DS 705 Final Project

Mark Johnston Olson

06/29/2021

### Executive Summary

In this work, we set out to create a statistical model that can predict whether a loan is bad or good. The model we built is very effective at detecting bad loans, and it would have increased your profits by as much as $3 million if you had used it on the loans we tested it on. You should certainly implement the model, but we think we could refine it further.

Your overall accuracy in predicting good loans was quite high – almost 80%. In spite of this, your profit was less than 10% what it would have been if you had accepted all good loans and rejected all bad loans, because the worst loans resulted in massive losses to the bank. Of course, we never got to see the data on loans that you rejected.

To build our model, we examined all the data that the bank collected on 34,665 recent loans. There were 30 different factors that you collected data on; we rejected 10 of these that seemed unlikely to have predictive value and used 20 factors to make a model that predicts the probability that a loan would be good.

Next, we randomly selected 80% of the loans to serve as a “training” dataset and 20% of the loans to serve as a “testing” dataset. We made a model to be as accurate as possible on the “training” dataset, and then looked at how well it did on the “testing” dataset. The numbers we report refer to the “testing” dataset alone, which we believe is representative.

The model itself can only predict the probability that a loan will be good. It is up to the user to determine the “classification threshold” – how high a probability of payoff the model must predict before you are willing to accept the loan.

For example, you could set a classification threshold of 0, accepting even the loans that the model would have indicated had a very low probability of paying off. At the other extreme, you could set a classification threshold of 1 and reject every loan application out of hand.

Clearly the best answer is somewhere between these two extremes. It turns out that the best model we found is most effective when you reject any loans the model predicts are less than 70% likely to pay off.

Although the best model with a profit-maximizing classification threshold of 70% was far from perfect, it would have increased your profit by $3 million on the 20% of loans in the testing set just by rejecting the worst half of the bad loans and only a fifth of the good loans. This would mean issuing 22% fewer loans.

In conclusion, our model could nearly quadruple your profits based on testing results, but we also think we could potentially build a model that’s even better.

### Introduction

Our goal in this project is to develop a logistic regression model using diverse categorical and numeric historic data about prospective borrowers to predict if they will default on loans.

The loan data being examined includes 50,000 loans ranging in size from $1,000 to $35,000 to borrowers in the District of Columbia and every US state except Iowa and Idaho.

### Preparing and Cleaning the Data

We considered only loans that were paid off (“good” loans) and loans with a status of “Default” or “Charged Off” (“bad” loans).

This left 34,655 loans, of which 27074 were good and 7581 were bad.

The table of loans initially had 32 columns.

As any user of databases might expect, the *loanID* column is useless to us; it’s just a primary key for uniquely identifying each loan.

The *employment* column also turns out to be useless, because there are 20,584 distinct jobs in that column. Perhaps some advanced regular expressions could bin the jobs listed here into a more manageable number of categories, but we won’t do that.

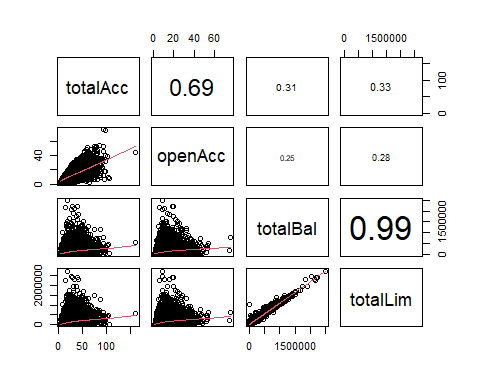
Every other categorical column has 50 or fewer distinct values, so *employment* is the only categorical column that clearly needs to go.

None of the numeric variables have fewer than 7 distinct values, so we won’t try converting any numeric variables to categorical variables.

One of the variables, “totalPaid”, can only be determined after the loan has been issued, so we can’t use it as a predictor for status.

Next we’ll look for variables that are strongly correlated with other variables. If two potential predictors are very strongly correlated, we only need one of them in our final model, because they’re redundant.

Unfortunately, the high dimensionality of the data means that we can’t visualize the full correlation matrix of all the numeric variables.

But here’s an example of how some variables are correlated with each other. 

We notice that totalAcc and openAcc are strongly correlated with each other, and totalBal and totalLim have an almost perfect correlation.

