Predicting Which Loans Default

Robert Dusel

7/27/2021

## Executive Summary

Which loans are good and which are bad? This project builds a model that attempts to answer that question and shows how its implementation can increase profit.

##### Building The Model

Think of this model as a tool. Like most tools, this model is designed to make a specific task easier. That task is to determine which loans are good and which are bad. The model was built off of 27,000 loans. Each loan was either good or bad, but also came with 31 different characteristics. These characteristics pertain to the loan or loanee. Characteristics like the term of the loan and the income level of the loanee. The model is created from the associations that exist between the status of the loan and these 31 characteristics. For example, a loan with an “A” grade is more likely to be a good loan. From all these different associations, the model gives a prediction of the loan status. The end result is this. If given a set of those 31 characteristics, the model gives a probability that the loan is good.

##### How the Model Performs

The model gives you the probability of a good status loan, but it is up to the users to determine what probability is high enough to actually accept that prediction. This is called threshold. If we set the threshold to 0.46 (loans given a probability of 0.46 or more are considered good), the model successfully predicted 79.32% of the loans in a 7,000 loan test set.

##### How the Model Performs - Profit

If every loan in the 7,000 loan test set was considered good, The bank would have awarded 7,000 loans resulting in $2,290,174 of profit. With our model set at a threshold of 0.675, the bank would have awarded 5458 loans at a profit of $4,377,690. That is a 91% increase in profit.

##### Recommendations/Conclusion

I recommend the implementation of this model set at a threshold of 0.675 as a tool used to decide whether or not the bank should award a loan. Use of this model, over just assuming all loans are good, would result in a 91% increase in profit.

## Introduction

Banks face a big challenge when it comes to deciding whom to loan to. Knowing who will pay back a loan in full and who will default is worth a lot to a bank. There is no crystal ball that can tell a bank who they can bet on, but there is data. This report will review some of that data and, with the help of logistic regression, attempt to predict who will default on a loan.

The task begins with preparing, cleaning, and transforming the data. From there, the logistic model can be built. Once the model is built, it will be optimized for accuracy and then for profit. Let’s begin.

## Preparing and Cleaning the Data

The data comes from <https://datascienceuwl.github.io/Project2018/TheData.html> and has 50,000 observations with 32 variables.

The first thing to complete in prep is to make the response variable. The variable “status” fills this role. The values of “status” that are of interest are ‘Fully Paid’, ‘Charged Off’, and ‘Default’. ‘Fully Paid’ is going to be changed to ‘Good’ and the rest are going to be changed to ‘Bad’. One status value of the 50,000 data points was missing. That whole observation was dropped.

*#libraries used in this report*  
**library**(tidyverse) *#mostly for dataframe manipulation*  
**library**(DMwR2) *#knn*  
**library**(ggformula) *#graphing*  
**library**(patchwork) *#graphing*  
**library**(knitr)*#tables*  
**library**(kableExtra)  
*#load the data*  
loans50k <- **read\_csv**('loans50k.csv')  
*#only keeping fully paid, charged off, and default loan status*  
loans50k <- loans50k **%>%**  
 **filter**(status **%in%** **c**('Fully Paid', 'Charged Off', 'Default'))  
*#updating the status variable with 'good' or 'bad'*  
loans50k <- loans50k **%>%**  
 **mutate**(status = **case\_when**(  
 status **==** 'Fully Paid' **~** 'Good',  
 status **==** 'Charged Off' **|** status **==** 'Default' **~** 'Bad'))  
*#making status a factor*  
loans50k <- loans50k **%>%**  
 **mutate**(status = **factor**(status))

With the response variable created, it is time to decide on variables to remove. One variable that can go is loanID. It is just a unique identifier for each observation and has zero predictive value.

*#dropping loanID*  
loans50k <- loans50k **%>%**  
 **select**(**-**loanID)

The rest of the variables will stay and here is the reasoning. This report is being done by a person with little domain knowledge on loans and banking. Keeping a variable, at worst, is keeping a variable with low predictive power. Removing a variable, at worst, is removing a variable with high predictive power.

Feature engineering is next. There are quite a few variables that can be changed from quantitative to categorical. The variables are:

delinq2yr - number of 30+ day late payments in last two years  
inq6mth - number of credit checks in the past 6 months  
pubRec - number of derogatory public records including bankruptcy filings, tax liens, etc.  
accOpen24 - how many accounts were opened in the past 24 months  
openAcc - number of open credit lines  
totalAcc - total number of credit lines in file, includes both open and closed accounts

What these variables all have in common is that they are discrete. The code to change these variables to categorical is all very similar, so only one example will be displayed. Consult the RMD file for the others.

