Credit Scorecard – a Classification Problem

Prepared by: Zhe Rao

The dataset contains 50,000 applicants information and 53 variables. This report aims to explore models that could utilize the applicants' information to predict the probability of default on a loan if it is given.

Preprocessing

The first step is to eliminate variables that don't contribute to help me distinguishing between good and bad customers. Specifically, variables with constant value across all customers should be dropped because good and bad customers will be treated as equivalence in those variables which is not desirable. As such, I used a *for* loop to compile an array containing those unhelpful variables (indicated below). I then dropped those variables and store the resulting dataset in a data frame called "df2".

```
1   CLERK_TYPE
4   QUANT_ADDITIONAL_CARDS
9   EDUCATION_LEVEL
20   FLAG_MOBILE_PHONE
44   FLAG_HOME_ADDRESS_DOCUMENT
45   FLAG_RG
46   FLAG_CPF
47   FLAG_INCOME_PROOF
49   FLAG_ACSP_RECORD
```

Secondly, I aimed to eliminate missing values and apply appropriate treatment. Firstly, I complied two arrays of 1) variables with null values and 2) how many null values they have. In addition, since an empty string is also a missing value, I compiled two arrays of those instances as well. To give an overview of what the missing values are situated, I added up the null values counts with the empty string values counts. The result is as follows,

```
STATE_OF_BIRTH                  2064
CITY_OF_BIRTH                   2064
RESIDENCIAL_BOROUGH             10
RESIDENCIAL_PHONE_AREA_CODE     8212
RESIDENCE_TYPE                  1349
MONTHS_IN_RESIDENCE             3777
PROFESSIONAL_STATE              34307
PROFESSIONAL_CITY               34114
PROFESSIONAL_BOROUGH            34713
PROFESSIONAL_PHONE_AREA_CODE    36532
PROFESSION_CODE                 7756
OCCUPATION_TYPE                 7313
MATE_PROFESSION_CODE            28884
MATE_EDUCATION_LEVEL            32338
```

Before applying treatment, I checked if any record (i.e. row) contains more than 50% of null values in which case the record should be deleted. I failed to find any such record.

Starting the treatment process, I first deleted the variables with more than 50% (i.e. 25,000 counts) that are certainly not informative. In addition, I replaced the missing values in variables that represent a category. For example, I replaced the missing values in "OCCUPATION_TPYE" as 6 because I believe that they could represent another category not enlisted. However, since the encoding of the categories is not available, it is more sensible to call operation department and double check. For "RESIDENCIAL_TYPE", it contains category "0" that should not exist, so I first replaced the "0"s by the mode "1" before treating the missing values. Note here, I should have split the dataset first and replace by the mode of training set, but the "1"s is so dominant in the case that it would still be the mode even after taking 30% of the "1"s out.

Before replacing the missing values by the mode/median, I created 3 new variables called: 1) "SUPPORT_POWER" which is total income divided by (number of dependents + 1), I believe the larger the ratio, the more capable the applicant is to pay back the loan because less people he/she has to support; 2) "ACCOMPLISHMENT" which is income divided by age, the larger the ratio, the larger the earning ability to the applicant; and 3) "DEBT" which is sum of flags for credit cards, the larger the number, the more indebted the applicant is

I also noticed that there are some people with a huge income that could distort the behavior of the overall population. So I decided to only include people with total income less than 4,000 which reduced 457 records.

Now, I split the data into 70% in training set and 30% in testing sets. For missing value treatment, I replaced the records with median/mode of training dataset to avoid leaking (replaced around 5% each). For outliers, I replaced the "AGE" by the median of training set for ages less than 18 because they cannot apply for a loan.

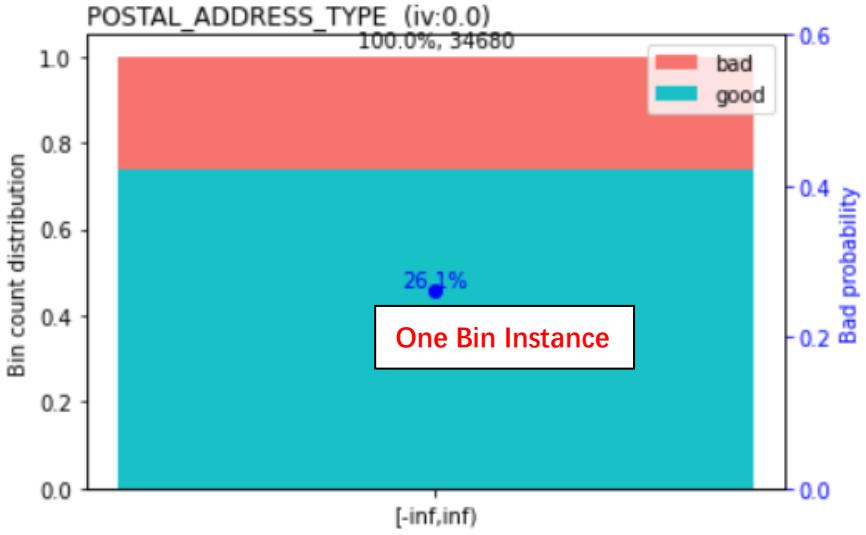
Lastly, I formatted the cities, boroughs in the dataset so that they are in upper case without white spaces.

