Predicting Loan Defaults with Logistic Regression

Josh Jarvey

6/5/2020

# 1.0 Executive Summary

Products within the banking industry are all typically based on a risk assessment, that is to say each offering comes with some sort of risk of “winning” or “losing” on that transaction. For instance, providing investment services to a consumer has an associated risk of increasing or decreasing in value based on numerous factors. When it comes to loans for home, auto, school, personal, etc. these products are no different - they have some associated risk of “paid in full” (with interest collected) or the loan will “default” - the win vs. loss metaphor mentioned earlier.

Historically, the ability to assess risk on loans was something more of an art than a science. Financial bankers would meet with consumers in person, and while it’s true that they would go over various details of finance and stability, the ultimate decision was grounded in intuition and “gut-feel” of that banker (in accordance to bank guidelines of course). Some were very successful at closing the right ones and making a profit for their organization, while others were not as successful. There’s also the ethical dilemma of bias being introduced into gut-feel and intuitive decision making, which can lead to missed revenues, but also potential legal troubles that organizations are wise to mitigate.

In today’s age, the science has caught up with the art - and in many cases has even surpassed it. Where you have had one or two individuals trying to “crunch the numbers” and come up with a decision in the times past, we now have computers to do this work. But where the computers and science really excel at this work is finding the hidden relationships between a consumer’s relevant “parameters” to the loan and aiding in the decision of win vs. loss. It crunches the numbers among thousands upon thousands of historical loan’s parameters, and understanding those outcomes it can provide a sound assessment of risk that’s rooted in science and not intuition.

In this project we will explore the use of one of these scientific tools called Regression. Being that our end goal is to have a decision to “give loan” or “decline loan”, we will use the Logistic version of the regression analysis, which is better suited for a yes/no decision. A more scientific approach to decision making will not only save time and money to improve the bottom line, but will also help to mitigate bias and the associated organizational risk that comes with it.

# 2.0 Introduction

In risk-based decision making, it can be hard for any one human to take into account all the necessary information and variables that go into making the *right* decisions that minimize losses and maximizes revenues. This is where Regression analysis can come into play, because it is designed specifically to tackle the analysis of many variables, at the same time, to predict an outcome. Since the end result of the product in this study is a successful financial loan, we know that the outcome will be one of two choices: either “paid in full” as a success, or “loan default” as a failure. Because of this binary in nature outcome, we will use the Logistic version of the Regression analysis, which is suited well for handling this type of task.

The remainder of this report will outline the steps taken to build, train, and test different candidate models to see which performs the best and will ultimately be released into production to serve its purpose of aiding in the loan decision making process.

As with any data project, we start out by summarizing the dataset of 50,000 observations both numerically and visually to explore it for anomalies - missing data, outliers, etc. - and determine how to best overcome these data quality issues. In the next step we’ll explore the dataset’s 30 unique variables. Some of these variables will be significant toward solving the problem, whereas some will not and will need to be removed from the model or lumped together in an “other” category. Once we’ve landed on the appropriate variables, we’ll take a look at the distribution of its values as data may need to be transformed depending on what is discovered.

In the second half of the project, we will describe fitting the model to the data and accessing its performance. From here we will test at optimizing for prediction accuracy in addition maximizing profit. Finally, we will end with a conclusion and summarize model performance, along with any room for enhancements.

# 3.0 Data Preparation and Cleaning

To get started in this step, first we will load the necessary packages we need in order to perform the data exploration and cleaning.

library(readr)  
library(dplyr)  
library(car)  
library(mice)

Now 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”. We can now drop “status” from the dataset altogether.

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

## 3.1 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)

Below are a list of the other variables we’ll remove from the dataset in the order of which they occur:

* “**loanID**”: The id in which a loan can be retrieved from the database. This adds no information.
* “**employment**”: A free form entry field that offers no consistency between values. Riddled with short hand and spelling mistakes.
* “**verified**”: Whether the source of income has been verified. If the bank does not have this verified, we would not make the loan.
* “**state**”: With loans in an overwhelming majority of the states, this seems more noise than signal.
* “**debtIncRat**”: Any ratios are a calculation of existing data within this set, so it will be removed since it provides no extra information.
* “**revolRatio**”: Any ratios are a calculation of existing data within this set, so it will be removed since it provides no extra information.
* “**totalAcc**”: Total accounts (including closed) - We’re only interested in open accounts that are currently active.
* “**totalPaid**”: Total amount repaid to the bank. This comes after a loan is issued, so it cannot be used as a predictor.
* “**bcRatio**”: Any ratios are a calculation of existing data within this set, so it will be removed since it provides no extra information.

