Gil Graybill – BAN 525
Final Project – Home Equity Bad Credit Risks
Professor Cetin Ciner
June 28, 2020

Introduction

Home equity lines of credit can be a lifeline for a consumer. My first home equity line came in really handy. My fiance (now wife) and I were buying a house and I had to sell my house for the down payment. One scenario was to sell my house first, move all my stuff to her garage, and then move it back in when we bought our together house (and take a warm bath to soothe my aching bones). The second scenario was to open a home equity line on my house, buy our new house with the equity line, and then sell my house and pay off the equity line (and only move the contents once). Total interest and costs for the loan that I had for 10 days was about $100. I was happy to write that check.

Home equity lines can also be the rope that a consumer can use to hang themselves, and the bank is left with a repossession and bankruptcy procedure, and no bank wants to go through that. If a bank could figure out who is a bad risk for a home equity loan it would cut down on loan defaults. Could a customer who is a “good risk” become a “bad risk” over time, and how could there be measurable factors to determine that?

The dataset we’re using consists of 5960 rows of home equity loan customers and variables about them. They are listed below, along with the number of missing values for that column (in parenthesis):

BAD (0) - Whether the customer is a good or bad credit risk. 1 is a bad risk.
LOAN (0) - How much was the loan
MORTDUE (518) - How much they need to pay on their mortgage
VALUE (112) - Assessed valuation
REASON (252) - Reason for Loan
JOB (279) - Broad job category
YOJ (515) - Years on the Job
DEROG (708) - Number of Derogatory Reports
DELINQ (580) - Number of Deliquent Trade Lines
CLAGE (308) - Age of Oldest Trade Line
NINQ (510) - Number of recent credit inquiries.
CLNO (222) - Number of trade lines
DEBTINC (1267) - Debt to income as a percentage

In total, 2596 rows have at least one missing value, which is 43% of the data. We could choose to ignore any row with a missing column, but what if there is a chance that rows with missing data can still be analyzed? Perhaps the presence of missing data could be an indicator of a bad risk (If they're sloppy when filling out forms, maybe they're sloppy when they take care of their finances?).

Our response variable will be "BAD", which is a Boolean variable, with 1 indicating a bad credit risk. I have run the following models against the data:

(1) OLS (Ordinary Least Squares)
(2) Stepwise Forward Regression
(3) Stepwise Backward Regression
(4) Adaptive Lasso
(5) Adaptive Elastic Net
(6) Ridge Regression
(7) Boosted Tree
(8) Bootstrap Forest
(9) Neural Net with 3 nodes for TanH
(10) Neural Net with 3 nodes each for TanH, Linear, and Gaussian
(11) Neural Net with 2 layers of 3 nodes each for TanH, Linear, and Gaussian
(12) Neural Net with 3 nodes each for TanH, Linear, and Gaussian with 10 boosting models and 20 tours
(13) Neural Net with 3 nodes for TanH with 10 boosting models and 20 tours
(14) Neural Net with 3 nodes each for TanH, Linear, and Gaussian with 20 boosting models and 40 tours

Analysis and Model Comparison

I arrived at this list of models by running the data against a cross-section of models, and then choosing to refine the model tuning as the results of the models were found. Sometimes the most complex models are not the best, especially if there is a linear relationship or an "easy" equation that can be found to describe the model. While OLS (1) creates a model using all variables, Forward and Backward Regression (2, 3) eliminate those variables that don't add much to the model. I added the "Adaptive" option to Lasso and Elastic Net (4, 5) because running OLS behind the scenes first leads to a better model. Boosted Tree and Bootstrap Forest

(6,7) are Decision Tree based methodologies that can uncover non-linear relationships and possibly an easy to understand graph or decision tree for output.

For Neural Net I ran several different options varying the number of nodes, layers, and boosting. After comparing all the model results I had a champion model with Neural Network, which led me to run a few more models and changing the parameters, which got me an even better model.

