Final Project DS 705

Joe Athas

**Executive Summary**

Our current method of loan approval lacks vetting and as a consequence many loans end up defaulting.  This has been a costly process, however it has given us a large amount of data to analyze and find a better way.  Analyzing this data we have gathered using statistics allows us to find a loan selection process that optimizes profits and reduces the chances of letting potentially bad loans get selected in the first place.

The first step after being handed the sufficient financial data is cleaning it and preparing it for analysis. Then, we determine which variables are actually associated with whether or not a loan is likely to default.  Using logistic regression, 30 variables such as “loan amount in dollars”, “number of credit checks in the past 6 months”, “total credit limits” and other common loan metrics were analyzed. We looked for common correlations between these variables and loan default rates.  Using a general linear equation, only nine of these variables were deemed statistically significant and therefore useful in predicting loan status. These variables are as follows:

Grade of loan

Term of loan

Debt to income ratio of applicant

Applicant’s total balance of all credit accounts

How many accounts applicant opened in the past 24 months

Total unused credit on credit cards of applicant

Amount of loan

Whether applicant owned/rented/has mortgage

Total number of credit lines in file for applicant (both open and closed)

Using these nine variables and a mathematical model that predicts loan status allows us to test our data to see the actual rate of successful loan prediction.  To do this, we first had to partition the dataset into two separate subsets, one for analysis and model building, one for testing and model verification. After the nine variables were extracted from the first set and an optimized model for loan prediction was found, the second set was tested against the model and found that it could accurately predict whether or not a loan would fail with a 78% maximum accuracy.  This is much more secure technique compared to our current method of simply accepting every loan. We are able to modify the mathematical model’s parameters to either maximize accuracy, or maximize profits. Interestingly enough, the most accurate parameter isn’t necessarily the most profitable. This is due to the fact that some loans on the fringe of being considered “good” by the model end up being partially or even having the majority paid off.

By simply accepting every loan, our subset of data indicated net profits to equal approximately $1.45 million.  Modifying the model’s parameters as suggested above, we can maximize profits to be around $3.5 million. We can more than double profits by utilizing this method of vetting during loan application processes.

Defaulting loans are expensive to our company.  By utilizing the logistic regression model appended below we can optimize for profits and accuracy during the loan application process.  A full technical summary will be appended in this report.

# Introduction

It is very important banks only give loans to those that are able to pay them back in full. Loans without a lien can be a potential risk for a bank. Loans that have a lien or collateral are also very costly in the event of a default. The goal for this data exploration and analysis is to determine a model with appropriate predictor variables using data often gathered by financial institutions and to use that model to predict the likelihood of a loan applicant defaulting or paying off the loan. To do this some manipulation of the data may be in order. I will be using R to do preliminary exploration, cleaning, and transformation of the provided sample data, “loans50k.csv”.

citation()

##   
## To cite R in publications use:  
##   
## R Core Team (2018). R: A language and environment for  
## statistical computing. R Foundation for Statistical Computing,  
## Vienna, Austria. URL https://www.R-project.org/.  
##   
## A BibTeX entry for LaTeX users is  
##   
## @Manual{,  
## title = {R: A Language and Environment for Statistical Computing},  
## author = {{R Core Team}},  
## organization = {R Foundation for Statistical Computing},  
## address = {Vienna, Austria},  
## year = {2018},  
## url = {https://www.R-project.org/},  
## }  
##   
## We have invested a lot of time and effort in creating R, please  
## cite it when using it for data analysis. See also  
## 'citation("pkgname")' for citing R packages.

# Section 3: Preparing and Cleaning the data

Upon receiving the data, the first thing I did was make the appropriate response variable: A binary variable (as is needed for logistic regression) produced from the original variable “status”. Fully paid loans were reclassified as good, those in “default” or “charged off” were reclassified as bad. All other levels of status (loans in progress or not complete) were removed from the data set.