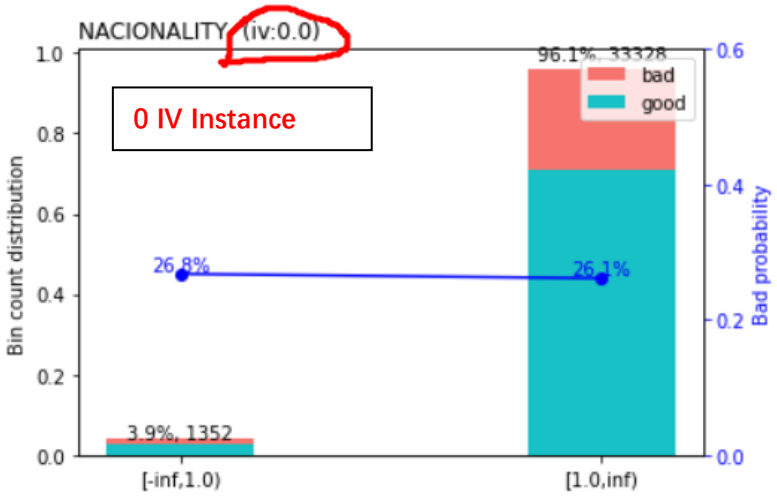
Now, the dataset is clean and available to be trained using models

WoE

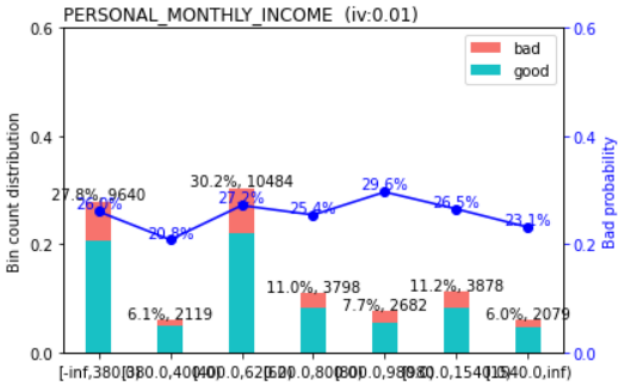
Due to large number of categories in variables associated with cities, boroughs, and ZIP codes. I combined the cases with total count less than 50 in the dataset by a category called "OTHERS". Then I split the data into training and testing sets again with the same seed (which would produce the same split as the first split).

For binning, I initially tried 100 cuts because of the small dataset; minimum 500 cases (i.e. 1% of the entire dataset) in each bin because I want the categories to be representative; and maximum 10 bins. From the resulting plot, I eliminated the variables with close-to-zero information values (<0.01) and the variables with only one bin due to their inability to help me distinguish the good customers from the bad ones (i.e. they provide 0 information!). An example for each instance is included as follows

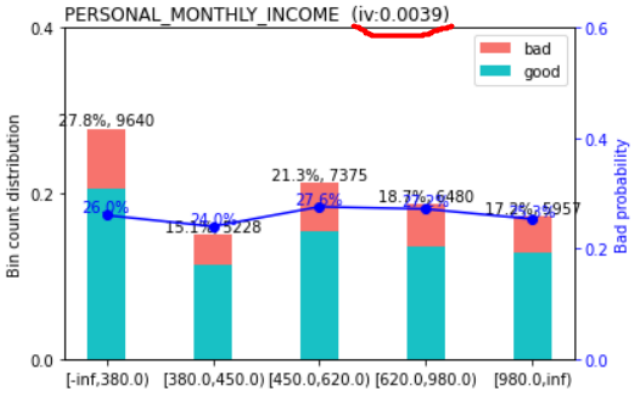




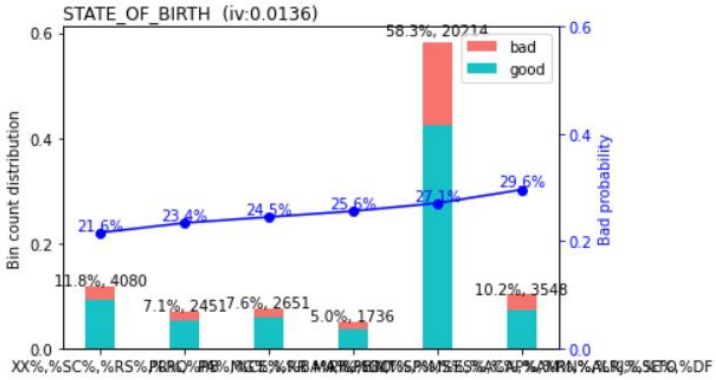
It is noticeable that the monthly income variable presented an unexplainable trend indicated below