For cross-validation we employed a 60-20-20 split for training-validation-test split. This gave us 3576 rows of training data, 1192 rows of validation data, and 1192 rows of test data, splitting up the data randomly.

"BAD" was chosen as the response variable and the rest of the variables were tested as dependent variables using JMP Pro 15. The calculated value for the response variable, the calculated value for "BAD" was determined and recorded and the resulting formulas determined using all models. Additionally, all of the models were run again using the "Informative Missing" option. Using this option tell JMP to analyze the rows with missing data by using the mean of the nonmissing values.

The best way to compare results for a logistical regression model is the check the "Area Under Curve", or AUC. The value consists of the ratio of "True Positives" vs. the ratio of "True Negatives", plotted against all possible cutoff values for success. The higher the AUC, the better the model is. Because we are using cross-validation, we'll be looking at the results for the Test data. The results are as follows:

Measures of Fit for BAD

Validation	Bad Risk	Creator	.2 .4 .6 .8	Entropy RSquare	Generalized RSquare	RMSE	Misclassification Rate	N	AUC
Test	OLS	Fit Nominal Logistic		0.2922	0.3576	0.2383	0.0661	681	0.8436
Test	OLS MI	Fit Nominal Logistic		0.4466	0.5739	0.2910	0.1107	1192	0.9104
Test	Stepwise FWD	Fit Ordinal Logistic		0.2767	0.3420	0.2474	0.0745	738	0.8312
Test	Stepwise FWD MI	Fit Ordinal Logistic		0.4444	0.5717	0.2912	0.1091	1192	0.9094
Test	Stepwise BKWD	Fit Ordinal Logistic		0.2831	0.3487	0.2420	0.0720	778	0.8173
Test	Stepwise BKWD MI	Fit Ordinal Logistic		0.4396	0.5668	0.2935	0.1149	1192	0.9082
Test	Lasso Adaptive	Fit Generalized Adaptive Lasso		0.2736	0.3380	0.2433	0.0681	778	0.8163
Test	Lasso Adaptive MI	Fit Generalized Adaptive Lasso		0.4462	0.5734	0.2915	0.1116	1192	0.9109
Test	Enet Adaptive MI	Fit Generalized Adaptive Elastic Net		0.4462	0.5735	0.2915	0.1116	1192	0.9109
Test	Enet Adaptive	Fit Generalized Adaptive Elastic Net		0.2736	0.3379	0.2433	0.0681	778	0.8163
Test	Ridge	Fit Generalized Ridge		0.4457	0.5730	0.2914	0.1107	1192	0.9104
Test	Ridge MI	Fit Generalized Ridge		0.4457	0.5730	0.2914	0.1107	1192	0.9104
Test	Boosted	Boosted Tree		0.4030	0.5288	0.3024	0.1250	1192	0.9230
Test	Bootstrap Forest MI	Bootstrap Forest		0.5200	0.6453	0.2714	0.0956	1192	0.9439
Test	Bootstrap Forest	Bootstrap Forest		0.3761	0.4999	0.3145	0.1359	1192	0.9158
Test	NN 3x1 2	Neural		0.4062	0.4806	0.2180	0.0529	681	0.8809
Test	NN 3x1 MI	Neural		0.4778	0.6049	0.2813	0.1023	1192	0.9206
Test	NN 3x3x3x1	Neural		0.4066	0.4810	0.2140	0.0485	681	0.8690
Test	NN 3x3x3x1 MI	Neural		0.4557	0.5830	0.2900	0.1158	1192	0.9143
Test	NN 3x3x3x3x3x3	Neural		0.4248	0.4999	0.2102	0.0543	681	0.8772
Test	NN 3x3x3x3x3x3 MI	Neural		0.4229	0.5497	0.2970	0.1191	1192	0.9022
Test	NN 3x3x3x10x20	Neural		0.4494	0.5251	0.2079	0.0543	681	0.8960
Test	NN 3x3x3x10x20 MI	Neural		0.5856	0.7048	0.2527	0.0864	1192	0.9515
Test	NN 3x10x20	Neural		0.4397	0.5152	0.2091	0.0529	681	0.8898
Test	NN 3x10x20 MI	Neural		0.5498	0.6729	0.2582	0.0864	1192	0.9379
Test	NN 3x3x3x20x40 MI	Neural		0.5953	0.7132	0.2486	0.0839	1192	0.9525
Test	NN 3x3x3x20x40	Neural		0.4635	0.5393	0.2027	0.0485	681	0.8932