#Make new variable: Set "Good" and "Bad" Loans, all else "blank" to be removed later  
df <-loans50k %>%  
 mutate(payment\_status = case\_when(status == "Fully Paid" ~ "Good", status == "Charged Off" ~ "Bad",  
 status == "Default" ~ "Bad",  
 TRUE ~ ""))  
#Probably best to get rid of the null values we aren't interested in payment\_status:  
df<-df %>% drop\_na(payment\_status)  
  
#Turning any fields into NA values makes it easier to eliminate later  
df[df==""]<-NA

I also removed “employed” immediately as there were too many levels and empty values. Last thing I did before really digging into the data was removing all the rows containing NA values. All the remaining NAs were in I noticed it only reduced n by approximately 1% which didn’t compromise the integrity of the original set all too much. A few other variables were removed for redundancy.

#Test where all NAs are to   
which(is.na(df), arr.ind=TRUE)  
#Found a bunch in "employment". Going to eliminate on account of too many levels/NA values.   
df$employment<- NULL  
#Test where the NAs are again:  
which(is.na(df), arr.ind=TRUE)  
#last NA values are in bcOpen and bcRatio, of which there are roughtly 400 incomplete entries,   
#or roughly 1% of the dataset. Eliminating these wont be a huge issue.   
#Finally omit all the rows containing NA values. (This will not eliminate the NA values in "length" as they are strings)  
df1<-na.omit(df)

Here a covariance matrix was constructed of the numerical predictors to aid in this process of determining redundancy. I chose an arbitrary value of .8 of Pearson’s r to classify variables as strongly correlated. The strong correlations were as follows: Total limit of credit cards was heavily correlated with Total unused credit on cards. Unused credit (bcOpen) seemed like a more logical choice than totalBcLim (total limit of cc) and it eliminated some redundancy. ‘Payment’ was heavily associated with ‘amount’ as monthly payment is a function of the total amount of the loan. I deleted this variable to eliminate redundancy. Proportion of revolving credit use(revolRatio) was eliminated due to redundancy with ratio of total credit card limits (bcOpen). Average balance and total credit limit were both strongly correlated with total Balance. I figured there is a lot of redundancy between these three variables, keeping just totalBal would suffice.

#make subset of numerical data check covariance matrix on numeric factors.   
dfNum<-select\_if(df1, is.numeric)  
m<-cor(dfNum)  
#easy way to find strong correlations; arbitrary value set at .8  
strongCorrelations = m > .8  
#Found few redundant variables...will eliminate

A few predictor variables were initially removed due to lack of relevance. These included “totalPaid”, “state”, and “loanID”. “State” and “loanID” have nothing really to do with the actual loan, whereas “totalpaid” cannot be a predictor variable of a completed loan. A few more were eliminated based on redundancy as mentioned above.

#Final check for number of levels in each categorical variable  
dfChar<-select\_if(df1,is.character)   
sapply((sapply(dfChar,unique)),length)

## term grade length home verified   
## 2 7 12 3 3   
## status reason state payment\_status   
## 3 13 49 2

cor(df$avgBal,df$totalLim)

## [1] NA

#Eliminating: totalPaid, employment (already done), payment,revolRatio, avgBal, totalLim,state,status,loanID  
df\_New = as.data.frame(subset(df1, select = -c(payment,revolRatio, avgBal, totalLim,state,status,loanID,totalIlLim,totalBcLim)))

A few other variables I touched up were “verified” and “reason”. “Verified” had three levels, two of which were redundant; “source verified” and “verified”. I changed all “source verified” to just “verified” to eliminate the mistake. “Reason” contained a few levels with very few instances which is considered less powerful for logistic regression. I modified “wedding” and “renewable\_energy” to “other” and “house” to “major\_purchase” to consolidate a few levels. After going about data transformation in a later step, a few more concerning variables were mutated. ‘pubRec’ and ‘delinq2yr’ (number of derogatory public records and number of 30+ day late payments in last two years respectively) both contained many instances of 0 due to their nature. Since many loans have ‘0’ represented in these respective variables, choosing to set it as categorical made more sense. I chose three levels for both variables (the same since they were similar scales) “None”, “1-3” and “More than 3”. Instructions

