Project Part 3

Hamid Aboutaher

5/5/2021

# 

# 1. Executive Summary

It is clear to anyone who witnessed the financial crisis of 2008 that approving anyone and everyone for a loan was behind it. The bank industry vows to make wise lending decisions because it is a major source of their long-term revenue. By correctly predicting which loans are “good” versus “bad”, the bank can avoid accumulating costly liabilities and maximize profit. The purpose of this report is to provide a summary of a model created to increase bank profitability and discuss the suitability of this model for bank operations.

The original dataset used to create the predictive model consists of 50,000 loans and 30 customer and loan-related fields. After data cleaning and preparation, the usable dataset consists of approximately 32,500 loans. The model was trained on roughly 26,000 of these loans and tested on the remaining ~6500.

The results of the predictive model are encouraging: the model predicts loan status with **80%** accuracy compared to actual (observed) classifications. By employing the model at this level of accuracy we should get:

1 - an increase of accuracy by 9%

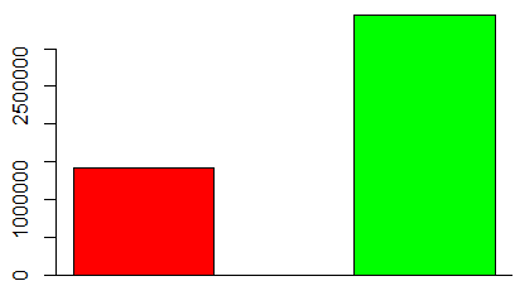
2 - an increase in good loans by 17%

3 - a decrease of bad loans by 21%

4 - maximum profit decreases by about -$569k to $3.4M.

5 - The threshold value to remain at 0.78.

**Old Model vs New Model**



2 - Introduction

This project is about building a model that predicts which applicants will likely default on bank loan and an attempt to mitigate any loss. The model we are going to build is based on a logistic regression giving a data set that includes 30 variables for 50,000 loans. The data set contains categorical and quantitative variables with missing values. The first step in preparing the data is to create a new column named response that will have the values “Good” and “Bad”. It is based on the variable status Prior to building a logistic regression, a close look at the association among variables would help classify applicants based on their loan status (fully paid vs charged off).

# 3 Data Preparation and Cleaning

Preparing and Cleaning the data: A- decide which columns to eliminate and which ones to keep. 1 - response 2- amount 3 - income 4 - home 5 - openAcc 6 - totalBalance 7 - totalLim 8 - delinq2yrs

B - dealing with missing values. C - determine the response variable. D - converting some categorical variables to numerical ones..

lapply(c("readr","dplyr","car","mice","ggformula","ggpubr"), require, character.only = TRUE)

## Warning: package 'mice' was built under R version 4.0.5

## Warning: package 'ggplot2' was built under R version 4.0.4

## Warning: package 'ggpubr' was built under R version 4.0.4

Now we can prepare our response variable based on the status column of the existing dataset. We will discard any “in progress” loans, and only keep those that are “complete” - these three status reflect completed loans, leaving 34,655 observations remaining:

loans = loans[loans$status %in% c("Fully Paid", "Charged Off", "Default"),]

Because our response variable must be binary in order to use logistic regression, we will condense the results into a 2-factor variable called “outcome”.

loans = loans %>% mutate(outcome = as.factor(if\_else(status == "Fully Paid","Good","Bad")))

## Feature Selection

In an effort to keep parsimony, we will a look at each of these variables in the dataset and evaluate if we expect them to add overall value to the model. To get us started, we can drop the “status” variable since we used it already to create our new response variable.

loans = subset(loans, select = -status)

The following variables to be removed:

loanID: this variable does not add any tangible value to the equation. employment: due to its inconsistency needs to be removed. verified: Whether it’s verified or not it just doesn’t matter. state: Whether you live in Washington state or NY, geographic location doesn’t matter. debtIncRat: This ratio does not provide meaningful information it can be obtained anytime form data. revolRatio: same as aboce. totalAcc: we deal only with active account. bcRatio": can be obtained from data when needed, now it doesn’t provide more info.

