Find best 8 variables  
summary.out$which[8,]

## (Intercept) amount rate payment income debtIncRat   
## TRUE TRUE TRUE FALSE FALSE TRUE   
## delinq2yr inq6mth openAcc pubRec revolRatio totalAcc   
## FALSE FALSE TRUE FALSE TRUE TRUE   
## totalPaid totalBal totalRevLim avgBal bcOpen bcRatio   
## TRUE FALSE FALSE FALSE FALSE FALSE   
## totalLim totalRevBal totalBcLim totalIlLim   
## TRUE FALSE FALSE FALSE

#create training model (without totalPaid)  
training\_model <- glm(statusGoodBad ~ totalIlLim \* amount \* rate \* openAcc \* revolRatio \* debtIncRat \* totalAcc, data = df\_train, family="binomial")

#Generate Predictions and analyze performance (accuracy) of model

#pass model created in with df\_test as the newData  
preds = predict(training\_model, df\_test, type = "response")  
pred.y = ifelse(preds > .5, 1, 0)  
y = df\_test$statusGoodBad  
  
#Graph confusion matrix  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 137 1316  
## Good 137 5272

#Calculate Percent Accuracy  
correctPredictions5 <- (137+5272)/(137+5272+137+1316)  
correctPredictions5

## [1] 0.7882542

Based on the percentages of the confusion matrix being accurate 78% of the time, this would be a decent way to predict if a loan will be repaid or not.

## Section 6 - “Optimizing the Threshold for Accuracy”

#Threshold at .6

pred.y = ifelse(preds > .6, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 277 1176  
## Good 355 5054

#Calculate Percent Accuracy  
correctPredictions6 <- (277+5054)/(277+5054+355+1176)  
correctPredictions6

## [1] 0.7768872

.6 as a threshold shows that it’s about 1% lower than it was at the 50% threshold.

#Threshold at .4

pred.y = ifelse(preds > .4, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 56 1397  
## Good 47 5362

#Calculate Percent Accuracy  
correctPredictions4 <- (56+5362)/(56+5362+47+1397)  
correctPredictions4

## [1] 0.7895657

.4 as a threshold shows a higher percentage (79%) than a .5 threshold but only by .1 %

#Threshold at .3

pred.y = ifelse(preds > .3, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 19 1434  
## Good 11 5398

#Calculate Percent Accuracy  
correctPredictions3 <- (19+5398)/(19+5398+11+1434)  
correctPredictions3

## [1] 0.78942

.3 as a threshold has a lower percent than .4 but only by .01%

#Doing all others above .6 to round out to 1.0 in .1 segments for the graph, and .2 and lower to 0 to round out the lower ones as well

#Trying at .7

pred.y = ifelse(preds > .7, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 581 872  
## Good 945 4464

#Calculate Percent Accuracy  
correctPredictions7 <- (581+4464)/(581+4464+945+872)  
correctPredictions7

## [1] 0.7352084

#Trying at .8

pred.y = ifelse(preds > .8, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 1013 440  
## Good 2245 3164

#Calculate Percent Accuracy  
correctPredictions8 <- (1013+3164)/(1013+3164+2245+440)  
correctPredictions8

## [1] 0.6087147

Better prediciton in both types!

#Trying at .9

pred.y = ifelse(preds > .9, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 1372 81  
## Good 4400 1009

#Calculate Percent Accuracy  
correctPredictions9 <- (1372+1009)/(1372+1009+4400+81)  
correctPredictions9

## [1] 0.3469834

Better prediction in both types!

#Trying at 1

pred.y = ifelse(preds > 1, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0  
## Bad 1453  
## Good 5409

#Calculate Percent Accuracy  
correctPredictions10 <- (1453+0)/(1453+0+5409+0)  
correctPredictions10

## [1] 0.2117458

#Threshold at .2

pred.y = ifelse(preds > .2, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 5 1448  
## Good 4 5405

#Calculate Percent Accuracy  
correctPredictions2 <- (5+5405)/(5+5405+4+1448)  
correctPredictions2

## [1] 0.7883999

#Threshold at .1

pred.y = ifelse(preds > .1, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 1 1452  
## Good 2 5407