It looks like the champion model is Neural Net with 3 nodes each for TanH, Linear, and Gaussian with 20 boosting models and 40 tours using the “Information Missing” option, with an AUC of 0.9525. From looking at these results, it seems like the “Missing Information” models are overall performing better than the models that are just including the rows with complete information. To be sure, let’s break it out.

Here is the model comparison for the complete rows:

Measures of Fit for BAD

Validation	Bad Risk	Creator	.2 .4 .6 .8	Entropy RSquare	Generalized RSquare	RMSE	Misclassification Rate	N	AUC
Test	OLS	Fit Nominal Logistic		0.2922	0.3576	0.2383	0.0661	681	0.8436
Test	Stepwise FWD	Fit Ordinal Logistic		0.2767	0.3420	0.2474	0.0745	738	0.8312
Test	Stepwise BKWD	Fit Ordinal Logistic		0.2831	0.3487	0.2420	0.0720	778	0.8173
Test	Lasso Adaptive	Fit Generalized Adaptive Lasso		0.2736	0.3380	0.2433	0.0681	778	0.8163
Test	Enet Adaptive	Fit Generalized Adaptive Elastic Net		0.2736	0.3379	0.2433	0.0681	778	0.8163
Test	Ridge 2	Fit Generalized Ridge		0.2763	0.3396	0.2443	0.0734	681	0.8522
Test	NN 3x1 2	Neural		0.4062	0.4806	0.2180	0.0529	681	0.8809
Test	NN 3x3x3x1	Neural		0.4066	0.4810	0.2140	0.0485	681	0.8690
Test	NN 3x3x3x3x3x3	Neural		0.4248	0.4999	0.2102	0.0543	681	0.8772
Test	NN 3x3x3x10x20	Neural		0.4494	0.5251	0.2079	0.0543	681	0.8960
Test	NN 3x10x20	Neural		0.4397	0.5152	0.2091	0.0529	681	0.8898
Test	NN 3x3x3x20x40	Neural		0.4635	0.5393	0.2027	0.0485	681	0.8932

All of the Neural Net models are within 2.4% of each other for AUC, with the champion model being “Neural Net with 3 nodes each for TanH, Linear, and Gaussian with 20 boosting models and 40 tours” with an AUC of 0.8932.

Here is the model comparison for all of the rows, including those with missing data:

Measures of Fit for BAD

Validation	Bad Risk	Creator	.2 .4 .6 .8	Entropy RSquare	Generalized RSquare	RMSE	Misclassification Rate	N	AUC
Test	OLS MI	Fit Nominal Logistic		0.4466	0.5739	0.2910	0.1107	1192	0.9104
Test	Stepwise FWD MI	Fit Ordinal Logistic		0.4444	0.5717	0.2912	0.1091	1192	0.9094
Test	Stepwise BKWD MI	Fit Ordinal Logistic		0.4396	0.5668	0.2935	0.1149	1192	0.9082
Test	Lasso Adaptive MI	Fit Generalized Adaptive Lasso		0.4462	0.5734	0.2915	0.1116	1192	0.9109
Test	Enet Adaptive MI	Fit Generalized Adaptive Elastic Net		0.4462	0.5735	0.2915	0.1116	1192	0.9109
Test	Ridge MI	Fit Generalized Ridge		0.4457	0.5730	0.2914	0.1107	1192	0.9104
Test	Bootstrap Forest MI	Bootstrap Forest		0.5200	0.6453	0.2714	0.0956	1192	0.9439
Test	NN 3x1 MI	Neural		0.4778	0.6049	0.2813	0.1023	1192	0.9206
Test	NN 3x3x3x1 MI	Neural		0.4557	0.5830	0.2900	0.1158	1192	0.9143
Test	NN 3x3x3x3x3x3 MI	Neural		0.4229	0.5497	0.2970	0.1191	1192	0.9022
Test	NN 3x3x3x10x20 MI	Neural		0.5856	0.7048	0.2527	0.0864	1192	0.9515
Test	NN 3x10x20 MI	Neural		0.5498	0.6729	0.2582	0.0864	1192	0.9379
Test	NN 3x3x3x20x40 MI	Neural		0.5953	0.7132	0.2486	0.0839	1192	0.9525