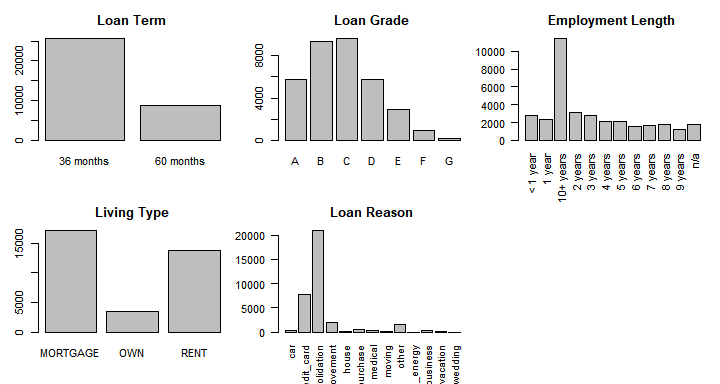
## Feature Engineering

The next step is to convert the remaining variables to character string.

``

loans = mutate\_at(loans, vars(term,grade,length,home,reason), as.factor)

Now that we have the variable’s types coded correctly, we can review the number and counts (i.e. general distribution) of categories within each of the converted variables using a barplot as a visual.



As far as grade variable which ranges from A to G, where F and G have less counts than others. So we merge F and G into new category we call it E or less. One more thing worth to mention here is that the majority of accounts are in 36 terms.

loans$grade = recode(loans$grade,"c('E','F','G')='E or less'")

The employment length variable has many different categories, and the 10+ year is skewing the distribution, so let’s combine into three categories as follows (we also have NA, and will deal with that in the next section):

* “0-4 years”
* “5-9 years”
* “10+ years”

When it comes to home variable (rent, mortgate) we are dealing with only 3 categories with frequency aboce 5%. So no adjustment needed. What is needed here is to concatenate some categories using recode function, due to their low number of observation. Not to mention the similarity in nature such as house and home improvement. Let’s take look at this variables. Home Expense = home\_improvement, house, moving, renewable\_energy Other = car, major\_purchase, medical, other, small\_business, vacation, wedding

## Missing Data

There are two variables that contain missing values - length (of employment) & bcOpen. Because “length” is a categorical variable, we won’t be able to impute its value, so these 1,823 observations will be dropped.

loans = loans[which(loans$length != "n/a"),]

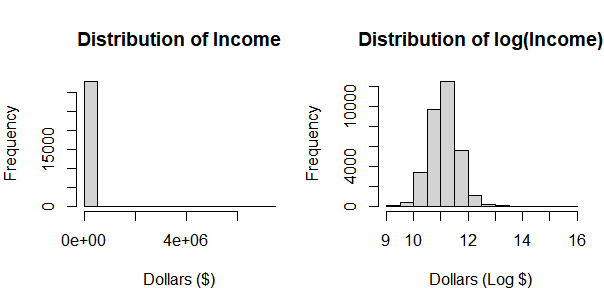
We start by imputing BcOpen since it is quantitative variable to determine the best fit values for the missing data. Using imput\_bcOpen function which takes in the original dataframe and performs the imputation with the help of the mice package then replace the na values with new values.

impute\_bcOpen <- function(df, seed){  
 index\_NA = which(is.na(df$bcOpen))  
 imputation = mice(loans, seed = seed)  
  
 for(i in 1:length(index\_NA)){  
 sum = 0  
 sum = sum + imputation$imp$bcOpen$`1`[i]  
 sum = sum + imputation$imp$bcOpen$`2`[i]  
 sum = sum + imputation$imp$bcOpen$`3`[i]  
 sum = sum + imputation$imp$bcOpen$`4`[i]  
 sum = sum + imputation$imp$bcOpen$`5`[i]  
 df$bcOpen[index\_NA[i]] = sum/5  
 }  
 return(df)  
}

## Warning: Number of logged events: 25

# 4 Data Transformation

In preparation for model fitting, we need to take a look at the distribution of our quantitative variables to understand their shape. Should a distribution appear skewed in either direction, we can attempt a transformation to make it more “normal” in shape.