*#creates category bins for delin2yr*  
loans50k <- loans50k **%>%**  
 **mutate**(delinq2yr = **case\_when**(  
 delinq2yr **==** 0 **~** '0',  
 delinq2yr **==** 1 **~** '1',  
 delinq2yr **==** 2 **~** '2',  
 TRUE **~** 'more than 2'))

Employment could be an important variable. It is populated with job titles and, as far as I can tell, every entry equates to a job possessed by a person. There are 1,918 missing values for this variable. They could represent everything from a mistake happening during data collection, to unemployed people. I think the missing data is informative so it will stay, but be renamed to ‘Unknown’. The employment variable will be changed to a categorical variable with two levels, ‘Employed’ and ‘Unknown’.

*#employment - changes to employed or unknown*  
loans50k <- loans50k **%>%**  
 **mutate**(employment = **case\_when**(  
 **is.na**(employment) **~** 'Unknown',  
 TRUE **~** 'Empolyed'))

Length is a variable that describes how long an applicant has been continuously employed. There are missing values in this variable, and will be kept because they may be informative. This categorical variable is spread out over many levels, and will be condensed to five levels. The 5 levels start at Unknown and end at 10+ years.

*#combining some years and renaming the n/a values as 'unknown' for consistency*  
loans50k <- loans50k **%>%**  
 **mutate**(length = **case\_when**(  
 length **==** '10+ years' **~** '10+ years',  
 length **%in%** **c**('9 years', '8 years', '7 years',   
 '6 years', '5 years') **~** '5 - 9 years',  
 length **%in%** **c**('4 years', '3 years',   
 '2 years', '1 year') **~** '1 - 4 years',  
 length **==** 'n/a' **~** 'Unknown',  
 TRUE **~** length))

That is it for the categorical variables. The factor function is now used to convert all the categorical variables to factors. Consult the RMD file to see how this is done.

So far, missing values in the data have been kept missing and rebranded. There are three more variables with missing data still. They are:

revolRatio - proportion of revolving credit in use bcOpen - total unused credit on credit cards bcRatio - ratio of total credit card balance to total credit card limits

These values will actually be replaced. They will be replaced using the k-nearest neighbors algorithm.

*#replaces missing values using KNN*  
loans50k <- **knnImputation**(loans50k)

## Exploring and Transforming Data

Cleaning is done. It is time to explore the data, look for interesting relationships, and transform variables if needed. Transformations will happen first. Continuous quantitative variables are the ones that may need a transformation. Some variables don’t need any extra touch, because they have an approximately symmetric distribution, but there are quite a few that need a transformation. The following variables are receiving a cube root transformation. The cube root works well for these variables because they are right skewed and they all have many values equal to, or very close to, zero. The variables receiving this transformation are:

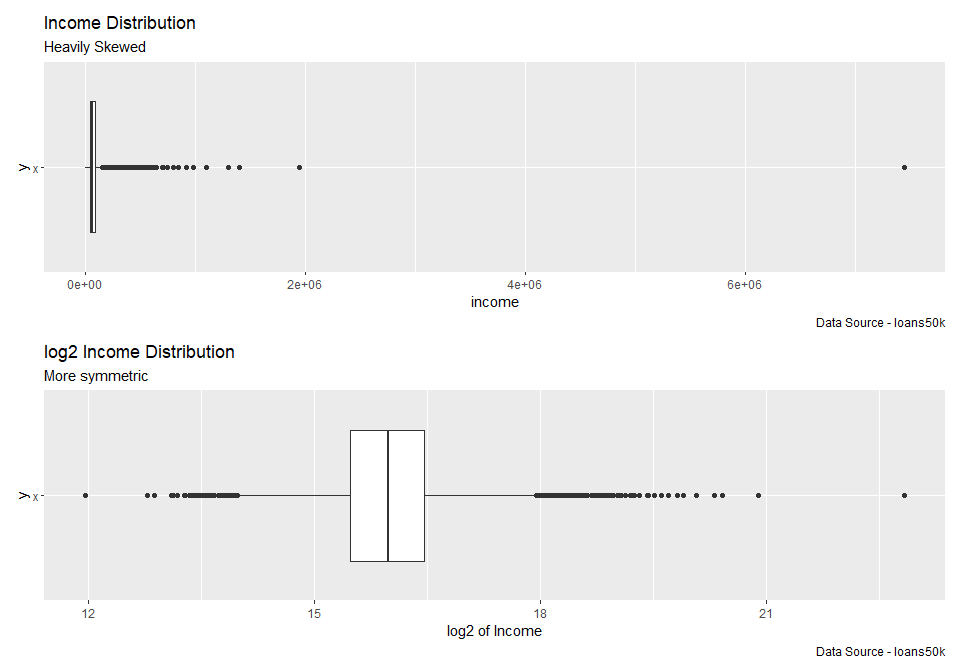
payment - monthly payment amount  
totalBal - total current balance of all credit accounts  
avgBal - average balance per account  
bcOpen - total unused credit on credit cards  
totalLim - total credit limits  
totalRevBal - total credit balance except mortgages  
totalBcLim - total credit limits of credit cards  
totalIlLim - total of credit limits for installment accounts