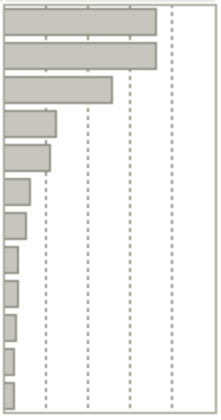
It is interesting to note that <u>all</u> of the models that include all rows have a higher AUC than <u>any</u> of the models that use only complete rows.

Interpretation

It would be easy to only analyze rows of observations that have complete data and be able to justify that decision (“The data is incomplete, so you’re not representing the data properly by making stuff up!”). But JMP’s algorithm for Missing Information gave us a better model every time for this data. There’s obviously enough meaningful data there that JMP can glean more insight with more rows, even if some of the data is missing. The extra 400-500 rows of incomplete data helps the Neural Network make more connections and create a better model. But is our champion model different depending if we use Missing information or not?

When running variable importance with these variables against our champion model with <u>all rows included</u>, we see the following:

Column	Main Effect	Total Effect
DELINQ	0.058	0.732
DEBTINC	0.048	0.728
DEROG	0.03	0.515
NINQ	0.012	0.25
JOB	0.016	0.221
CLAGE	0.008	0.132
VALUE	0.008	0.111
MORTDUE	0.005	0.074
YOJ	0.005	0.068
LOAN	0.004	0.062
CLNO	0.003	0.053
REASON	0.005	0.051

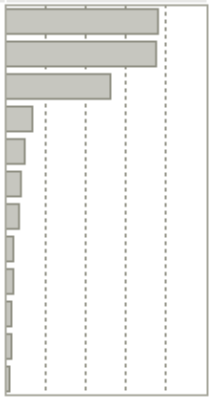


DELINQ and DEBTINC run very close with 73.2% and 72.8% effect on the total, followed by DEROG with a respectable 51.5%. 4 more variables have a total effect between 11% and 25%, and all variables have at least a 5% total effect on the model. While 5% is not a lot, it still shows that all variables have a contributory effect. This shows non-linear relationships which is why the non-linear models performed well with this data and the linear ones did not.

When running variable importance with these variables against our champion model with only complete rows, we see the following:

Overall

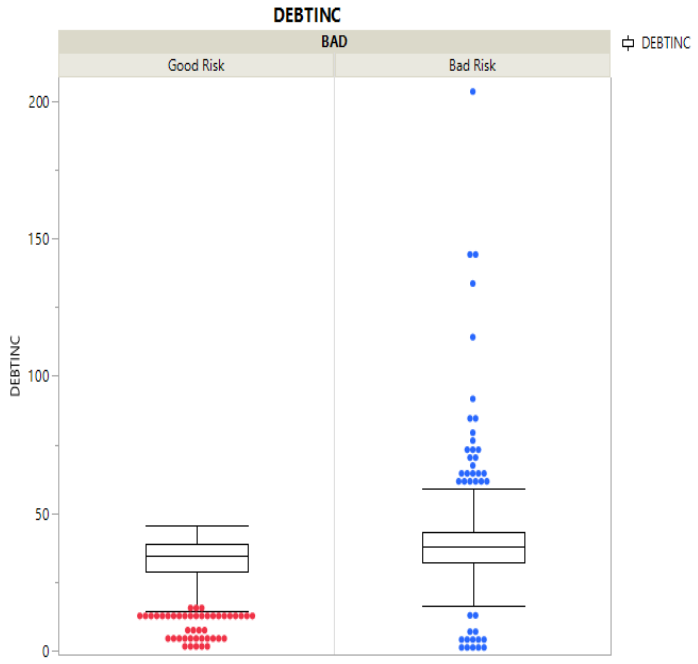
Column	Main Effect	Total Effect
DEBTINC	0.076	0.768
DELINQ	0.076	0.754
DEROG	0.03	0.527
NINQ	0.009	0.141
CLAGE	0.008	0.097
VALUE	0.006	0.079
JOB	0.006	0.068
CLNO	0.004	0.043
LOAN	0.005	0.038
MORTDUE	0.003	0.036
YOJ	0.004	0.032
REASON	0.002	0.02