#Calculate Percent Accuracy  
correctPredictions1 <- (1+5407)/(1+5407+2+1452)  
correctPredictions1

## [1] 0.7881084

#Threshold at 0

pred.y = ifelse(preds > 0, 1, 0)  
table(y, pred.y)

## pred.y  
## y 1  
## Bad 1453  
## Good 5409

#Calculate Percent Accuracy  
correctPredictions0 <- (1453)/(1453+5409)  
correctPredictions0

## [1] 0.2117458

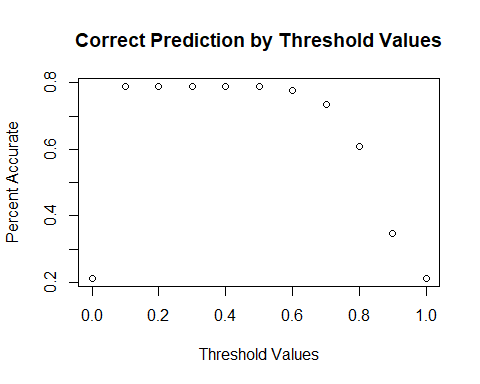
#What value of the threshold produces maximum overall accuracy and what is that accuracy? Explore the tradeoff between correctly predicting good and bad loans.

The threshold at .4 is the hightest value of accuracy at 79%. Going lower doesn’t add much in terms of less prediction percentage until 0, but going up has a much more drastic value drop, especially at .7 and higher.

The tradeoff here is that the more you change the prediction threshold you are changing the results instead of improving the model, so it is limited and effects negative positives and negative negatives more or less based on going lower or higher in the threshold.

#Graphing thresholds

index <- c(0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)  
  
percentCorrect <- c(correctPredictions0, correctPredictions1, correctPredictions2, correctPredictions3, correctPredictions4, correctPredictions5, correctPredictions6, correctPredictions7, correctPredictions8, correctPredictions9, correctPredictions10)  
  
df2 <- data.frame(index, percentCorrect)  
  
plot(df2, main="Correct Prediction by Threshold Values", xlab="Threshold Values", ylab = "Percent Accurate")

 ## Section 7 - “Optimizing the Threshold for Profit”

#Applying your model using the test data

#create model from test data  
test\_model <- glm(statusGoodBad ~ totalIlLim \* amount \* rate \* openAcc \* revolRatio \* debtIncRat \* totalAcc, data = df\_test, family="binomial")

#Compute the profit by assuming the bank denies all of the loans that your model predicts as “bad”.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .4  
df\_test$predictionGoodOrBad = ifelse(preds > .4, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit4 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit4 <- sum(totalProfit4)  
  
totalProfit4

## [1] 2421031

#How does this change the total profit?

The highest prediction percentage was using the .4 threshold which had 79% accuracy, and the total profit was only 17% as much if we use no prediction.

#Now investigate how changing the classification threshold affects the total profit if loans that are predicted as bad are denied by the bank. #0 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .4  
df\_test$predictionGoodOrBad = ifelse(preds > 0, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit0 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit0 <- sum(totalProfit0)  
  
totalProfit0

## [1] 2087807

#0.1 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .4  
df\_test$predictionGoodOrBad = ifelse(preds > 0.1, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit1 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit1 <- sum(totalProfit1)  
  
totalProfit1

## [1] 2068655

#0.2 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .4  
df\_test$predictionGoodOrBad = ifelse(preds > 0.2, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit2 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit2 <- sum(totalProfit2)  
  
totalProfit2

## [1] 2127394

#0.3 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .4  
df\_test$predictionGoodOrBad = ifelse(preds > 0.3, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit3 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit3 <- sum(totalProfit3)  
  
totalProfit3

## [1] 2196291

#0.5 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .4  
df\_test$predictionGoodOrBad = ifelse(preds > 0.5, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit5 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit5 <- sum(totalProfit5)  
  
totalProfit5

## [1] 2792503

#0.6 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .6  
df\_test$predictionGoodOrBad = ifelse(preds > 0.6, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit6 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit6 <- sum(totalProfit6)  
  