Consult the RMD file to see how these variables are transformed.

There is one other variable that could use a transformation. It is the income variable that measures the annual income in dollars. It is heavily skewed to the right. A log2 transformation will be done on this variable because it is a powerful transformation. The minimum value for income is 1000, so that is another reason why the log2 transformation works well.

*#creating the transformed variable - log2*  
loans50k <- loans50k **%>%**  
 **mutate**(income\_log2 = **log2**(income))

Here is the result of that transformation.

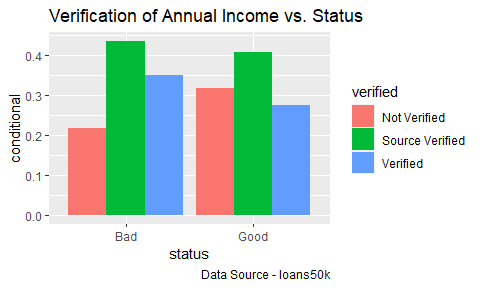


With all of the transformations complete, removal of the old variables can happen.

*#removing the columns that were transformed*  
loans50k <- loans50k **%>%**  
 **select**(**-c**(payment, totalBal, avgBal, bcOpen, totalLim, totalRevBal, totalBcLim, totalIlLim, income))

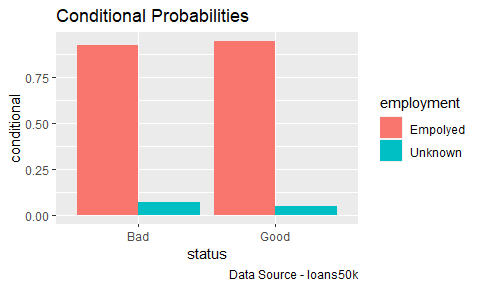
Time to explore the data. Conditional bar graphs are a great way to see any relationships in categorical variables. Here is one looking at status along with the verified variable. Code for the first graph is shown and the code for the other bar graphs is similar.

*#finds total number of good and bad loans*  
gb\_verf\_den <- loans50k **%>%**  
 **group\_by**(status) **%>%**  
 **summarise**(den = **n**())  
*#finds breakdown of verification, based on good or bad status*  
gb\_verf\_num <- loans50k **%>%**  
 **group\_by**(status, verified) **%>%**  
 **summarise**(num = **n**())  
*#joins dfs*  
con\_gb\_verf <- **left\_join**(gb\_verf\_num, gb\_verf\_den)  
*#adds a column with the conditional probabilities*  
con\_gb\_verf <- con\_gb\_verf **%>%**  
 **mutate**(conditional = num **/** den)  
*#conditional bar graph*  
**gf\_col**(conditional **~** status, fill =**~** verified, position = **position\_dodge**(), data = con\_gb\_verf) **%>%**  
 **gf\_labs**(title = 'Verification of Annual Income vs. Status',  
 caption = 'Data Source - loans50k')



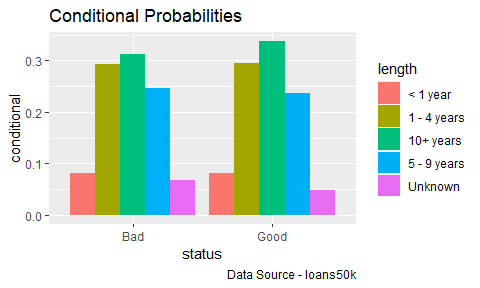
They have similar breakdowns. Given that an applicant’s income was verified, they were more likely to have a bad status. A higher proportion of the good applicants were not verified compared to the bad. That seems counter-intuitive.