After adjusting bins to a better trend, the information value dropped below 0.01 (indicated below). As such, the variable is eliminated



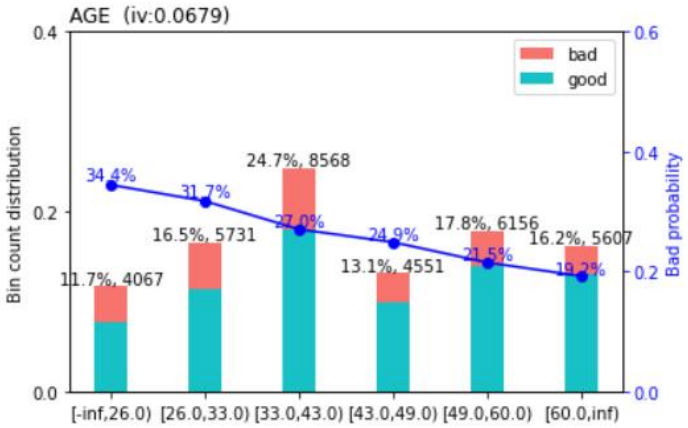
In addition, the “STAT_OF_BIRTH” variable is too concentrated in one category as indicated below. Since the variable is too clustered for me to manually adjust the bins, I decided to exclude this variable.



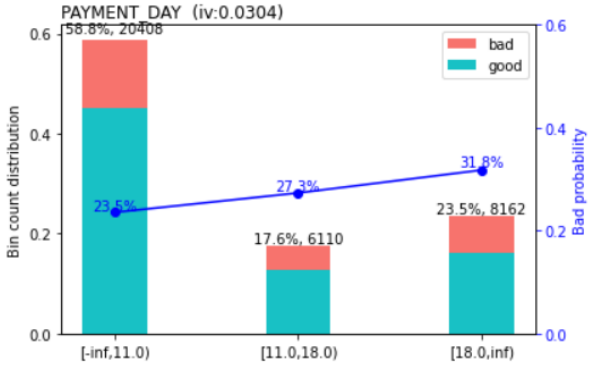
After eliminating the uninformative variables. I used a decision tree process to create the bins again with minimum 2,500 cases (i.e. 5%) to make behavior in each bin more significant, and maximum 7 bins.

After the program created the optimal binning, I manually adjusted some variables to help make sense of the trend presented. Note that, since the encodings for categorical variables are not available, I should first contact with operation department before I adjust the bins. In this case, I would only adjust the bins for non-categorical variables.

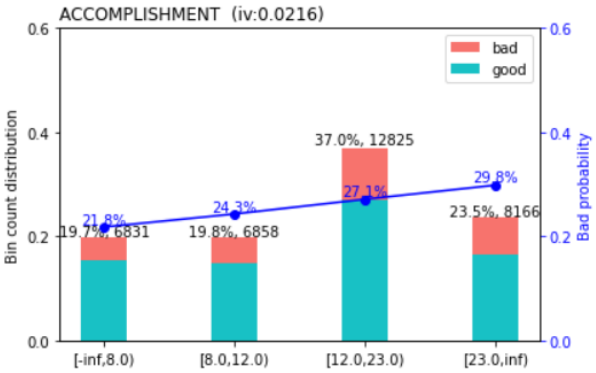
Firstly, for age, the general trend is that the younger the age, the more likely to default. I adjusted the bins to the following to represent this trend



For “PAYMENT_DAY”, it would make sense that people who pays back the loan at the beginning of the month have less probability of default because their payments are better managed. As such, I adjusted the bins to the following



For “ACCOMPLISHMENT”, it would make sense that higher the ratio, higher the ability to repay and better managed those applicants are. Accordingly, the bins are adjusted to represent this trend