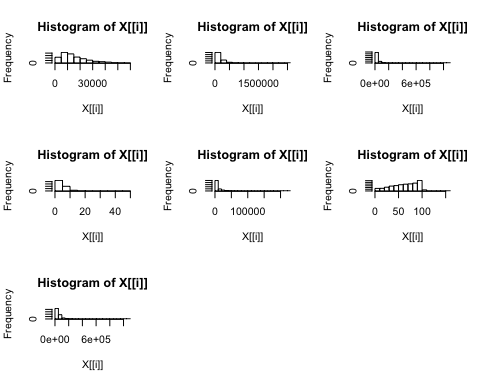
#consolidate "verified" variable into two levels instead of three  
df\_New$verified<-revalue(df\_New$verified, c("Source Verified"="Verified"))  
unique(df1$verified)  
  
#going to combine a few levels in "reason" due to the lack of instances  
df\_New$reason<-revalue(df\_New$reason,c("wedding"="other","renewable\_energy"="other","house"="major\_purchase"))  
#make df of only continuous variables  
dfCont<-select\_if(df\_New, is.character)  
#Test for instances of each level  
x=sapply(dfCont,count)  
x  
#lowest #of instances is now above 200

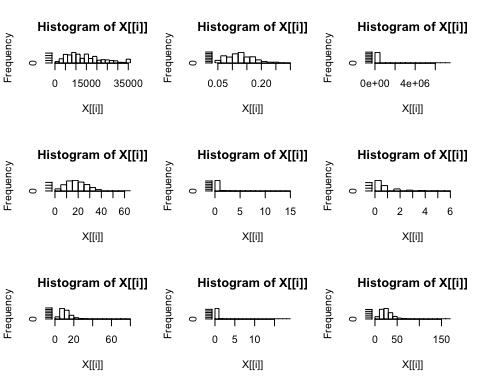
### After removing redundencies and a few predictor variables, initial cleaning of dataset brings n = 34271.

# Section 4 - “Exploring and Transforming the data”

Certain data transformations were necessary in order to meet the assumption of normality for logistic regression. I first analyzed the skewness of each distribution. Considering a skewness greater than the absolute value of 1 is considered “highly skewed”, I created a r function to attempt various transformations, specifically a log, natural log, square root, cube root, 4th root, and reciprocal, then return the skewness of the transformed distribution. I applied this function to every variable with an initial skewness greater than the absolute value of 1 and determined which transformation technique was the most successful at reducing skewness. Obviously this is a blanket technique, and each variable still needs to be looked at individually. After determining which transformation to use for each variable, I performed the transformations as follows income: log transformation openAcc: log + 1 transformation totalBal: cubed root transformation totalRevLim: cubed root transformation accOpen24: square root transformation bcOpen: cube root transformation totalRevBal: cube root transformation

#make new df of only numeric variables.  
dfNum<-select\_if(df\_New, is.numeric)  
#check out skewness of all numeric variables  
par(mfrow=c(3,3))  
sapply(dfNum, hist)



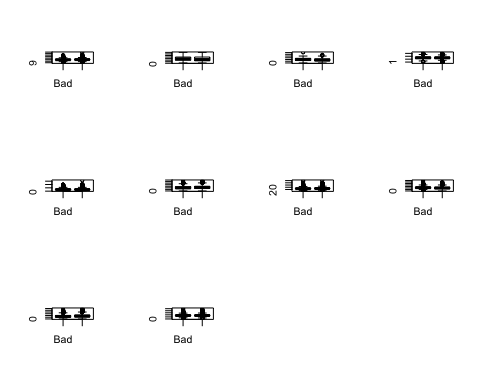


sapply(dfNum,skewness)  
#Distributions that have a skewness greater than 1 are generally considered "highly skewed"  
#will transform: income, delinq2yr,inq6mth,openAcc,pubRec,totalBal,totalRevLim,accOpen24,bcOpen,totalRevBal  
  
#function to success of various transformations  
tansformAndGetSkew<-function(dist1){  
 log=skew(log(dist1))  
 lognat=skew(log(dist1,base=exp(1)))  
 sqrt=skew(sqrt(dist1))  
 curt=skew(sign(dist1) \* abs(dist1)^(1/3) )  
 fthrt=skew(sign(dist1)^(1/4))  
 reciprical=skew(1/dist1)  
   
 list1<- c(log,lognat,sqrt,curt,fthrt,reciprical)  
   
 return(list1)  
}  
#apply function to all numerical variables  
sapply(dfNum, tansformAndGetSkew)  
  
#Use results to infer which might be the best type of transformation  
#Changing pubRec & delinqyr to categorical variables due to too many instances of '0' in both.   
  
