Project Report: Granting Loans

Wenbo

Jan 2018

1 Background

In this project, I have access to a specific bank loan data. The data contains all loans asked to the bank, whether the bank decided to grant it and, finally whether the borrower managed to repay it. On the features side, it contains information regarding the loan itself and its borrower such as application date, borrowers checking account balance, saving account balance and other related information. I am asked to finish the following tasks:

- Build a model which is better than the current bank model. The scoring criterion is
  - If our model grant the loan but it doesn't get repaid, we lose 1 point
  - If our model grant the loan and it gets repaid, we gain 1
  - If our model doesn't grant the loan, we gain 0
- Describe the impact of the most important variables on the prediction. Investigate the impact of variable "is employed".

2 Exploratory Analysis

2.1 Summary of Dataset

Repay Status	Set Size	Percentage
Loan Granted	47654	47%
Loan not Granted	53446	53%

From table above, the bank accepts 47% of all loan applications

Granting Status	Set Size	Percentage
Loan Repaid	30706	64%
Loan not Repaid	16948	36%

From table above, borrowers repaid 64% of all granted loans.
Also, there are 16 columns in the dataset among which "loan repaid" is the target variable.

2.2 The Use of Data in Classification

Since I will build a classification model for this project, only granted loans' data will be used in this project. In addition, as one column corresponds to variable 'loan id' which is an unique key and should no predictive, it will be removed. In the end, the dataset used in this classification project is 14 features and 47654 observations.

2.3 Interesting Findings in Exploratory Analysis

Number of Dependents

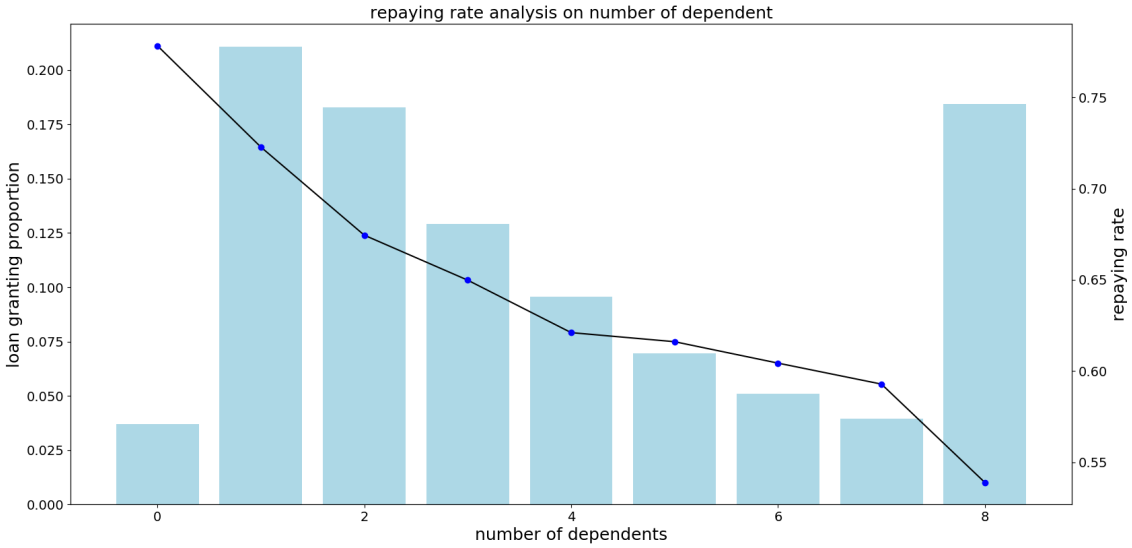
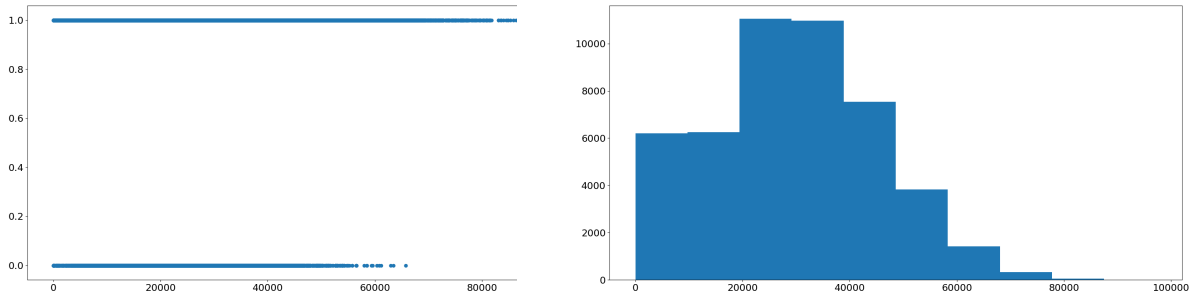


Figure 1: Barplot:Loan Granting Percentage. Lineplot:Repaying Rate

From figure above, we can see

- The repaying rate is negatively associated with the number of dependents that a borrower has.
- However, only 2.5% loan are granted to applicant who doesn't have a dependent whereas 17.5% loan are granted to applicant who has 8 dependents.
- The bank may have other factors to justify this fact/decision or it may need to modify its strategy here.