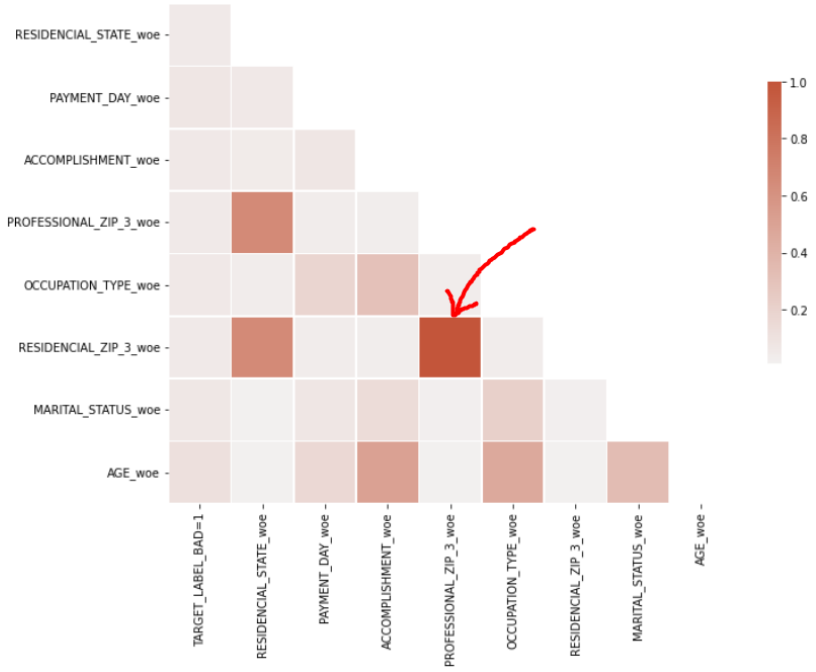


After the adjusted binning, I obtained the following information values which is a representation of the variable importance

	variable	info_value
5	AGE_woe	0.067863
6	PAYMENT_DAY_woe	0.030390
7	MARITAL_STATUS_woe	0.027959
1	ACCOMPLISHMENT_woe	0.021630
2	OCCUPATION_TYPE_woe	0.018840
0	RESIDENCIAL_STATE_woe	0.016862
3	PROFESSIONAL_ZIP_3_woe	0.015906
4	RESIDENCIAL_ZIP_3_woe	0.015906

Correlation

After calculating and plotting the correlation between the 8 variables, I discovered an extremely high correlation between the professional zip code and residential zip code (as indicated below). Although elastic net algorithm is able to deal with correlations by using some ridge regression, this correlation is too high so that I decided the remove residential zip code because it is also highly correlated with residential state.



Scorecard

Now I can train a model with 7 variables I selected using logistic regression with training data. I used elastic net to make the model more flexible, utilizing both ridge and LASSO regression. I included an array from 10 to 50 as inputs for "Cs" to have a range of penalties to try. Note here, the numbers are large because I don't want large penalties (=1 / C) to regularize away all my parameters. I used 3-fold cross validation given the size of the train set is small. I used the "saga" solver to take advantage of elastic net. I included a range of ratios weighting LASSO over ridge to find an optimal ratio.

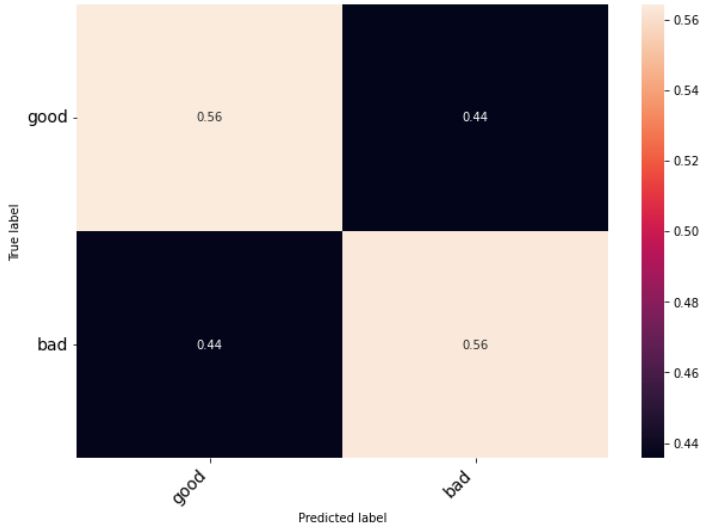
After training the model, it sets all the weights to ridge (probably due to the correlation between variables), and used 10 as the penalty constant. I obtained the following coefficients

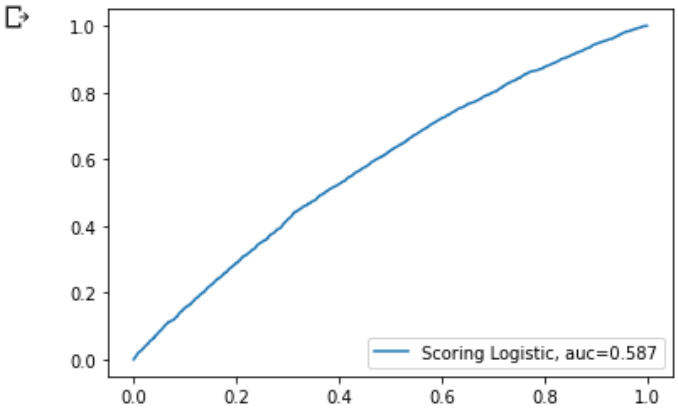
	columns	0
0	RESIDENCIAL_STATE_woe	0.576933
1	PAYMENT_DAY_woe	0.750327
2	ACCOMPLISHMENT_woe	0.216293
3	PROFESSIONAL_ZIP_3_woe	0.590490
4	OCCUPATION_TYPE_woe	-0.054264
5	MARITAL_STATUS_woe	0.514044
6	AGE_woe	0.765557