#use unique() to determine breaks  
unique(df\_New$pubRec)  
  
#Make three levels, one containing zero, 1-3, and more than 3  
df\_New$pubRec<-cut(df\_New$pubRec,breaks=c(-.10,.1,3,14))  
levels(df\_New$pubRec)<-c("None","1-3","More than 3")  
  
unique(df\_New$pubRec)  
#DO THIS AGAIN But with delinqyr  
df\_New$delinq2yr<-cut(df\_New$delinq2yr,breaks=c(-.10,.1,3,12))  
levels(df\_New$delinq2yr)<-c("None","1-3","More than 3")

#begin transformations  
df\_New$income<-log(df\_New$income)  
df\_New$inq6mth<-(df\_New$inq6mth)^(1/4)  
df\_New$openAcc<-log(df\_New$openAcc+1) #add 1 to oppenAcc to avoid taking log(0)  
df\_New$totalBal<-(df\_New$totalBal)^(1/3)  
df\_New$totalRevLim<-(df\_New$totalRevLim)^(1/3)  
df\_New$accOpen24<-sqrt(df\_New$accOpen24)  
df\_New$bcOpen<-(df\_New$bcOpen)^(1/3)  
df\_New$totalRevBal<-df\_New$totalRevBal^(1/3)  
  
#Check if any NA values induced  
which(is.na(df\_New), arr.ind=TRUE)  
df\_New<-na.omit(df\_New)  
#make new data frame to test skew on numeric data to see if successful.  
dfNum1<-select\_if(df\_New, is.numeric)

Plots and skew are looking more normal which passes the assumptions of logistic regression.

# Exploration:

Side-by-side box plots for continuous data and frequency tables for categorical data were created to compare each factor to the dependent variable. Few variables popped out immediately as good indicators, however, few were found. The variable “term” has a much higher ratio of bad loans to good in the loans that are 60 months vs 36 months. Loans rated ‘D’ or lower also have a much higher ratio of being “bad” than their counterparts. 

## Splitting the data into test and trainig data.

## 80% of the sample size  
smp\_size <- floor(0.8 \* nrow(df\_New))  
## set the seed to make partition reproducible  
set.seed(123)  
train\_ind <- sample(seq\_len(nrow(df\_New)), size = smp\_size)  
  
train <- df\_New[train\_ind, ]  
test <- df\_New[-train\_ind, ]  
  
#eliminated totalPaid as variable  
train$totalPaid<-NULL  
  
#Coerce "payment\_status" response variable into factor to perform logistic regression.   
train$payment\_status <- as.factor(train$payment\_status)

# Begin Logistic Regression Exploration

Using forward step-wise selection in R, a few more predictors were eliminated based on AIC.

full\_model2<-glm(payment\_status~.,data=train,family="binomial")  
null\_model2 <- glm(payment\_status~1,data=train,family="binomial")  
#step(null\_model2,scope=list(lower=null\_model2,upper=full\_model2),direction="forward")

The new model chosen via forward step-wise selection (AIC as selection criteria) is represented by ‘newModel’ below.

newModel<-glm(formula = payment\_status ~ grade + term + debtIncRat + totalBal + accOpen24 + bcOpen + reason + amount + length + home + totalAcc + delinq2yr + inq6mth + income + rate + openAcc + totalRevLim + verified, family = "binomial", data = train)  
summary(newModel)

After examining a full model there are many statistically significant predictors. Eliminating a few would likely be beneficial to reduce the potential for multicollinearity in addition to impose some simplicity. Using the regsubsets function from the ‘Leaps’ package allows comparing many potential models using the full model. Using regsubsets is typically used for linear models only, so some interpretation is needed here.