Income - Yearly Salary



(a) x:salary, y: repay status

(b) histogram of yearly salary

Figure 2: Yearly Salary and Its Impact on Repay

From figure above, we can see

- When annual salary is above 60,000, the repaying rate is very high.
- However the bank currently grants most of its loans to borrowers with less than 60,000 annual salary.
- It maybe due to the bank's general business strategy that is servicing low-to-middle income population. Otherwise the bank may want to modify its strategy.

3 Model Building

3.1 Candidate Model

Gradient boosting trees model (GBM) is widely used in classification tasks. Some of its advantages are:

- GBM is not sensitive to outliers and correlated variables. This would relieve burdens on data pre-processing and exploratory data analysis.
- GBM can deal with categorical variables.

Since there are many categorical variables in our dataset and there exist correlated variables, we use gradient boosted trees as the candidate model.

3.2 Train Test Split

In order to evaluate final model's generalized performance on an unseen dataset and compare it with the bank's current strategy, the 47654 granted loans are split into training and testing set. Stratified sampling is used in the split to make certain loan repay rate is same between training and testing set.

The testing set isn't used until the final model is determined (after hyper-parameters tuning)
The training set (further split in cross validation) is used to learn parameters and tune hyper-parameters.

Dataset	Set Size	Repaying Rate
Training	42888	64.43%
Testing	4766	64.43%

3.3 Feature Preprocessing/Engineering

In the model building phrase, a bunch of feature preprocessing and engineering techniques have been experimented. Their performance are evaluated through 5-fold stratified cross validation using untuned hyper-parameters. Unfortunately I don't get a new feature which have great predictive power. However some of the treatment are worth mentioning among which I will specifically detail the following two.

Creating a New Feature Based on Three Columns

Among the original columns, there are three variables "Is First Loan", "Fully Repaid Previous Loans" and "Currently Repaying Other Loans" that can be potentially combined together to engineer a new feature "Previous Loan Status" which indicates if borrowers have previous loan, paid previous ones or paying a current loan. Based on my analysis, I create 5 levels for the new feature "Previous Loan Status" as follows. It turns out the newly created feature "Previous Loan Status" alone is able to substitute the original three columns above in terms of predictive power. Therefore, "Is First Loan", "Fully Repaid Previous Loans" and "Currently Repaying Other Loans" are dropped in the model development phase.

- Category 1: Not repaid previous loan; Currently paying other loans
- Category 2: Fully repaid previous loan; Currently paying other loans
- Category 3: This is the first loan for the borrower
- Category 4: Not repaid previous loan; Not paying other loans
- Category 5: Fully repaid previous loan; Not paying other loans

Category	Category Proportion	Repaying Rate
1	1.41%	21.58%
2	12.21%	27.78%
3	54.20%	65.03%
4	3.08%	75.51%
5	29.08%	79.61%

One-Hot Encode Variable "Loan Purpose"

Since the "Loan Purpose" variable has no natural order among each levels but the branch splitter in GBM treats its as common numerical variable (assume order), I use one-hot encoding to transform it into 5 binary variables.

3.4 Cross Validation and Hyper-parameters Tuning

After data preprocessing and feature engineering, I did 5-fold stratified cross validation to tune the hyper-parameters in Gradient Boosted Trees in order to achieve the minimal logloss on validation set. The optimal hyper-parameters are as follows:

Hyper-parameters	Value
max depth per tree	12
η	0.01
number of trees	800
sampling percentage of features	0.5
sampling percentage of observations	0.8

Table 1: Hyper-parameters

Dataset	Logloss	Mean Score
Training	0.07	0.63
Validation	0.18	0.57

Table 2: Result

The mean score is calculated as scores my model got divided by number of loans on that dataset. It is a metric indicates how much score my model get per loan. (score's formula is specified by this challenge at the beginning of the report)

4 Evaluation

4.1 Model Performance

The comparison between bank's current model and my model is conducted on the leave-out testing set. It has 4766 loans. The result is as follows.

Model	Score of All Loans	Mean Score per Loan
Bank's Current Model	1376	0.289
My Final Model	2699	0.566

Table 3: Comparison

Based on the table above, my model outperforms the bank's current model on the testing set.

4.2 Strategy on Denied Applications

Model Prediction	True Result	Number of Cases
Repay	Repaid	2859
Repay	Defaulted	160

Table 4: True Positive and False Positive on Testing Set

From the table above, we can see that our model has much more true positive cases than false positive cases. If we grant a loan for predicted positive case, we would gain 0.89 score on average. Therefore if the bank would run our model on the previously denied applications and grants loans to those predicted repay, the bank would make more profits/scores.