Notice that the coefficient of occupation type is negative, indicating the correlation with other variables. After refitting the model without the correlated variable, the predictive performance on the test set is not affected. Therefore, I choose to use the smaller model. And I obtained the following coefficients (they represent the variable importance)

	columns	0
0	RESIDENCIAL_STATE_woe	0.576052
1	PAYMENT_DAY_woe	0.745908
2	ACCOMPLISHMENT_woe	0.211065
3	PROFESSIONAL_ZIP_3_woe	0.589237
4	MARITAL_STATUS_woe	0.511594
5	AGE_woe	0.755076

The predictive power of the model seems relatively undesirable with accuracy of 58.7%. The AUC curve and confusion matrix are included as follows. I recommend using the 6 variables indicated above.





After building the logistic model, I can convert the predicted relative probabilities of default of applicants to a scoring system for better representation of the information. I chose a base point of 800 with odds of 20:1, and 200 points to double the odds. This would produce a nice range of scores that makes the scorecard flexible enough to be applied. The range of scores for my dataset is as follows.

	score
count	49543.000000
mean	344.528834
std	97.520239
min	74.000000
25%	275.000000
50%	341.000000
75%	415.000000
max	629.000000

Cut-off Strategy

With the score system, I can use different cutoff scores to adjust the acceptance decision of the applicants based on the bank's business strategy. If I raise the cutoff score, then I accept lower portion of applicants because only applicants with higher score than the increased cutoff is accepted. This way, I reduce the cost of accepting bad customers, but also reduce the revenue of accepting good customers. Alternatively, if I lower the cutoff score, then I predicted more good customers because any score higher than the lowered cutoff is accepted. This way, I increase the revenue of accepting more good customers, but also increase the cost of accepting more bad customers. The exact cutoff-point-table is demonstrated as below

CutOff	Accepted %	Accuracies %		
		Good	Defaulter	Total
98	99.94%	99.96%	0.12%	73.86%
148	98.52%	98.84%	2.00%	73.52%
198	93.54%	94.62%	9.39%	72.34%
248	82.16%	84.36%	23.84%	68.54%
298	65.39%	68.70%	43.67%	62.16%
348	47.52%	50.96%	61.54%	53.73%
360	43.46%	46.92%	65.73%	51.84%
372	38.80%	42.12%	70.11%	49.44%
384	34.71%	37.93%	73.96%	47.35%
398	30.35%	33.28%	77.82%	44.92%
448	15.75%	17.67%	89.48%	36.44%
498	5.51%	6.20%	96.29%	29.76%

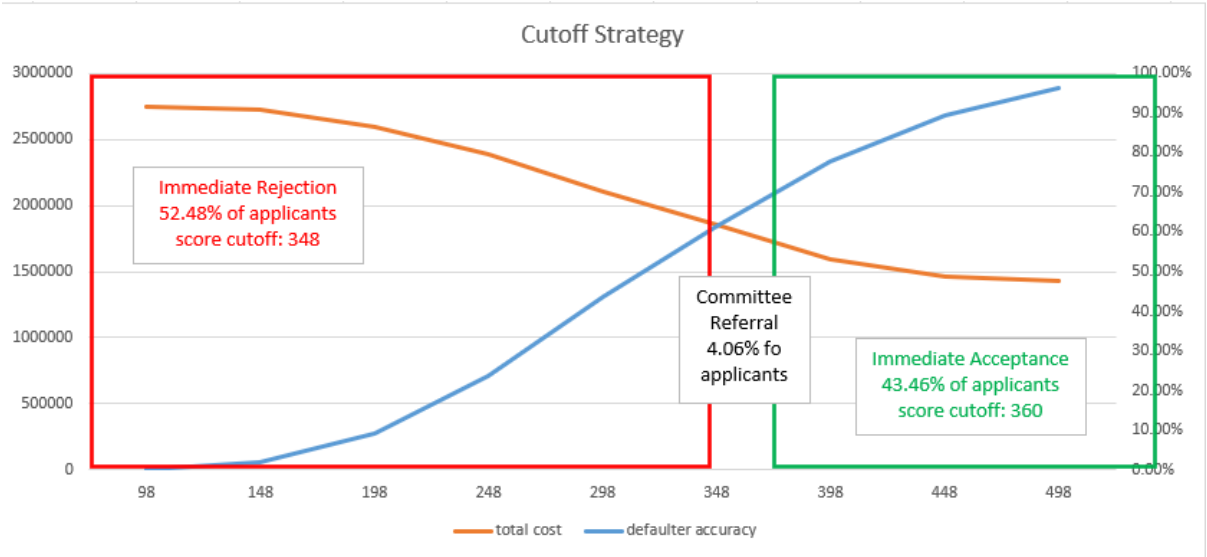
CutOff	Average Cost R$		Total Cost		
	Good	Defaulter	Good	Defaulter	Total
98	$ 43.49	$ 212.35	$ 608.86	$ 2,746,959.60	$ 2,747,568.46
148	$ 43.61	$ 213.31	$ 18,534.25	$ 2,707,330.52	$ 2,725,864.77
198	$ 43.88	$ 213.72	$ 86,311.96	$ 2,508,004.20	$ 2,594,316.16
248	$ 44.52	$ 216.21	$ 254,832.48	$ 2,132,479.23	$ 2,387,311.71
298	$ 45.11	$ 217.86	$ 515,336.64	$ 1,589,288.70	$ 2,104,625.34
348	$ 44.29	$ 211.83	$ 794,739.76	$ 1,055,125.23	$ 1,849,864.99
360	$ 43.89	$ 208.03	$ 852,519.36	$ 923,237.14	$ 1,775,756.50
372	$ 43.90	$ 206.65	$ 929,758.10	$ 799,942.15	$ 1,729,700.25
384	$ 43.55	$ 202.92	$ 989,151.15	$ 684,246.24	$ 1,673,397.39
398	$ 42.21	$ 195.63	$ 1,030,514.94	$ 562,044.99	$ 1,592,559.93
448	$ 40.20	$ 184.73	$ 1,211,025.00	$ 251,786.99	$ 1,462,811.99
498	$ 39.22	$ 173.97	$ 1,346,069.62	$ 83,505.60	$ 1,429,575.22