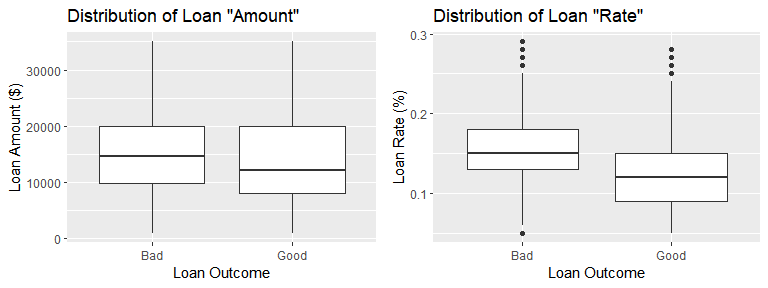


A prime example of skewed data is the “income” variable. In the first histogram we see a severely right-skewed distribution, so we attempt a log() transformation. The results of the log(income) show in the graph to the right - these are now much more normally distributed, so we will keep this transformation on the “income” variable.

A look at the remaining dollars-based variables suggests a log() transformation applied to each one with the exception of (loan) amount.

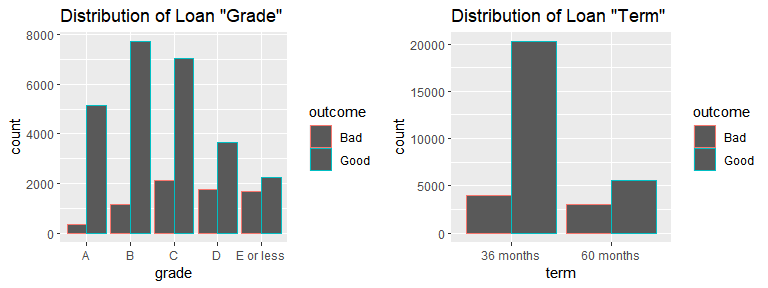
## Data Exploration

Our final step before model fitting is to explore some of the data within the variables to gain some intuition of distribution as it relates to the our outcome variable of “Good vs. Bad”.



We set the loan amounts in $ as seen in fig (boxplot). Loans that have a bad outcome seem evenly distributed in their amounts, whereas good loans appear slightly skewed to the right. To check the significance between these distribution we implement the Wilcoxon Sum Rank Test. At the 95% level of confidence, there is enough evidence to that the bad loan’s median amount is at least $1,000 greater than the good loan’s median amount (p=2.2e-16).

The second boxplot display’s loan percentage rate distribution. Both data sets appear to be normally distributed with a some outliers, however, similar to the above, the bad loans appear to have a higher median. We perform a Wilcoxon Sum Rank test to check for significance between these distributions, and at the 95% level of confidence, there is enough evidence to that the bad loan’s median rate is at least 2.99% greater than the good loan’s median rate (p=2.2e-16).

 In comparison of the two the distribution of bad vs. good outcomes based on letter grade of the loan. We see some disproportion, with lesser graded loans having a higher number of bad outcomes. Chi-squared goodness of fit to test against the theory the distributions among categories is different. The test provides enough evidence to support the distributions of bad loans are different by loan grade based on p-value of p=2.2e-16.

Finally, we review the proportions of bad vs. good outcomes for the 36- and 60-month loan terms. Visually, there appears to be a much lower proportion of bad loan outcomes in the 36 month term vs. the 60 month term, so we perform a proportion test. At a 95% level of confidence, there is enough evidence to support the proportion of bad loans in the 36-month term group is at least 18.07% less than the proportion of bad loans in the 60-month term group (p=2.2e-16).

### Section 5 - “The Logistic Model”

To build a logistic regression, we need to create a dataset and we are going to use the cross-validation method. We divide the dataset into two different sets, the training set which contain 80% of data and the testing set with 20% of data.

