Predicting Loan Defaults

Dave Weller

November 28, 2018

## Executive Summary

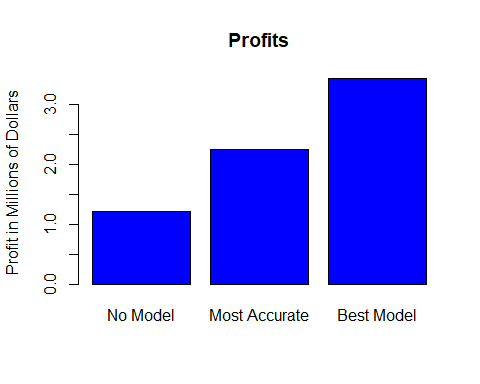
We have been asked to come up with a model that will predict whether or not a loan will be paid in full and not defaulted on or charged off. We were given data on 50,000 loans that were issued by the bank.

The first thing we did was prepare the data, keeping just the loans that were paid off, defaulted on, or charged off. We removed varialbes that would have no effect on the outcome of the loan and variables that were redundant.

We used logistic regression to develop a model that could be used to predict whether a loan would be good or bad. This model was accurate in 78% of the cases it was tested on, including getting 98% of the good loans right. However, it only caught about 10% of the bad loans.

Predicting based on a model is a balancing act. The way to change the balance of our prediction is to modify the classification threshold. Basically, the classification threshold is used to determine whether a loan will be good or bad. By changing the threshold, we can change the accuracy of the prediction.

It turns out that our original threshold of 0.5 was the most accurate, but by changing the threshold to 0.7, we were able to increase the accuracy of predicting bad loans up to 48%, though the overall accuracy fell to 73%. The profits, however, reached their maximum with this threshold.



## Introduction

The purpose of this exercise is to determine if there are variables that can predict whether or not a loan will be defaulted on.

### Data

The dataset contains information about 50,000 loans. There are 30 variables that describe these loans.

## Setup

### Load Packages

library(dplyr)  
library(ggformula)  
library(mice)  
library(gridExtra)  
library(HH)  
library(leaps)

### Load Data

loans <- read.csv("loans50k.csv")

## Preparing and Cleaning the Data

The response variable, status, currently has 8 values. The only loans we are interested in are ones that have a status of “Charged Off”, “Default”, or “Fully Paid”. We will remove the rows that do not have one of those values for status.

loans <- loans[(loans$status=="Charged Off" | loans$status=="Default" | loans$status=="Fully Paid"),]

We now have 34,655 records. Now our response variable has 3 values. We will add a new variable that will hold 2 values, “Good” and “Bad”, that we will use as the response variable.

responseVariable <- ifelse(loans$status == "Fully Paid", "Good","Bad")  
loans$response <- factor(responseVariable)

There are some variables that will not be good predictors, and they will be removed.

The Loan ID is just a unique identifier given after the loan is issued, and thus will not have any predictive value. Total Paid is another variable that is not known until after the loan is issued. We will keep Total Paid for now, but we will remove it from the training set after we split the data into training and test sets. This way, we can figure out the total profit for of the predicted results of the test set.

Employment has far too many job titles for any of them to be useful. There are over 21,000 factors in the data set of 34,655 loans.

Some of the variables are redundant. There are several ratios that are based on variables in the data set. These will also be removed.

Finally, status is what the response variable is based on, so it is of no use to us any more.

loans <- subset(loans, select = -c(loanID, status, revolRatio, bcRatio, debtIncRat, employment))

There are 14 factor levels for reasons, and several account for less than 1% each. These will be combined into the “other” factor.

p <- .01  
lf <- names(which(prop.table(table(loans$reason))<p))  
levels(loans$reason)[levels(loans$reason) %in% lf] <- "other"

The dataset now contains 26 variables describing 34,655 loans.