Here is another conditional bar graph, this time looking at the employment variable and how it interacts with status. Remember, we changed the employment variable to a two level factor, the levels being ‘Employed’ and ‘Unknown’. Let’s look at the graph.



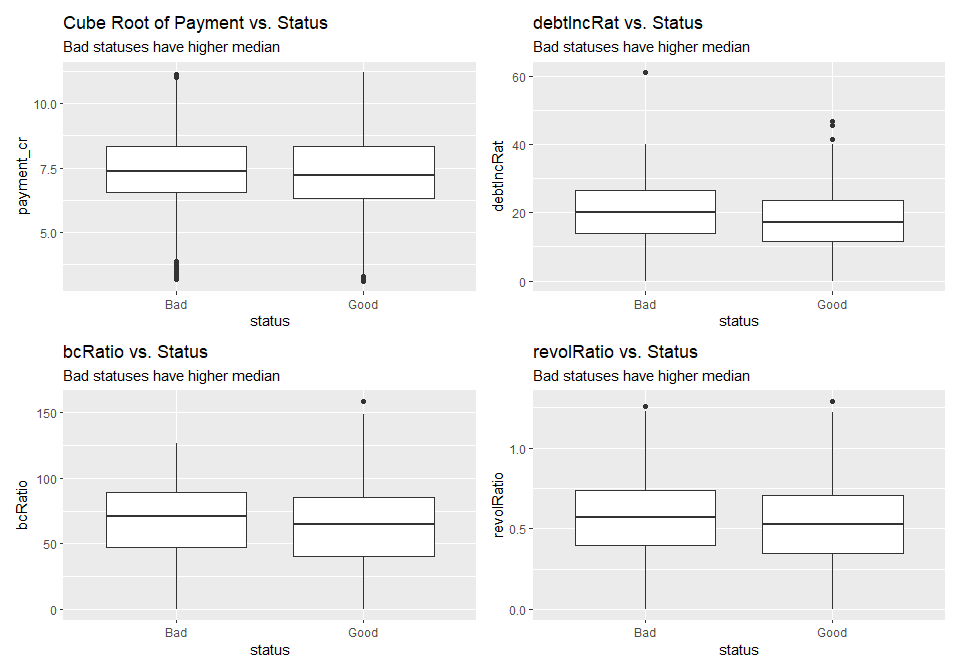
Of the bad status, 7.2% were unknown, whereas 5.01% of the good were unknown. I left the missing variables in because they may be informative. Given an applicant has an unknown employment, they were more likely to be in bad status.

One more conditional bar graph. This time with the length variable. The length variable measures how long an applicant was continuously employed.



The two distributions look similar. Again, the bad status has a higher proportion of Unknown length. The bad status has a noticeably less 10+ years proportion.

Time to review some quantitative variables. Side-by-side boxplots are a good way to compare distributions.



The boxplots above show some differences. Applicants with a bad status have higher median debt to income ratios (debtIncRat), higher median revolving credit ratios (revolRatio), higher median credit card balance to total credit card limits ratios (bcRatio) and higher median cube root of payment.

## The Logistic Model

It is finally time to start model building. Our model needs data to train on, so the first thing to do will be to create a training data set. The training data is going to be an 80% subset of the loans data set, but chosen randomly. Before the random sample, an idex column is added back into the data. This index provides a unique identifier for each observation that just aids writing code for the subsetting process you will see later. Let’s create the training set!

*#adding back a column of indexes (this helps identifying each row)*  
loans50k <- loans50k **%>%**  
 **mutate**(index = **seq**(1, **nrow**(loans50k)) )  
*#seed set*  
**set.seed**(321)  
*#training set*  
training\_set <- **sample\_n**( loans50k, **round**( **nrow**(loans50k) **\*** 0.8) )

Great, the model can now be trained. How does the model get tested? That’s where the remaining 20% of the loans data comes in. The data not chosen to be in the training set will be used as a test dataset. After the creation of this set, the index column that was created can now be removed.