By inspection of the full correlation matrix (not shown), it can be shown that payment and amount are strongly correlated, and totalIlLim and totalLim are both strongly correlated with other variables thatWewon’t drop.

Finally, bcRatio is moderately correlated with bcOpen (corr = -0.535).

SoWewill consider dropping the totalBal, openAcc, totalLim, payment, bcOpen, and totalIlLim columns.

At this point, we can deal with missing values in the columns we haven’t yet removed.

#count the NA values in each remaining column  
nancts <- sapply(loans, function(x) {sum(is.na(x))})  
nancts

## amount term rate grade length home   
## 0 0 0 0 0 0   
## income verified status reason state debtIncRat   
## 0 0 0 0 0 0   
## delinq2yr inq6mth pubRec revolRatio totalAcc totalPaid   
## 0 0 0 15 0 0   
## totalRevLim accOpen24 avgBal bcRatio totalRevBal totalBcLim   
## 0 0 0 384 0 0   
## profit   
## 0

There are 384 missing values in bcRatio, and almost no missing values in any other columns.

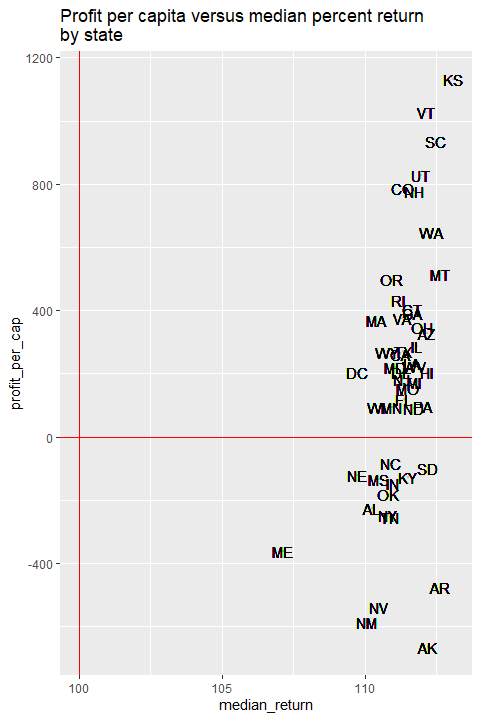
Because there are so few missing values (at most about 1% of any column) and by far the cheapest nondestructive way to deal with missing values is median/mode imputation,Wewill use median/mode imputation to fill missing values.

loans <- imputeMissings::impute(loans, method = 'median/mode')

### Exploring and Transforming the Data

It is tempting to remove the “state” variable, because it has 49 categories, and that could make the logistic model difficult to work with.

However, there is a surprising amount of variation between states!



Let’s look at this plot of two metrics of loan performance broken down by state.

The x-axis is statewide median percent return on investment in the state (the median ratio of loan amount paid back to amount initially loaned).

The y axis is average profit (amount paid back - loan amount) per person in the state.

When we look at the states this way, it’s obvious that states can be roughly categorized into groups based on how profitable the average loan in that state is.

For example, Kansas (KS) clearly has a lot in common with Vermont (VT), but not Alaska (AK), at least as far as paying back these loans is concerned.

As suggested by the graph, we’ll group the states into four groups: “excellent”, “good”, “bad”, and “awful”, and drop the original “state” column.

loans$stategroup <- case\_when(  
 loans$state %in% c('KS','VT','SC','UT','CO','NH') ~ 'excellent',  
 loans$state %in% c('NC','SD','KY','IN','MS','NE','OK','AL','NY','TN') ~ 'bad',  
 loans$state %in% c('ME','AR','NV','NM','AK') ~ 'awful',  
 TRUE ~ 'good'  
)  
loans <- loans %>% select(-state)

The “length” column also has 12 categories, which is a few too many. We’ll remake this into five categories as well: “0-3 year”, “4-6 years”, “7-9 years”, “10+ years”, and “n/a”.

We now have only manageably sized categorical columns, except “reason”, which has 13 categories. Unfortunately, it’s not clear how to reduce the number of categories in “reason”.