Next the full model is run along with the summary function to see which variables are significant and which should be included in the model. Actually, before removing some variables, we check for collinearity among variables, then remove those who have the highest “VIF” score above 10 using the VIF() function. After the removal of certain variables, we check VIF scores again, repeat the process until there are no more variables above 10.

fullmodel = glm(outcome~., data = train, family = "binomial"); vif(fullmodel)

To continue the building of the model, we use stepwise method with a forwarding direction which begins by running the null model to add to the full mode while taking in consideration the AIC score. It works by evaluating the model’s fit on the training dataset and add a penalty term for the complexity of the model. The purpose is to find the lowest possible AIC. The variables that decrease the AIC the most are the ones added to the model. We want to find the lowest AIC which indicates the best balance of the model fit. We keep repeating this process until adding one more variable would increase the AIC. At that point, maximizing the fit meaning the best version of the model has been reached.

nullmodel = glm(outcome~1, data = train, family = "binomial")  
step(nullmodel, scope = list(lower = nullmodel, upper = fullmodel), direction = "forward")

This is what the algorithm suggested is the best model: **outcome ~ grade + term + accOpen24 + home + income + payment + bcOpen + delinq2yr + avgBal + totalIlLim + rate + totalRevLim + totalRevBal + inq6mth**.

These results suggest removing inq6mth and totalIlLim because of their insignificance. and AIC score is 24809.

## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 27515 on 26264 degrees of freedom  
## Residual deviance: 24746 on 26248 degrees of freedom  
## AIC: 24780  
##   
## Number of Fisher Scoring iterations: 5

AS we can see, below is the prediction of the status of the loans using the test data. Building the confusion matrix leads us to the overall accuracy of the model. The row labels show the loan outcome, and the column labels show the model’s prediction.

## predictions  
## Bad Good Sum  
## Bad 566 5148 5714  
## Good 450 20101 20551  
## Sum 1016 25249 26265

## [1] "Proportion correctly predicted = 0.80"

A little over 20k loans correctly predicted as good loans and 567 loans correctly predicted as bad loans for a total accuracy percentage of 79%. This matches with the baseline of 79% of the loans in the dataset. While this model leaves a room for optimization in term of predicting the bad loans accurately, it does a great job at accurately predicting the good ones. We complete this same process with the holdout testing sample and determine the model would perform almost as accurate on previously “unseen” data - 0.78, or 78% accurate.

## predictions  
## Bad Good Sum  
## Bad 149 1198 1347  
## Good 124 5096 5220  
## Sum 273 6294 6567

## [1] "Proportion correctly predicted = 0.70"

Dealing with bad loans prediction and accuracy, needs a different approach. The first step is to change the threshold from 0.5. Creating a confusion matrix would be very helpful.

To calculate overall accuracy, we take the sum of the correctly predicted observations and divide it by the total number of observations. In this case (506 + 20271)/26265 = 0.79, or 79% accurate.

We complete this same process with the holdout testing sample, and determine the model performs almost as accurate on previously "unseen" data - 0.78, or 78% accurate. However, this unbalanced result does seem to indicate another look at the model’s probability threshold.

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

The question remains how to balance between bad and good predictions accuracy. In another word how to choose a threshold value to reach a good level. Thus, create a customer function that takes a vector of predicted probability outcome and a threshold value. The function will return the current threshold, the accuracy score, and the type of accuracy it computes.

def\_accuracy <- function(probabilties, threshold){  
 predictions=cut(probabilties,breaks=c(-Inf,threshold,Inf),labels=c("Bad","Good"))  
 cTab=table(test$outcome,predictions); addmargins(cTab)  
 all\_accuracy=round(sum(diag(cTab))/sum(cTab),2)  
 return(list(threshold,all\_accuracy,"overall"))  
}

We can plot the resulting dataframe onto a single line plot. With threshold along the x-axis, each variable’s accuracy value is plotted along the y-axis and given a unique color. Where the lines intercept, this is our optimal selection of the threshold value for the accuracy measure. data frame for each value of a possible threshold setting from 0.00 to 1.00 in 0.01 increments.