In the table, the calculation are as follows

$$a = \#of\ accppted\ customers$$

$$c = total\ \#of\ good\ customers$$
$$d = total\ \#of\ bad\ customers$$
$$e = \#of\ accpeted\ good\ customers$$
$$f = \#of\ rejected\ bad\ customers$$
$$g = revenue\ for\ accepted\ good\ customers$$
$$h = cost\ of\ accepted\ bad\ customers$$
$$accepted\ \% = \frac{a}{b}$$
$$accuracy\ (good) = \frac{e}{c}$$
$$accuracy(bad) = \frac{f}{d}$$
$$accuracy(total) = \frac{e+f}{c+d}$$
$$average\ cost(good) = \frac{g}{e} = \frac{\sum income_{aceepted\ good\ customers} * 0.32 * 0.2}{e}$$
$$= average\ opportunity\ cost\ of\ rejected\ good\ customers$$
$$average\ cost(bad) = \frac{h}{f} = \frac{\sum income_{accepted\ bad\ cutomers} * 0.32}{f}$$
$$total\ cost(good) = average\ cost(good) * rejected\ good\ customer$$
$$total\ cost(bad) = average\ cost(bad) * accpeted\ bad\ custtomer$$

Here is the cutoff strategy that I designed

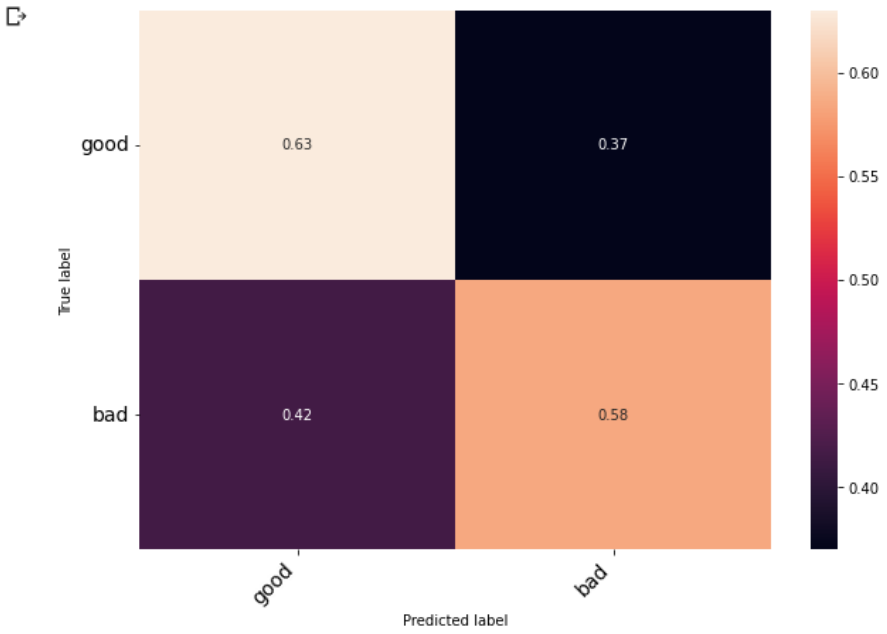


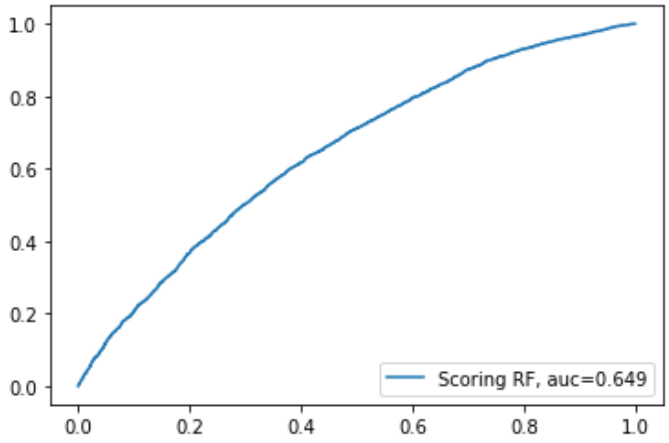
My goal is to sustain losses in order to attract customers. So I am not trying to minimize cost. Since my model doesn't know how to distinguish goods and bads well at score of 360 (51% accuracy), my committee referral should be focused on that region. Since I wish to attract customers, I would reduce my immediate rejection cutoff to 348 so applicants with score between 348 and 360 can be accessed by the committee. Those customers account for 4.06%, which is assumed to be within the capacity of the committee.

Random Forest and XG Boosting