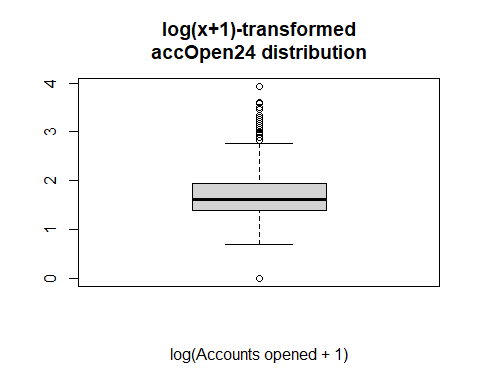
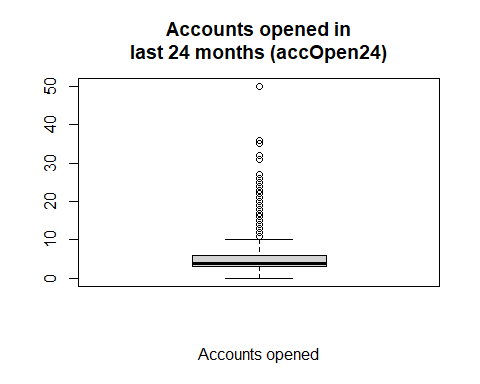
Now we’ll investigate skewness of numeric variables.

Most numeric variables had very non-normal distributions. For one, some of the numeric variables had very few values.

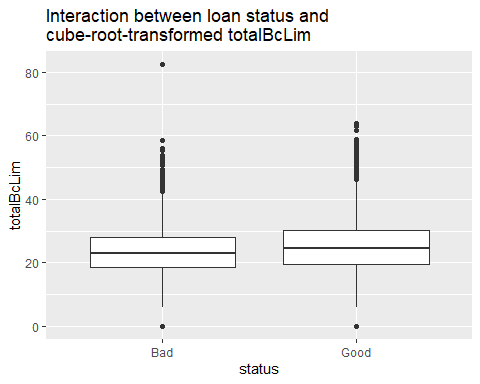
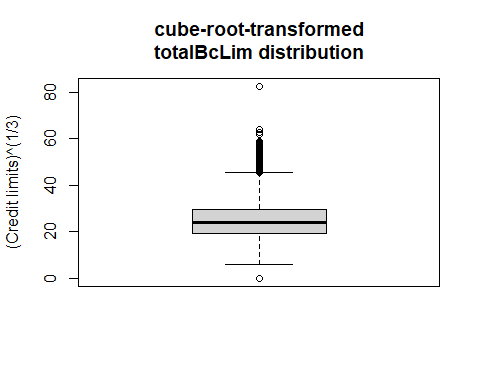
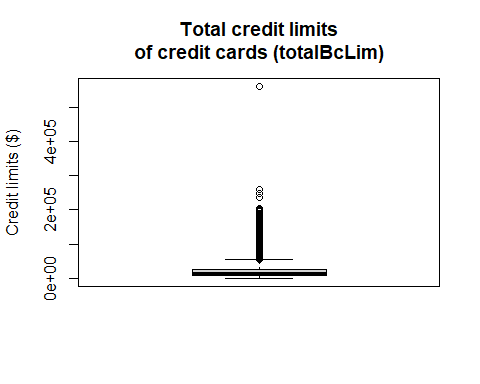
inq6mth, an indicator of how many credit checks an individual made in the past 6 months, has only 7 distinct values, so it makes sense to transform inq6mth into a categorical variable.

Some variables like are heavily right-skewed and need to be log-transformed.

However, if a variable is both highly right-skewed and has some values of 0 (like accOpen24), the best transformation is probably log(x+1), which ensures that 0 values get included in the transformed variable.

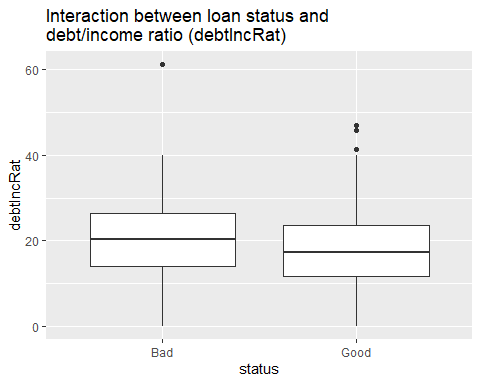
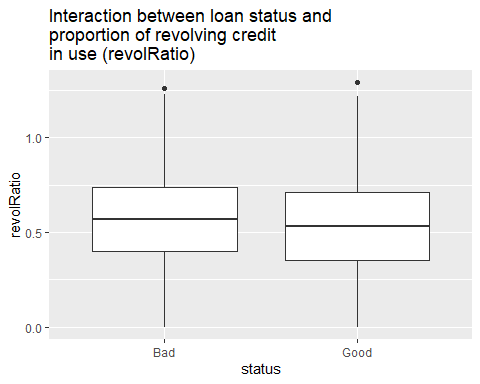


Since none of the numeric variables have any negative values, any root or log transformation can generally be used except the above mentioned caveat.