4.3 Important Variables

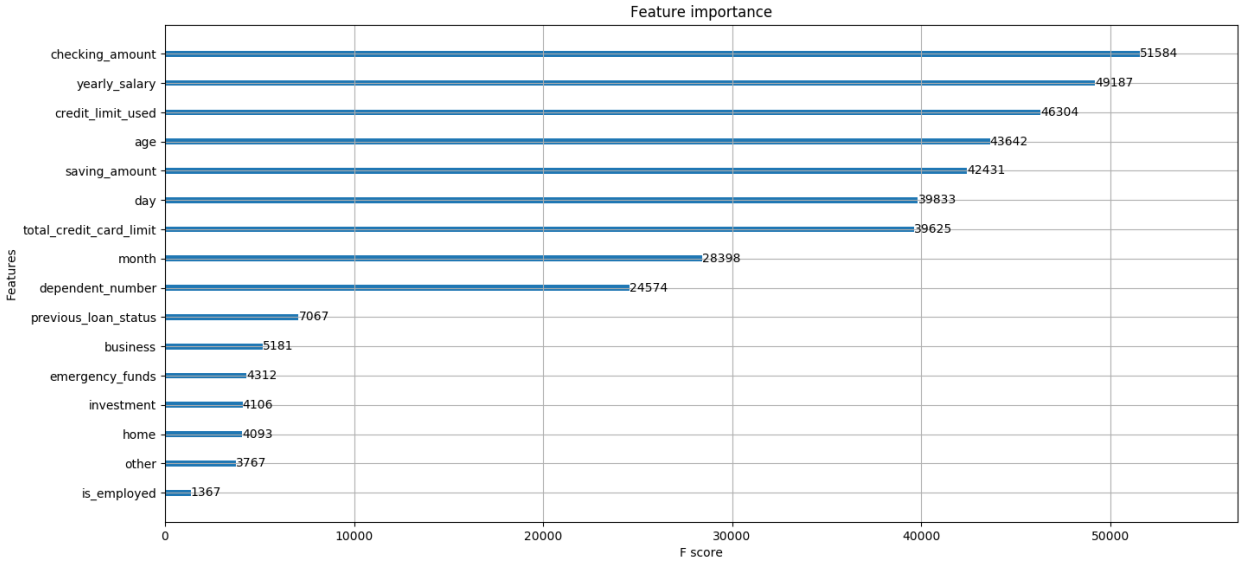
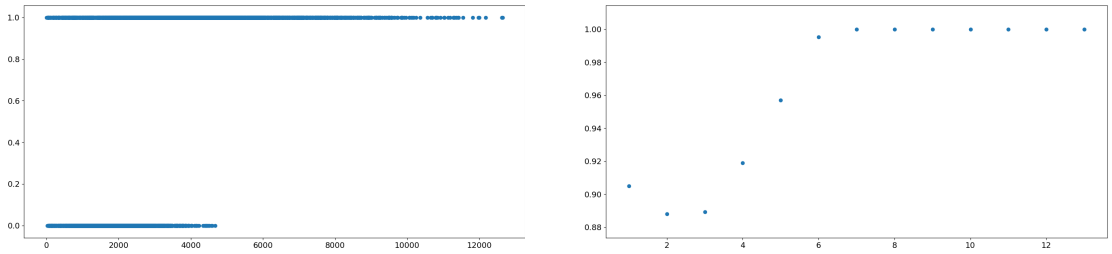


Figure 3: Feature F Score

- Variable 'Checking Amount' has the highest F-score and is important variable
  - From applicants' perspective (shown in figure 4(a)), if checking amount is greater than 5,000, my model predicts 100% borrowers will repay the loan and therefore grant all the applications. However, if the checking account amount is less than 5000, my model only grants 52.5% application.



(a) Checking Amount vs Predicted Repayment (b) Prediction Accuracy on Checking Amount Interval

Figure 4: Prediction Accuracy condition on Checking Amount Interval

  - In figure 4(b), the X-axis represents checking amount interval. For example, 1 means checking amount falls in [0,1000), 2 falls in [1000,2000) and so on. The Y-axis represents my model's performance that is if my model grants a loan, how much percentage of them get repaid on the testing set.
  - From 4(b), we can see variable 'checking amount' is very important deciding the model's prediction accuracy. Roughly speaking, the larger the checking amount, the higher the accuracy is.

- F-score for "is employed" is relatively low and its actual impact on prediction accuracy is insignificant.
  - On the testing set, there are 4357 employed applicants and 409 unemployed applicants.
  - If an applicant is employed and my model grants the loan, 94% borrowers will repay the loan. If an applicant is unemployed and my model still grants the loan, 90% borrowers repay the loan. The difference is not huge.
  - Meanwhile from the applicants perspective, "is employed" is important. Among employed applicants on the testing set, my model predicts 68% borrowers will repay and grant the loan whereas among unemployed applicants, my model only predicts 16% ones repay (grants only 16% application).

5 Conclusion

Based on the analysis and result above, We can conclude

- The bank may want to modify its strategy to increase repaying rate such as lending more to borrowers without dependents and less to borrowers with 8 dependents.
- There is more powerful classification models outperforming the current one. My well-tuned gradient boosted trees model double the current profitability on testing set.

6 Appendix

The python code related to this project can be found at

https://github.com/wenbo5565/AppliedProject_GrantingLoan/blob/master/GrantingLoan%20core.py