I compared the logistic model with a random forest model trained over the cleaned dataset (i.e. “df2”) with dummy coding. For the regressor, I set 1000 as the number of trees considering the small size of the dataset. For the same reason, I chose to implement bootstrap. Minimum samples leaf was set to 0.0001 (i.e. 5 cases) so that the program doesn’t take forever to run. Minimum impurity decrease was set to 0.0001 so that the program stops when the entropy gain is less than 0.0001. I want to start from scratch, so set warm start to false.

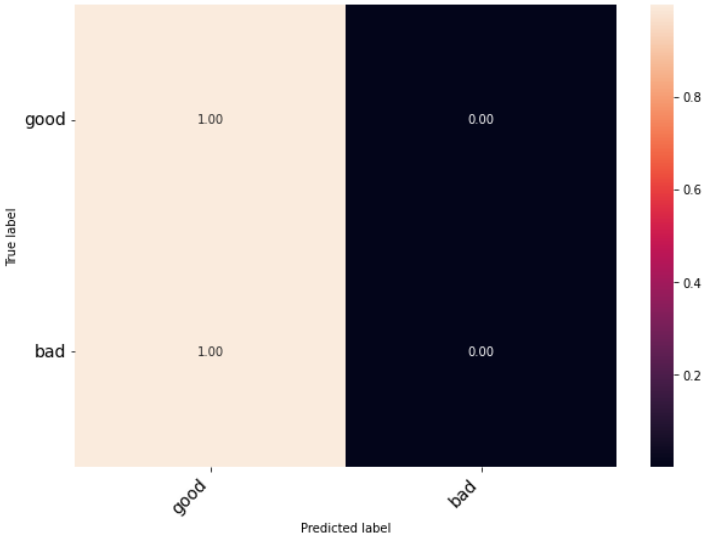
After applying the model on the test set, the results are as follows. It is performing better than logistic regression with higher overall accuracy (0.649) and better predictive power in both good and bad customers

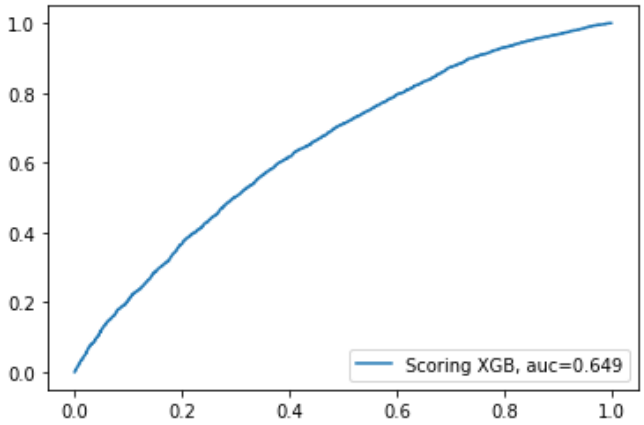




For XGBoosting, I first tuned the parameters using 5-fold cross validation. After trying 64 models with different combination of number of trees, max depths, and learning rates, I found the optimal combination with max depth of each tree set to 2, learning rate of 0.1, and 100 trees. In addition, subsample size was set to 0.632 as it is the possibility of record being selected at infinity. Three method was set to gpu_hist to utilize the computation power of GPU. Since histograms don’t require precision, GPU is perfect for the job.

The results are as follows





Although the accuracy is comparable to random forest, 64.9%, XGBoosting predicted almost all applicants as good customers. This is terrible because the accuracy is due to the imbalance of the classes instead of the predictive power of the model. I tried to lower the base_score parameter, which should in theory increase the predictions for bad customers. However, this did not affect the model. This is strange because XGBoosting should outperform random forest with small datasets.