## Exploratory Data Analysis

### Missing Values

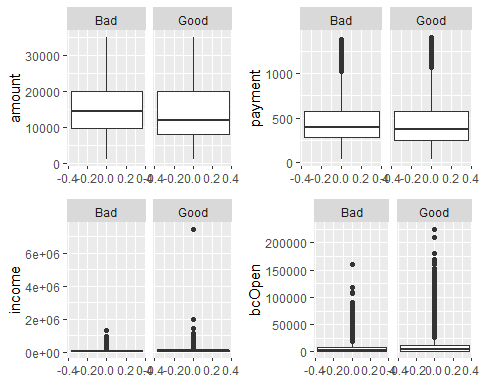
md.pattern(loans, plot = FALSE)

## amount term rate payment grade length home income verified reason state  
## 34295 1 1 1 1 1 1 1 1 1 1 1  
## 360 1 1 1 1 1 1 1 1 1 1 1  
## 0 0 0 0 0 0 0 0 0 0 0  
## delinq2yr inq6mth openAcc pubRec totalAcc totalPaid totalBal totalRevLim  
## 34295 1 1 1 1 1 1 1 1  
## 360 1 1 1 1 1 1 1 1  
## 0 0 0 0 0 0 0 0  
## accOpen24 avgBal totalLim totalRevBal totalBcLim totalIlLim response  
## 34295 1 1 1 1 1 1 1  
## 360 1 1 1 1 1 1 1  
## 0 0 0 0 0 0 0  
## bcOpen   
## 34295 1 0  
## 360 0 1  
## 360 360

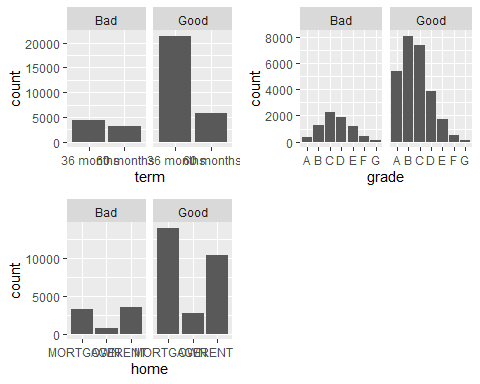
The only variable with missing values is bcOpen. Since less than 5% of the values are missing, it will be safe to impute them.

### Relationships Between Response Variable and Other Variables

p1 <- gf\_boxplot(~income, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
p2 <- gf\_boxplot(~amount, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
p3 <- gf\_boxplot(~payment, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
p4 <- gf\_boxplot(~bcOpen, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
  
grid.arrange(p2, p3, p1, p4, nrow=2)



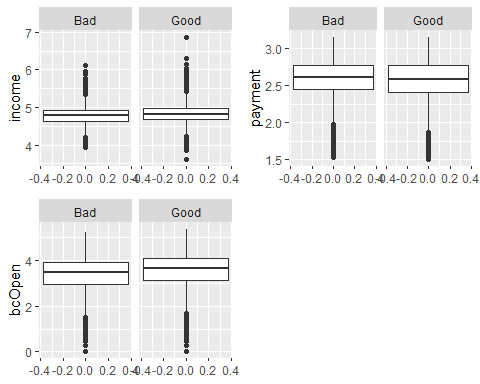
p1 <- gf\_bar(~ term, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
p2 <- gf\_bar(~ grade, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
p3 <- gf\_bar(~ home, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
  
grid.arrange(p1, p2, p3, nrow=2)



### Adjust for Skewness

Income, Payment, and bcOpen are all heavily skewed to the right. In order to minimize this effect, we will replace income with the logarithm of the income. We will add 1 to the values in order to avoid missing values if we take the log of 0.

loans$income <- log10(loans$income + 1)  
loans$payment <- log10(loans$payment + 1)  
loans$bcOpen <- log10(loans$bcOpen + 1)  
  
p1 <- gf\_boxplot(~income, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
p2 <- gf\_boxplot(~payment, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
p3 <- gf\_boxplot(~bcOpen, data = loans) %>%  
 gf\_facet\_grid(. ~ response)  
  
grid.arrange(p1, p2, p3, nrow=2)



## The Logistic Model

We will look for collinearity by using the VIF function.

vif(loans)