The black lines pinpoint this optimal value for threshold which occurs at 0.78, with the corresponding accuracies as follows:

1. *Overall* Accuracy: 0.66
2. *Bad Outcome* Accuracy: 0.65
3. *Good Outcome* Accuracy: 0.66

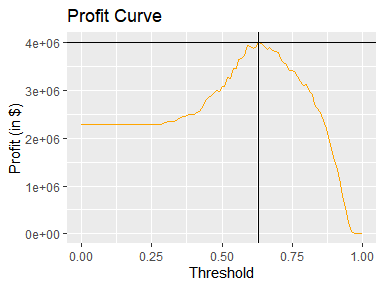
This is a very critical measurement for the model per se. To reach a perfection, we need to optimize the model and that is the top of the next section.

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

After computing the critical threshold value, it is time to compute the maximum loan profit. Which can define as the total amount paid on the loan minus the initial loan amount. We need to calculate it at each threshold level. To accomplish this task, look at the plot and find the threshold value that maximizes the profit. The test data would play a major role. We will create a custom function that uses the predicted probabilities from the “test” holdout sample as calculated in section 5.3. For each threshold value, it will return the calculated profit using that value. The function only focuses on those loans that the model predicted as having a “good” outcome per the given threshold and predicted “bad” outcome loans are removed under the assumption those would be denied if the model was in production. We also need to be mindful of the bad loans that the model predicted as good which implies lower profit than it should be.

find\_profit <- function(probabilties, threshold){  
 predictions = cut(probabilties, breaks=c(-Inf, threshold, Inf),labels=c("Bad", "Good"))  
 t\_bad <- test[which(predictions == "Good"),c("amount","totalPaid")]  
 t\_bad$profit <- round(t\_bad$totalPaid - t\_bad$amount,2)  
 return(list(threshold,sum(t\_bad$profit)))  
}

We now compute the profit for each threshold value, and plot the curve:

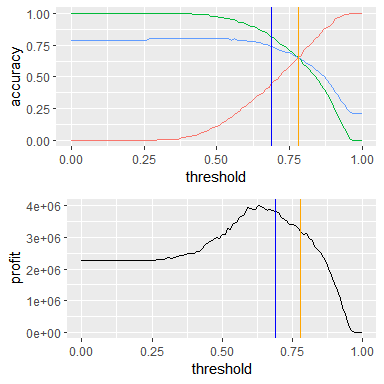


As we did in the previous section, we identify the threshold that maximizes profit of the predicted “good” outcome loans using the black lines - a threshold of 0.63, with a total of amount of profit at ~$3.99M. At this point there is urge need to eliminate the predicted bad loans by computing the ratio of profit between what the model produces vs. current state (no model in use). We used the $3.99M calculated in the previous step and divide this by the profit of the entire test dataset.

## [1] 1.757273

According to the model the profit will increase by 1.75.

As we can see applying the model would generate more than $12. M. This is approximately around 30 % profit while the current situation would generate around 7% of the total theoretical profits. Finally, we compare optimizing for profits vs. optimizing for accuracy using the graphic below to illustrate the gap and tradeoff between the two:



We can easily see that there is a difference in all three accuracies when maximizing for profit vs. accuracy. Below is a table with the numbers at these given threshold values:

## threshold accuracy outcome objective curve.color  
## 235 0.78 0.65 overall accuracy blue  
## 236 0.78 0.65 good accuracy green  
## 237 0.78 0.65 bad accuracy orange  
## 208 0.69 0.74 overall profit black  
## 209 0.69 0.82 good profit black  
## 210 0.69 0.45 bad profit black

When shifting from a model that is optimized for accuracy in exchange for a model that is optimized on profit.

### Section 8 - “Results Summary”

It all started with 30 different variables. took appropriate action in either combining categorical values where applicable, removing some variables, or transforming skewed quantitative variables. Finally, we reviewed missing data conditions and imputed values where applicable. Those observations unable to be imputed were removed (1,823 observations). Next, we split dataset into training dataset and testing dataset. Then dealt with variables that showed some sort of collinearity based on the VIF score. The final step was to test the model on the testing dataset. Finally, we had to choose optimization in favor of profit over accuracy. if we choose profit over accuracy then the model will deliver as follow:

1 - an increase of accuracy by 9%

2 - an increase in good loans by 17%

3 - a decrease of bad loans by 21%

4 - maximum profit decreases by about -$569k to $3.4M.

5 - The threshold value to remain at 0.78.