# Use regsubsets to find best model based on

all\_models <- regsubsets(payment\_status~.,nvmax=12,data=train)  
summary(all\_models)  
summary(all\_models)$adjr2

After cross examining a few of the models created by the regsubsets function in r and the ‘best’ model based on step-wise selection, a few more predictors were eliminated. The ones deemed worth of keeping for now are represented by the formula below.

modelA<-glm(formula = payment\_status ~ grade + term + debtIncRat + totalBal +   
 accOpen24 + bcOpen + amount + home + totalAcc + rate,data = train,family = "binomial")  
summary(modelA)

##   
## Call:  
## glm(formula = payment\_status ~ grade + term + debtIncRat + totalBal +   
## accOpen24 + bcOpen + amount + home + totalAcc + rate, family = "binomial",   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.717 0.304 0.517 0.728 1.603   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.20e+00 1.47e-01 21.82 < 2e-16 \*\*\*  
## gradeB -3.77e-01 8.31e-02 -4.54 5.7e-06 \*\*\*  
## gradeC -7.10e-01 1.09e-01 -6.51 7.6e-11 \*\*\*  
## gradeD -8.85e-01 1.50e-01 -5.91 3.3e-09 \*\*\*  
## gradeE -9.04e-01 1.85e-01 -4.89 1.0e-06 \*\*\*  
## gradeF -9.70e-01 2.46e-01 -3.95 7.8e-05 \*\*\*  
## gradeG -1.10e+00 3.10e-01 -3.56 0.00038 \*\*\*  
## term60 months -6.14e-01 3.97e-02 -15.48 < 2e-16 \*\*\*  
## debtIncRat -2.40e-02 1.92e-03 -12.53 < 2e-16 \*\*\*  
## totalBal 1.04e-02 1.28e-03 8.10 5.4e-16 \*\*\*  
## accOpen24 -3.39e-01 2.45e-02 -13.85 < 2e-16 \*\*\*  
## bcOpen 1.38e-02 2.19e-03 6.30 2.9e-10 \*\*\*  
## amount -1.01e-05 2.21e-06 -4.57 4.8e-06 \*\*\*  
## homeOWN -1.25e-01 5.62e-02 -2.23 0.02603 \*   
## homeRENT -1.80e-01 4.10e-02 -4.40 1.1e-05 \*\*\*  
## totalAcc 6.43e-03 1.55e-03 4.15 3.4e-05 \*\*\*  
## rate -3.96e+00 1.35e+00 -2.93 0.00341 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 28750 on 27409 degrees of freedom  
## Residual deviance: 25781 on 27393 degrees of freedom  
## AIC: 25815  
##   
## Number of Fisher Scoring iterations: 5

After the model was fit, the testing data was input to the model to predict probabilities of loan status. Below is a contingency table and the given proportions of correctly predicted loans both good and bad.

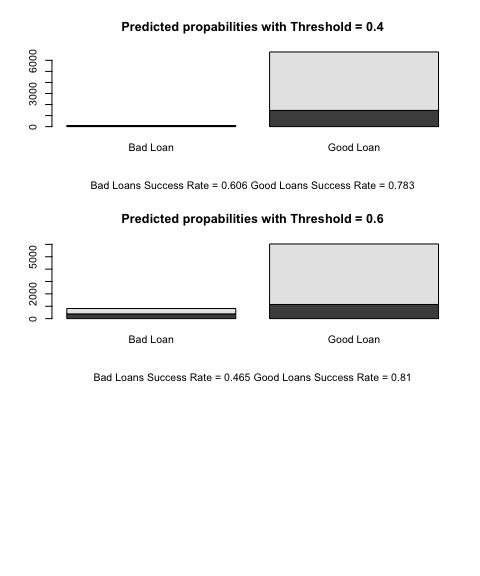
## Bad Loan Good Loan Sum  
## Bad 660 865 1525  
## Good 900 4428 5328  
## Sum 1560 5293 6853