But as we see above for totalBcLim, most of the numeric variables don’t seem to have much interaction with loan status, no matter how you transform them.

Two of the only numeric variables obviously worth including in the model are debtIncRat and revolRatio, which conveniently don’t need transformation.



### The Logistic Model

Having determined how to transform and remove our variables, we can now build our logistic model. Unfortunately, not all of the variables we wanted to use in our model could allow the model to converge if log-transformed, so we cube-root-transformed the variables that were most right-skewed (“totalRevBal”, “totalBcLim”, “income”, “totalRevLim”) instead.

The intercept of this model, along with many of the predictor variables, have highly significant associations with loan status.

Of note, our grouping of states into “excellent”, “good”, “bad”, and “awful” was a success, with the “excellent” and “good” states both having significant positive (but different) associations with loan status (relative to “awful” states).

Compressing the “length” variable into “lengthgroup” also worked out well; the ‘n/a’ and ‘4-6 years’ values are significantly different from the default “0-3 years”.

Redefining “inq6mth” as a categorical variable also paid off, as values 1-2 have significantly negative associations with loan status relative to the default of 0. It is likely that this association would not have been picked up if it had been left as an integer variable.

There are many variables in this model that have insignificant associations with loan status, but remember that a lot of the variables are just levels of categorical variables. A categorical variable is still worth keeping in the model even if many of its categories are not significantly different from the default category, so long as at least one is significantly different. “reason” is a great example of this.

After regenerating the test and train sets several times to see which variables consistently had an insignificant relationship with loan status, we determined that “bcRatio” and “totalBcLim” added nothing to the model and could safely be removed.

Below are two association tables showing the predictions of the model before and after removing the unwanted variables. The initial model has an accuracy of 78.39% and the final model has an accuracy of 78.38% - almost exactly the same!

Since the final model doesn’t generalize better than the original model, it’s a bit of a tossup whether to use the first or the second model we generated. We decided to use the second.

Model 1

|  |  |  |
| --- | --- | --- |
|  | Actually Bad | Actually Good |
| Predicted Bad | 194 | 133 |
| Predicted Good | 1365 | 5240 |

Model 2 (after removing “bcRatio” and “totalBcLim”)

|  |  |  |
| --- | --- | --- |
|  | Actually Bad | Actually Good |
| Predicted Bad | 192 | 132 |
| Predicted Good | 1367 | 5241 |

### Optimizing the threshold for accuracy

With a working generalized linear model in hand, we are ready to examine the effect of classification threshold on predictive accuracy.

Classification threshold in this context means that if the model predicts a probability of the loan being “Good” that is below a certain threshold, the loan will be predicted as “Bad”. The obvious value of this threshold is 0.5, but others are also plausible.

At the default classification threshold of 0.5, the model has an overall accuracy of 78%, 98% accuracy in predicting good loans, and 12% accuracy in predicting bad loans.

We will compare the accuracy at each classification threshold to our accuracy and profit with “no model”, by which we mean the true past in which all loans in the test dataset were accepted regardless of how likely the model would have said they were to pay off. Thus, we mean “no model” as equivalent to using the model with a classification threshold of 0.