totalProfit6

## [1] 3400942

#0.7 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .7  
df\_test$predictionGoodOrBad = ifelse(preds > 0.7, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit7 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit7 <- sum(totalProfit7)  
  
totalProfit7

## [1] 3539394

#0.8 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .8  
df\_test$predictionGoodOrBad = ifelse(preds > 0.8, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit8 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit8 <- sum(totalProfit8)  
  
totalProfit8

## [1] 2785960

#0.9 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .9  
df\_test$predictionGoodOrBad = ifelse(preds > 0.9, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit9 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid-df\_test$amount,0)  
#sum totalSaved  
totalProfit9 <- sum(totalProfit9)  
  
totalProfit9

## [1] 763288

#1.0 threshold.

#Create variable with prediction "Good" or "Bad" strings if the threshold is greater than .9  
df\_test$predictionGoodOrBad = ifelse(preds > 1.0, "Good", "Bad")  
  
#If the prediciton is bad, and the actual status is bad, we saved money with the prediction  
totalProfit10 = ifelse(df\_test$predictionGoodOrBad == "Good", df\_test$totalPaid - df\_test$amount,0)  
#sum totalSaved  
totalProfit10 <- sum(totalProfit10)  
  
totalProfit10

## [1] 0

#Compared to not using your model, what is the maximum percentage increase in profit that can be expected by deploying your model?

#sum all all paid  
totalPaid <- sum(df\_test$totalPaid)  
#sum of all given  
totalAmountOfLoans <- sum(df\_test$amount)  
#total without model  
profitNoModel <- totalPaid - totalAmountOfLoans  
  
#What percent of extra money can be saved with the model  
(totalProfit7 - profitNoModel) / (profitNoModel)\*100

## [1] 69.52686

The maximum increase of 16.0% when using the highest threshold of 0.4.

#How does this increase in profit compare to the increase in profit from a perfect model that denies all of the truly bad loans?

totalProfit7-profitNoModel

## [1] 1451587

The profit compared to a perfect model that denies all truly bad loans would be $1,451,587.

#For your best profit threshold, what is the overall accuracy and percentages of correctly predicted good and bad loans?

As shown above the percentage for 0.7 threshold where it was the most profitable, the model was 73.5% accurate.

#Does the maximum profit threshold coincide with the maximum accuracy threshold?

No.The highest prediction percentage was using the .4 threshold which had a accuracy of 79%, but the 0.7 threshold at 73.5% percentage is lower and it has a highest profitability.

## Section 8 - “Results Summary”

#Details for the final classification model for bank

The classification model for the bank we will use is: training\_model <- glm(statusGoodBad ~ totalIlLim \* amount \* rate \* openAcc \* revolRatio \* debtIncRat \* totalAcc, data = df\_train, family=“binomial”).

#Final value of the classification threshold

The value of the classification model with the highest level of profit for the bank will be at the .7 threshold.

#Overall profit at 0.7

totalProfit7

## [1] 3539394

#Accuracy of the model - Breakdown of the percentages of correctly predicted good and bad loans.

pred.y = ifelse(preds > .7, 1, 0)  
table(y, pred.y)

## pred.y  
## y 0 1  
## Bad 581 872  
## Good 945 4464

#Calcualte Good Loan Percent Accuracy  
4464/(4464+945)

## [1] 0.8252912

#Calculate Bad Loan Percent Accuracy  
581 / (581+872)

## [1] 0.3998624

#Calculate Overall Percent Accuracy  
correctPredictions7 <- (581+4464)/(581+4464+945+872)  
correctPredictions7

## [1] 0.7352084

The model predicts good loans very well at 82.5%, but not so well bad loans at 40.0%. The overall percentage however, is still fairly high at a 73.5% accuracy.

The reason for this is that there are much more good loans than bad loans, so the higher percentage of good loans is a larger amount of data that is being used as an overall predicter.

This difference in good/bad amounts is also likely why we taking a lower accuracy allows us to generate a higher amount of profit than a higher level of accuracy, since more accuracy may not mean that we are accepting some of the additional big loans that are more profitable and worth taking in some that aren’t.

It also was more profitable in order to use a model that was made in order to increase profit rather than to deny all “bad” loans altogether.