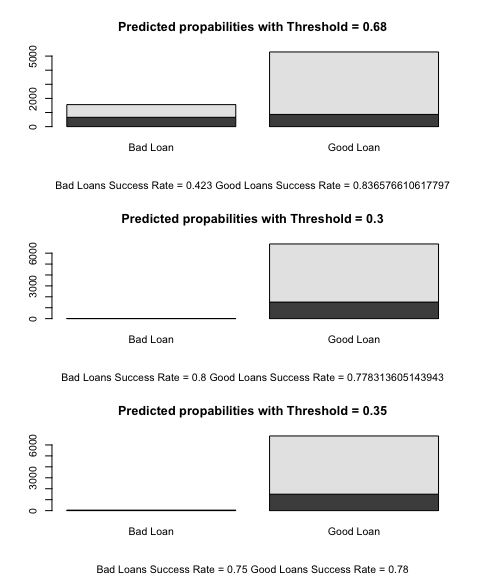
## [1] "Proportion correctly predicted = 0.742448562673282"

## [1] "Proportion of correctly predicted Bad Loans = 0.423076923076923"

## [1] "Proportion of correctly predicted Good Loans = 0.836576610617797"

Although our “Good” loans have a pretty high success rate at 83.6%, the “bad” ones are a little less accurate coming in at a success rate of 42.3%. Here is where a little intuition helps; a bank doesn’t care about predicting good loans as much as it does bad ones. This model is more helpful than chance alone, but the risks are still pretty high due to the inability to properly predict more than 43% of the bad loans. In other words, approximately 17% of the loans predicted as good will actually be bad, whereas 42.3% of the loans predicted as bad could have actually been good. At this point I wouldn’t use this model to actually predict loans.

After playing around with the threshold of the predict() function, I saved a few bar plots of the contingency tables of various thresholds. Below are a few of the graphs for comparison. 



It appears the lower the threshold, the more loans are predicted as ‘Good’. Overall accuracy is pretty consistent from a threshold of .3 to .6. Accuracy maximizes around .52.

#Join predprobs with test data  
test1<-cbind(test, predprob)

Here is a function to produce profits given a threshold for prediction probability. Using this function and modifying the threshold in the parameter allows comparison of profit return. I systematically input thresholds from .1 all the way to .9 to see which threshold maximized profits. After some trial and error, a threshold around .68 maximized profits around 3.5 million dollars.

get\_profits<-function(threshhold,test1){  
 new\_data <- subset(test1, predprob > threshhold)  
 profits<-new\_data$totalPaid-new\_data$amount  
 x<-sum(profits)  
 return(x)  
}  
get\_profits(.3,test1)

## [1] 1526474

get\_profits(.68,test1)

## [1] 3494849

get\_profits(.6803 ,test1)

## [1] 3497823

get\_profits(.6807,test1)

## [1] 3481459

After some trial and error, the threshold that returns the max profit is approximately .6803. The profits for the test data set with a threshold of .6803 are around $3,497,823.

## [1] "If we could predict every loan considered 'Good' with 100% accuracy, profits are $12,365,881"

## [1] "If every loan was accepted, good or bad, profits are only $1,457,687"

## [1] 0.1179

The hypothetical perfect model would be by far the most profitable weighing in at $12,365,881. Using no model, and simply accepting every loan equates to $1,457,687 in profit which equates to 11.79% of the profits.

Using the best logistic model above represented by modelA and a threshold of .6803, profits are around $3,497,823, or 28.29% the perfect predictor. Of course it is still impossible to perfectly predict whether or not a person will default on a loan. However, using the model we can accept more loans and more than double profits than simply accepting every loan.

Interestingly enough the threshold used for categorizing Good vs Bad loans based on based on profit maximization(.68) is less accurate than the thresholds maximizing accuracy (.52). However, not by a significant margin (.781 and .742 respectively.)

# Results Summary

Using logistic regression, we can more accurately predict whether or not a loan will fail than chance alone. In addition to this, we can maximize profits using a logistic model. The following predictor variables were chosen as after redundancy was eliminated and those only deemed statistically significant: grade, term, debtIncRat, totalBal, accOpen24, bcOpen, amount, home, and totalAcc. The classification threshold chosen to maximize profits was .6803. This model predicted loan classification with 74% accuracy. The model also is deemed more profitable than accepting every loan outright. Room could potentially be made for improvement, the for this model only weighed in around .10. 90% of variance is unaccounted for using these variables alone, other missing variables might provide additional insight.