*#test set*  
test\_set <- loans50k **%>%**  
 **filter**( **!**(index **%in%** training\_set**$**index) )  
*#removing index*  
training\_set <- training\_set **%>%**  
 **select**(**-**index)  
test\_set <- test\_set **%>%**  
 **select**(**-**index)

The removal of the totalPaid column from the training set is also necessary. This information is directly tied to the status of the loan and the bank wouldn’t have this information to make predictions on.

*#removing totalPaid column from training set*  
training\_set <- training\_set **%>%**  
 **select**(**-**totalPaid)

It is time to create the model. Because loan status is binary, it is either good or bad, logistic regression is going to be used to predict it. This report will focus on a full first order model. This means the predictors in our model will be every single explanatory variable, but none of their interactions.

*#full model*  
full\_model <- **glm**(status **~** ., family = 'binomial', data = training\_set)

You will be spared the pages of output the summary of this model wants to show you. It will be included in the supplementary RMD file. With the model trained and completed, It is time to make some predictions on the test data set.

*#predicting status of the test set*  
predictions <- **predict**(full\_model, test\_set, type = 'response')

The way this works, the model is going to give a probability to every observation in the test data set. The probabilities range from 0 to 1. Probabilities closer to one mean the model is predicting a loan with a good status, while probabilities closer to 0 mean a loan with a bad status. These probabilities have all been saved in the predictions vector.

So what probability is high enough to constitute a good loan? The probability that is set to determine a good loan is called a threshold. With the threshold set at 0.50, observe how the model performs with this classification table:

*#class table*  
threshold <- 0.5  
Predicted.Status <- **cut**(predictions, breaks=**c**(**-**Inf, threshold, Inf),   
 labels=**c**("Bad", "Good"))  
cTab <- **table**(test\_set**$**status, Predicted.Status)   
**addmargins**(cTab)

## Predicted.Status  
## Bad Good Sum  
## Bad 199 1274 1473  
## Good 184 5274 5458  
## Sum 383 6548 6931

The last number in each row of the table represents actual counts, while the columns are predicted counts. With this table, the proportion of good loans predicted as good, the proportion of bad loans predicted as bad, and overall correct prediction proportion can all be calculated.

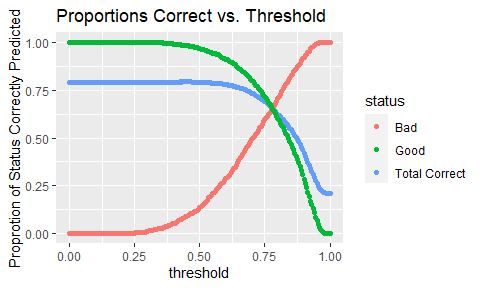
The results are lackluster. At the 0.50 threshold, the model only found about 13% of the bad loans and even misidentified over 3% of the good loans. This resulted in 78.96% of the predictions being correct. If a person had just guessed that all of the loans in the test set were of good status, they would be 78.74% correct. Our model, as of right now, doesn’t appear to be adding much. Let’s further explore how threshold changes what our model produces and later see how adding profit to the mix may change our view on this model.

## Optimizing the Threshold for Accuracy

The best way to show how threshold changes the predictive power of the model is through a graph. Before the graph is made, data must be collected. With a loop, the correctly predicted proportions of our loans at many different thresholds can be collected.

*#predicting status of the test set*  
predictions <- **predict**(full\_model, test\_set, type = 'response')  
*#initializing proportion vectors*  
pcp <- **c**() *#total*  
bp <- **c**() *#bad*  
gp <- **c**() *#good*  
*#threshold*  
th <- **c**()  
*#testing different thresholds*  
**for** (i **in** **seq**(0, 1, by = 0.005)){  
 *#saving the threshold and adding it to the threshold vector*  
 threshold <- i   
 th <- **append**(th, threshold)  
 *#this creates the table we saw previously, we extract the numbers we want from it*  
 predStatus <- **cut**(predictions, breaks=**c**(**-**Inf, threshold, Inf), labels=**c**("Bad", "Good"))  
 cTab <- **table**(test\_set**$**status, predStatus)   
 cTab <- **addmargins**(cTab)  
 *#the various proportions we are interested in*  
 b <- cTab[1] **/** cTab[7] *#bad*  
 g <- cTab[5] **/** cTab[8] *#good*  
 p <- (cTab[1] **+** cTab[5]) **/** cTab[9] *#total*  
 *#those proportions being added to their respective vector*  
 bp <- **append**(bp, b)  
 gp <- **append**(gp, g)  
 pcp <- **append**(pcp, p)}

With the data collected, here is the graph.