The top three variables (DEBTINC, DELINQ, and DEROG) have very similar effects, with DEBTINC and DELINQ having about 3% more effect. However, we notice that only two variables fall into the 10-25% range (NINQ and rounding up CLAGE), with 5 variables falling below the 5% range. This indicates that analyzing the Missing Information for the Neural Network model really helps find relationships for the variables of secondary importance.

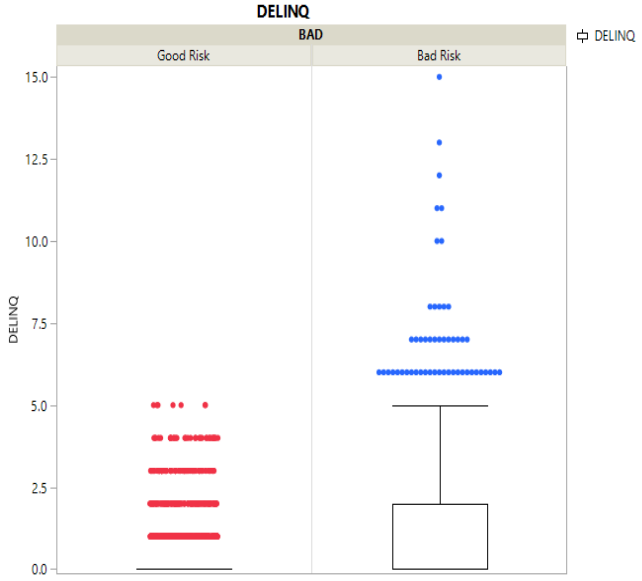
From a real-world standpoint, what does our champion model tell us?

DEBTINC is “debt to income, as a percentage”. It makes sense that the higher the percentage of debt a homeowner has, the more likely they are to become a bad risk for paying off their home equity loan. Anybody with a mortgage has debt, so totally eliminating it is not an option. Financial advisors recommend that a mortgage payment take up no more than 28% of income, with another 8% for other debt lines, making 36% the benchmark



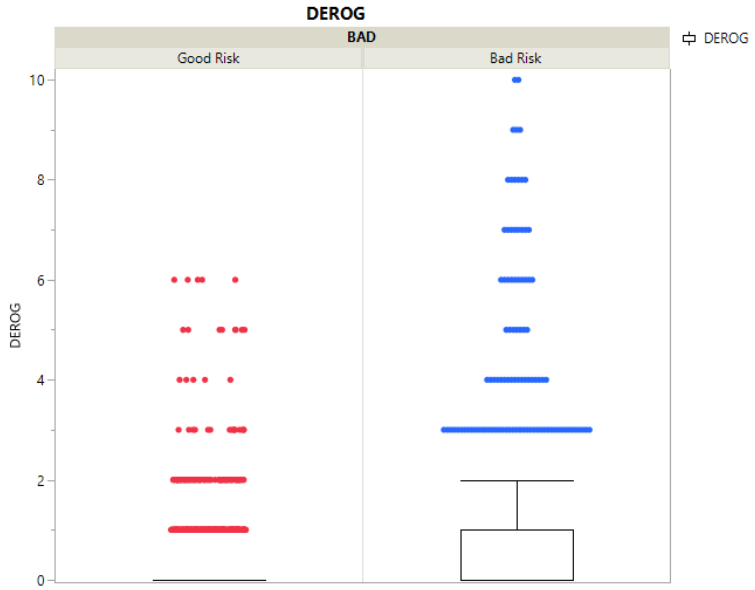
(https://www.investopedia.com/ask/answers/081214/whats-considered-be-good-debttoincome-dti-ratio.asp) . It's easy for a homeowner to fall into a trap and take on more debt, baited with seemingly low interest rates and possibly a feeling of "I'll never get out of debt anyhow, so I might as well live it up!". From our sample, 90% of our customers have a DEBTINC of 41.4% or less. Any customer with a DEBTINC of 60 or more was classified as a BAD risk (including the one with a value of 203%! Yikes!). Perhaps this has been a hard cut-off that was used in the past, and the model backs up using criteria like that.