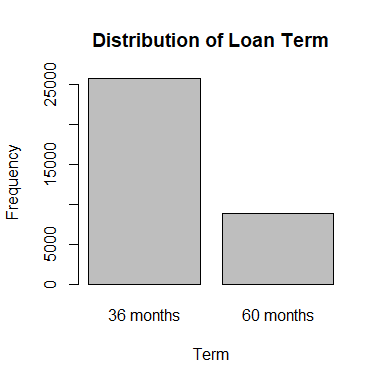
## 3.2 Feature Engineering

Next we will take a look at the remaining variables with our str() function to check the variable types in our dataset.

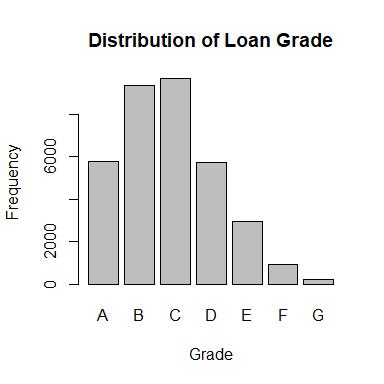
We see right away there are variables that are not coded appropriately (as a character, instead of as a factor), so we will make those adjustments now before moving forward.

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 of categories within each of the converted variables using a barplot as a visual.

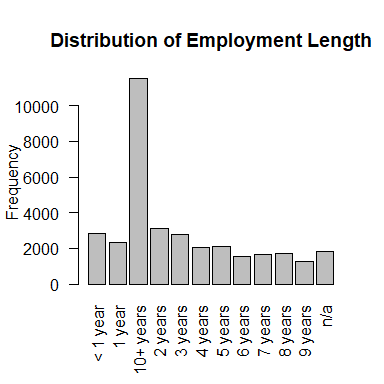


First, we take a look at the loan term variable. While 36 month loans account for the vast majority, there are only two categories for this variable, so we wont make any adjustments on this one.



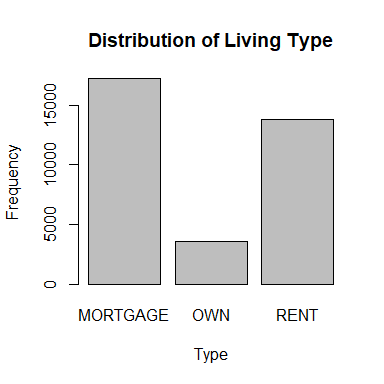
Next, we take a look at the grade variable, which ranges in values from A to G. In reviewing the summary, we notice the F and G groups have a considerably less count than the other grades, so we’ll combine these two groups into the “E” grade and form a new category “E or less”.

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

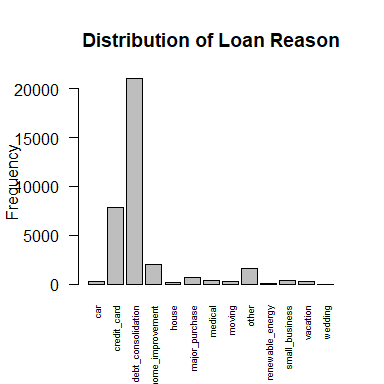


The employment length variable has many different categories, and the 10+ year is skewing the distribution, so let’s combine into three categories as follows (we also have NA, and will deal with that in the next section):

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



We now review the home variable, which explains the type of living arrangements: rent vs. mortgage vs. owning. There are only three categories here, and the frequencies for each category look to be above 5% of the total number of observations, so we wont be making any adjustments here.



Finally, we have loan reason. Similar to the employment length variable, we will need to combine categories as some of these contain a very low number of observations. There also seems to be similar in nature categories such as “house” and “home improvement”, so those will be a natural combination. The combined groups will be as follows:

* “**Credit Card**” = credit\_card
* “**Debt Consolidation**” = debt\_consolidation
* “**Home Expense**” = home\_improvement + house + moving + renewable\_energy
* “**Other**” = car + major\_purchase + medical + other + small\_business + vacation + wedding

## 3.2 Missing Data & Imputation

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"),]

BcOpen, however, is a quantitative variable, therefore we can try an imputation process to determine “best fit” values for these missing data. We create a custom function impute\_bcOpen() that takes in the original dataframe, performs the imputation process using the mice() package, and replaces the NA values with the imputed values.

impute\_bcOpen <- function(df, seed){  
 #create a vector of the indices that are "NA"  
 index\_NA = which(is.na(df$bcOpen))  
   
 #complete the imputation process  
 imputation = mice(loans, seed = seed)  
  
 #iterate through each missing value and calculate the average of the imputed set, replace NA with the imputed value.  
 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