## amount term rate payment grade length   
## 14.957151 NA 14.839405 6.735776 NA NA   
## home income verified reason state delinq2yr   
## NA 2.013435 NA NA NA 1.086439   
## inq6mth openAcc pubRec totalAcc totalPaid totalBal   
## 1.158235 2.929416 1.064225 2.249589 8.817502 93.524075   
## totalRevLim accOpen24 avgBal bcOpen totalLim totalRevBal   
## 6.955069 1.650714 5.627529 1.636800 95.657548 15.069034   
## totalBcLim totalIlLim response   
## 3.457774 12.246789 NA

We removed totalLim and totalBal from the dataset because of a high collinearity.

loans <- subset(loans, select = -c(totalLim, totalBal))

We then split the data set into 2 data sets, one to train the model and one to test the model. The training set includes a random sample of 80% of the original data set, with the test set made up of the remaining 20%.

sampleSize <- floor(0.8\*nrow(loans)) # Training set will be 80%, the test set is the remaining 20%  
  
set.seed(2015)  
trainIndex <- sample(seq\_len(nrow(loans)), size = sampleSize)  
  
train <- loans[trainIndex, ]  
test <- loans[-trainIndex, ]  
  
train <- subset(train, select = -totalPaid) #remove totalPaid from the training set

model <- glm(response ~ ., data = train, family = "binomial")

## Optimize the Model for Accuracy

We examined the predictions with a classification threshold of 0.5.

prediction <- predict(model, newdata = test, type = "response")  
  
getPredictionResults <- function(prediction, test, threshold){  
  
 confusionMatrix <- addmargins(table(prediction<threshold, test$response))  
 confusionMatrix  
   
 truPos <- confusionMatrix[1,2]  
 falsePos <- confusionMatrix[1,1]  
 truNeg <- confusionMatrix[2,1]  
 falseNeg <- confusionMatrix[2,2]  
 total <- confusionMatrix[3,3]  
   
 accuracy <- (truPos + truNeg)/total  
 goodPct <- truPos/(truPos + falseNeg)  
 badPct <- truNeg/(truNeg + falsePos)  
   
   
 result <- (c(accuracy, goodPct, badPct))  
 names(result) <- c('Acc', 'Good', 'Bad')  
   
 return(result)  
  
}  
  
preds <- getPredictionResults(prediction, test, .5)  
  
paste('Accuracy: ', preds['Acc'])

## [1] "Accuracy: 0.788630789207907"

paste('Percentage of Good Loans predicted as good:', preds['Good'])

## [1] "Percentage of Good Loans predicted as good: 0.979448250324014"

paste('Percentage of Bad Loans predicted as bad:', preds['Bad'])

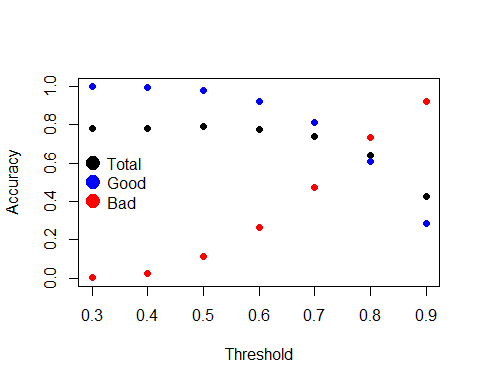
## [1] "Percentage of Bad Loans predicted as bad: 0.115032679738562"

The overall accuracy of the model is 78%, which is pretty good. The model was extremely good at predicting good loans, getting that right almost 98% of the time. It was not a good model for predicting bad loans, though, only catching about 10% of them. Considering that the bank would like to predict loans that will be defaulted on, this may not be the best model.

We varied the threshold from 0.3 and 0.9. Anything below 0.3 did not produce any Bad Loan predictions.

thresholds <- (3:9)/10  
acc <- c()  
good <- c()  
bad <- c()  
  
for (x in thresholds){  
 newPreds <- getPredictionResults(prediction,test, x)  
 acc <- c(acc, newPreds['Acc'])  
 good <- c(good, newPreds['Good'])  
 bad <- c(bad, newPreds['Bad'])  
}

plot(thresholds, acc, ylim = c(0,1),  
 xlab = 'Threshold',  
 ylab = 'Accuracy',  
 pch=19)  
points(thresholds, good, pch=19, col='blue')  
points(thresholds, bad, pch=19, col='red')  
legend("left", legend = c('Total', 'Good', 'Bad'),  
 col = c('black', 'blue', 'red'), pch = c(19.19,19),   
 bty = "n", pt.cex = 2, cex = 1,   
 text.col = "black", horiz = F , inset = c(0.0, 0))