DELINQ is the number of delinquent trade lines. A "Trade line" is an individual



borrowing account for which a customer has been approved for (such as a car loan, mortgage, credit cards, store credit cards, etc.), and delinquent one is an account where a payment has been missed (https://www.investopedia.com/terms/t/trade-line.asp). Note that CLNO, the number of tradelines, is a variable in this dataset, but that does NOT have a large effect on credit risk. If a customer has been delinquent on many of their tradelines they are a bad risk. From the data, anybody with a DELINQ above 6 is considered a bad risk. Perhaps this has been a hard cut-off that was used in the past, and the model again backs up using a criteria like that.

DEROG is the number of derogatory reports, and it indicates serious and frequent delinquency for a tradeline (https://www.investopedia.com/terms/t/trade-line.asp). While even a good risk can be delinquent a few times (and it looks like 5 is the cutoff from that from the previous DELINQ graph), a small number of derogatory reports can indicate a bad risk. From looking at the data, it appears that values from 3 to 6 are possibly bad risks, with anything above 6 considered a bad risk.



If we stopped there, we would just be using the three most effective variables and not using the power of analyzing the Missing Information. Note that the AUC for complete rows for our champion model was 0.8932, while it was 0.9525 for those with missing information. Our data had 4771 customers who were "Good Risk", and 1189 who were "Bad Risk", and the "BAD" ones are the ones that cost the bank money. We want to find as many of those as we can. Even though NINQ (Number of recent credit inquiries: 25%), JOB (Broad Job Category, 22%), CLAGE (Age of oldest Trade line, 13%), and VALUE (Assessed Valuation, 11%) aren't big effectors, utilizing them can help increase the number of bad risks that can be found with the model that used the missing information.

Complete Rows

Test

BAD

Measures	Value
Generalized RSquare	0.2960368
Entropy RSquare	0.2381074
RMSE	0.2435651
Mean Abs Dev	0.1046464
Misclassification Rate	0.071953
-LogLikelihood	158.21952
Sum Freq	681

Confusion Matrix

Actual BAD	Predicted Count Good Risk	Bad Risk
Good Risk	606	13
Bad Risk	36	26

Confusion Rates

Actual BAD	Predicted Rate Good Risk	Bad Risk
Good Risk	0.979	0.021
Bad Risk	0.581	0.419

Missing Information

Test

BAD

Measures	Value
Generalized RSquare	0.6471385
Entropy RSquare	0.5218918
RMSE	0.2622936
Mean Abs Dev	0.1288764
Misclassification Rate	0.0964765
-LogLikelihood	294.59548
Sum Freq	1192

Confusion Matrix

Actual BAD	Predicted Count Good Risk	Bad Risk
Good Risk	902	37
Bad Risk	78	175

Confusion Rates

Actual BAD	Predicted Rate Good Risk	Bad Risk
Good Risk	0.961	0.039
Bad Risk	0.308	0.692

As we can see from the confusion matrices for the models, the rate of finding bad risks is 41% for Complete Rows, but 69% for Missing Information. That's a lot more bad risks that are being detected.