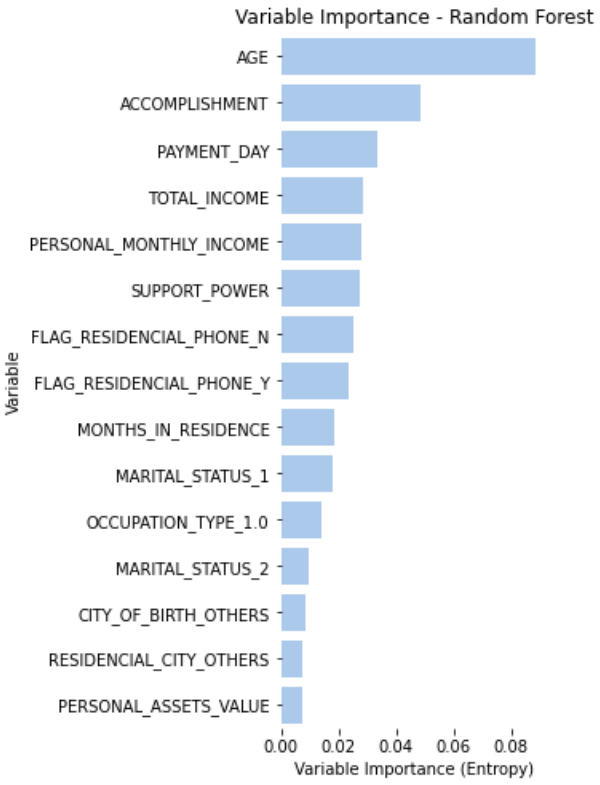
In summary, I would choose random forest over the other two models

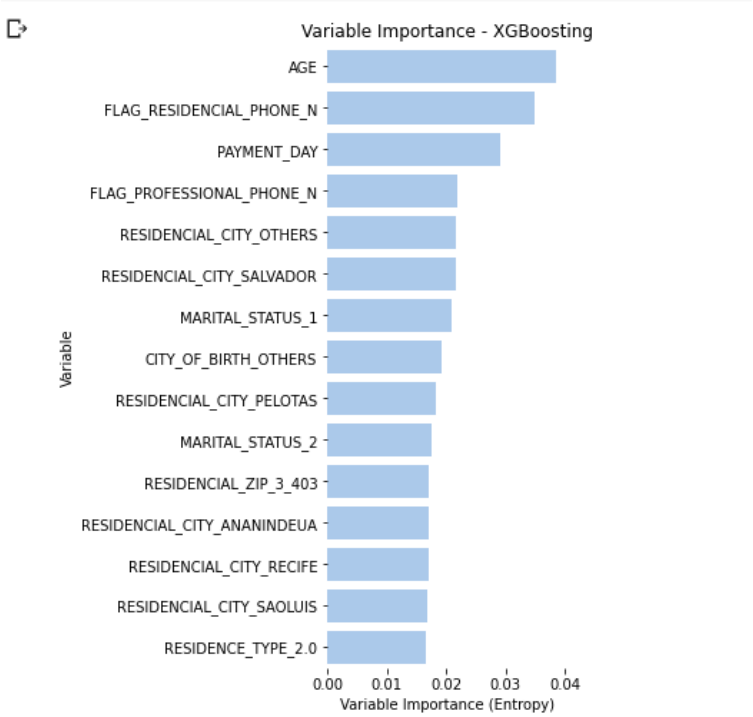
Variable Importance

The variable importance can be assessed using information value for logistic model and entropy gain for tree-based methods. The variable importance (or the most important ones) is indicated as follows

	variable	info_value
5	AGE_woe	0.067863
0	PAYMENT_DAY_woe	0.030390
3	MARITAL_STATUS_woe	0.027959
1	ACCOMPLISHMENT_woe	0.021630
4	RESIDENCIAL_STATE_woe	0.016862
2	PROFESSIONAL_ZIP_3_woe	0.015906

Variable importance for logistic model





Notice that some variables showed up in all three tables (e.g. age), some do not (e.g. income). Also their values are different. I believe that the discrepancy is due to the difference in variable selection process in each model. In logistic regression, I manually determined which variables and what levels for each variable would be included in the model based on their independent information value. In random forest, the algorithm randomly selected a subsample of variables for each tree and the levels to include determined by the entropy gain that cut could bring. Note that this is not independent because this entropy gain is conditioned on the previously selected variables. In XGBoosting, the algorithm would try to pick the variables and levels that attempts to "fix" the errors in the previous tree. Therefore, due to the different processes in selecting the variables and their levels, their importance could be different in different models. However, for important predictors, their contribution would be correctly presented.

Appendices

Link to my code for the analysis part
https://colab.research.google.com/drive/1COIxugZ2ioDeGUgX4qoMj4Kq222S7dct?usp=sharing