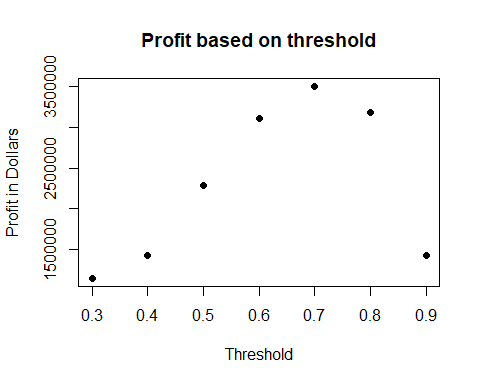


Overall accuracy is best (78.4%) when the threshold is set at 0.5. However, the accuracy for predicting the bad loans is only 10.4% at that point. When the threshold is set at 0.7, the overall accuracy drops to 73.6%, but the accuracy for finding bad loans jumps to 48%. If we are willing to settle for an overall accuracy of 62.8%, the accuracy for finding bad loans is 73.2% with a threshold of 0.8.

## Optimizing the Model for Maximum Profit

We then optimized the model to get maximum profit. We did this by looking at totalPaid - amount for each loan that would be approved. We did this for thresholds between 0.3 and 0.9.

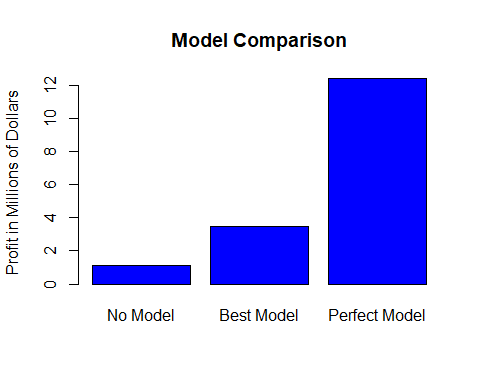
perfectModel <- subset(test, response == 'Good')  
profitPerfect <- sum(perfectModel$totalPaid - perfectModel$amount)  
profitsNoModel <- sum(test$totalPaid - test$amount)  
profits <- c()  
for (x in thresholds){  
 i <- prediction>x  
 profits <- c(profits, sum(test[i,]$totalPaid-test[i,]$amount))  
}  
  
maxProfit <- max(profits)  
percentIncrease <- (maxProfit - profitsNoModel)/profitsNoModel  
percentPerfectVsMax <- (profitPerfect - maxProfit)/maxProfit  
  
plot(thresholds, profits, pch=19,  
 xlab = 'Threshold',  
 ylab = 'Profit in Dollars',  
 main = 'Profit based on threshold')



The profits can be maximized by placing the threshold at 0.7. This would have resulted in a profit of $3,423,096, a 184% increase over the profit of $1,202,024 that would have been made by approving all of the loans. This still comes up short of a perfect model, which would have seen a profit of $12,405,138, or a 262% increase over the best model.

Overall, the model that showed the largest profit has an accuracy of 73.6%. This model correctly categorized 80.8% of the good loans and 48.1% of the bad loans. This is not the model that had the best accuracy.

barplot(c(profitsNoModel, maxProfit, profitPerfect)/1000000,  
 names.arg = c('No Model', 'Best Model', 'Perfect Model'),  
 ylab = 'Profit in Millions of Dollars',  
 main = 'Model Comparison',  
 col='blue')



## Summary

A model can be made that will predict which loans will either be defaulted on or charged off. The model that we selected was based on the largest profit generated campared to not using any model. This model saw a 184% increase in profit.

This model was best used with a classification threshold of 0.7. This increased profits by correctly predicting a larger percentage of bad loans without affecting overall accuracy too much.

The overall accuracy of the best model is 73.6%, with 80.8% of the good loans correcty predicted and 48.1% of the bad loans correctly predicted.