The graph showcases how threshold affects the proportion of loans correctly predicted. The general trend is that as threshold increases, bad loans are correctly predicted more, but more good loans are incorrectly predicted. There is a trade off happening here. What threshold produces the highest overall prediction percent? Let’s look:

*#threshold and total proportion correct*  
x <- **data.frame**(th, pcp)  
x **%>%**  
 **filter**(pcp **==** **max**(pcp))

## th pcp  
## 1 0.46 0.7932477

The highest overall proportion of correct predictions happens at a threshold of 0.46 and is 79.32%. It was 78.96% at a .50 threshold, so this exercise garnered an extra 0.36%. This now puts our model 0.58% above the person that just guesses that every loan is good. Still doesn’t seem to be performing that well.

## Optimizing the Threshold for Profit

Here is where the model starts showing off its value. Banks, like any other business, need to make money. Can the model help maximize profit? To begin answering this question, a new column is created in the test dataset. A profit vector is created by taking the total amount a person paid (totalPaid) and subtracting the amount the loan was for (amount).

*#profit vector*  
profit <- test\_set**$**totalPaid **-** test\_set**$**amount

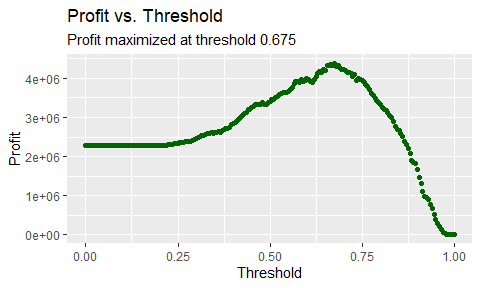
The profit vector is combined with the prediction vector created earlier. The profit for each loan now has an associated prediction probability.

*#profit with predictions dataframe*  
profit\_df <- **data.frame**(profit, predictions)

It is time to calculate some profits. Here is how it works. If the threshold is set to 0.50, every loan, whether good or bad, that has a prediction probability of 0.50 or above will be considered good. The profit of these loans will then be added up, while the profit for loans considered bad will be discarded. That profit and the threshold that produced it are recorded and added to a new vector. A loop will allow the testing of many different thresholds and the collections of many different profits.

*#initialize vectors*  
th <- **c**()  
model\_profit <- **c**()  
*#loop*  
**for** (i **in** **seq**(0, 1, by = 0.005)){  
 *#sets threshold and adds it to the threshold vector*  
 threshold <- i  
 th <- **append**(th, threshold)  
 *#filters profit\_df to only have the profit numbers with a high enough prediction*  
 prof\_sum <- profit\_df **%>%**  
 **filter**(predictions **>=** threshold)  
 *#adds the sum of the profit to the model\_profit vector*  
 model\_profit <- **append**(model\_profit, **sum**(prof\_sum**$**profit))}

With the thresholds and their corresponding profit amounts recorded, let’s look at a graph.



The graph clearly shows how profit changes with threshold. The maximum profit of $4,377,690 occurs at a threshold of 0.675. If the model wasn’t used, and every loan in the test set was just considered good, profit would be at $2,290,174. Our model increased profits by over 91%! Almost double. The person guessing that every loan in the test set is good is finally starting to look dumb. Let’s explore this 0.675 threshold further.

## Predicted.Status  
## Bad Good Sum  
## Bad 648 825 1473  
## Good 886 4572 5458  
## Sum 1534 5397 6931

So at this threshold, 84% of good loans are predicted correctly, 44% of bad loans are predicted correctly, and 75% of loans overall are predicted correctly. There is clearly room for improvement here. A lot of profit is being missed out on. A perfect model would produce $12,912,742 in profit! The 0.675 threshold for maximum profit is a lot different than the 0.46 threshold for maximum accuracy. This appears to be the case because bad loans are so bad for profit, it is worth losing some good loans to find the bad ones.

## Results Summary

The full first order model is recommended. A threshold of 0.675 should be used. This means that loans that the model gives a prediction of 0.675 or greater are considered good loans. At this threshold, profit increased by 91%, 84% of good loans were correctly predicted, 44% of bad loans were correctly predicted, and 75% of loans overall were correctly predicted.