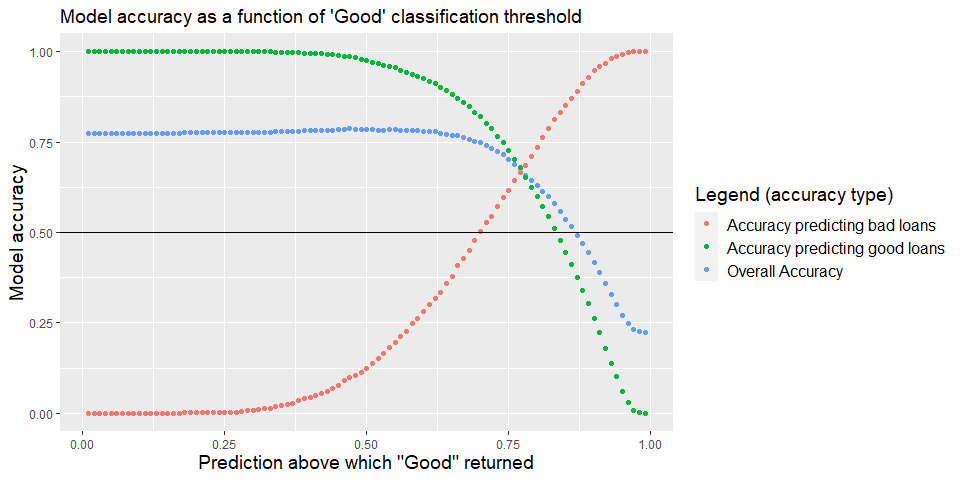
Of course, there’s another ‘no model’ scenario where you reject all loans regardless of the model’s prediction, but no bank would just reject all loan applications out of hand.

Below are plots showing the effects of classification threshold (from 0.01 to 0.99) on the model’s overall accuracy and its accuracy predicting good and bad loans when all loans are rejected for which the model’s predicted probability of paying off is below the classification threshold.

Since the overall accuracy is about 20% when all loans are flagged as ‘Bad’ and about 80% when all loans are flagged as ‘Good’, that means that 80% of all the loans were good.

A few things are worth noting here:

* Overall accuracy is blue, accuracy predicting bad loans is red, and accuracy predicting good loans is green.
* The model’s accuracy never gets much higher than the accuracy with no model. This means that it’s hard for the model to get much better at classifying bad loans without getting correspondingly worse at classifying good loans.
* When the classification threshold is about 0.8, the overall accuracy, accuracy predicting bad loans, and accuracy predicting good loans are all about 65%.

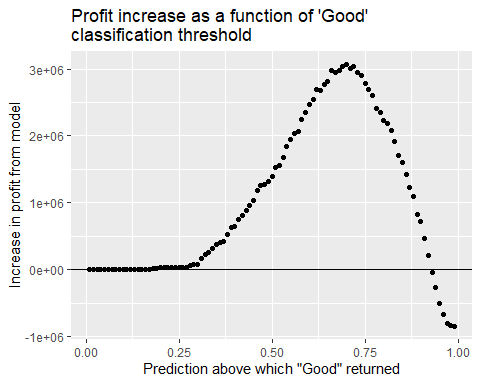


### Optimizing the model for profit

Again defining “no model” as meaning accepting all the loans in the test dataset, we compared the profit with no model to the accuracy at classification thresholds ranging from 0.01 to 0.99. At the default classification threshold of 0.5, a substantial profit increase of $1,396,995 is realized. However the maximum possible profit realized by a perfect model that rejects all bad loans and accepts all good loans would have been $12,437,143 - almost ten times as much!

Some things to note about the plot:

* The horizontal line at zero means the profit realized when all loans are accepted.
* The profit from using this model reaches a maximum of a little over $3 million when the classification threshold is 0.70, so the bank should reject loans unless they’re predicted to have at least a 70% probability of paying off.
* The profit from using this model is positive for every classification threshold from 0.18 to 0.92. This is a huge range of classification thresholds, indicating that the model makes it very easy to increase profits.
* If we compare this plot of profit to the plot of overall accuracy, we see that profit is maximized after overall accuracy starts dropping. This means that there are a lot of very bad loans that are pretty hard to classify, and you have to reject a significant percentage of good loans before you reject most of the really bad loans that greatly reduce profit.



### Results summary

In conclusion, we have developed a robust linear model with good predictive power (from 59% to 78% accuracy) and positive profit potential (from +$2,040,000 to +$3,060,000) for any classification threshold between 0.56 and 0.82. Clearly, there are many truly rotten loans out there, and this model is quite good at detecting them.

There are some other interesting take-aways from this exercise:

* Many states like Maine and Alaska are huge sucks on profit, and the bank should either stop working there or figure out why those states are bad.
* Some predictors like total credit limit of all credit cards (totalBcLim) have very little effect on the default rate, and the bank might consider stopping tracking them.

It is likely that this model would benefit from more sophisticated transformations of some of the variables, or elimination of the most stubbornly skewed variables. It’s possible that a less granular breakdown of “employment” would also reveal some useful information that could improve the accuracy of the model. Finally, we didn’t include terms for interactions between variables; that may be worth investigating.