Lending Club Loan Status Analysis

*Xi Chen xic6, Jingyi Zhu jingyiz9*

*the department of Statistics, the University of Illinois at Urbana-Champaign*

1 Introduction

In this project, our data is from a historical loan dataset issued by Lending Club, but this dataset has over 100 features, and some of them have too many NA values, and some are not supposed to be available at the beginning of the loan. Thus, based on our goals and needs, we use a cleaned dataset here to predict the chance of default/charged off for a loan, with 30 features in total including the response 'loan\_status' and a total of 844006 observations.

2 Data Processing

2.1 Data cleaning

First, we indicate the good loans in “Fully Paid” status with 0, and bad loans in “Default” or “Charged off” status with 1. Also, as for large numbers of annual\_inc and revol\_bal, we do a log transformation. Secondly, we handle different missing values in different ways. If the value of emp\_length is missing, we fill NA with “Others” category. If the value of pub\_rec\_bankruptcies is missing, we fill NA with zero. If the values of revol\_util, dti and mort\_acc are missing, we fill NA with their column mean respectively.

2.2 feature processing

First, we add a new feature, called “fico\_avg”, which is the average of the values of “fico\_range\_low” and “fico\_range\_high”. Also, we transfer “earliest\_cr\_line” from a date format into the number of months away from Jan.1st, 2018. Next, we remove some features “id”, “grade”, “emp\_title”, “purpose”, “title”, “zip\_code”, “addr\_state”, “application\_type”, “fico\_range\_low”, “fico\_range\_high”. After that, we transfer the categorical variables with their levels as new columns and set present as 1 and absent as 0.

2.3 generate training and test dataset

To test model performances later, we use three columns of ids to split the data into three sets of training/test pairs.

3 Method

In this report, we are trying five classification methods to pick the best one among all models, which can produce an average log-loss (on the three test sets) lower than 0.45. Now, we apply five models in five functions to three pairs of datasets one by one.

3.1 Model 1: Logistic Regression

We write a function using binomial to apply the logistic regression, and predict the proportion for default/charged off status by using predict() function. We acquired the fitted values for the test data, after that we calculate the average log-loss. All average log-loss for three datasets are larger than 0.45, though smaller than 0.46.

3.2 Model 2: Regression with lasso

In this model function, we first find the best penalty lambda, and build logistic regression model with lasso. Then we predict the fitted values for the test data. Later, we calculate the average log-loss. All average log-loss for three datasets are larger than 0.45, but smaller than 0.46.

3.3 Model 3: RandomForest

When we build a function of a random forest model of 500 tree, we predict the fitted values for the test data. Later, we calculate the average log-loss. All average log-loss for three datasets are larger than 0.45.

3.4 Model 4: Xgboost

In this function useing xgboost model, we set max number of boosting iterations is 300 and low learning rate. We predict the fitted values for the test data. Later, we calculate the average log-loss. All average log-loss for three datasets are less than 0.45. Therefore, this is the model we are going to use.

We use max.depth = , iterations = 300, learning rate = 0.09, subsampling rate = and loss function

objective = "binary:logistic" means we train a binary classification model.

How you specify tuning parameters? Even if you use the default values of the command, you need to list those values in the report. For example, if you use xgboost, you need to specify tree depth, iteration steps, learning rate, subsampling rate, and loss function.

3.5 Model 5: SVM

We build a svm model to train the data and predict the fitted value by probabilities type. Later, we calculate the average log-loss. All average log-loss for three datasets are larger than 0.45.

4 Result

4.1 model selection

Below are average log-loss from the evaluation outcomes of five models:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Set1 | Set2 | Set3 |
| Xgboosting model | 0.4474065 | 0.4488618 | 0.4477756 |

*Table 1: average log-loss of three Testing Datasets*

From what we try above, we can say xgboosting model is the best one, since can produce an average log-loss lower than 0.45 for all the three test sets.

4.2 Retrain and Evaluate Model

4.2.1Retrain Original Data

We retrain the xgboosting model using all original training data. Then, we use the xgboost model built on new data from 2018Q3 and 2018Q4, which should have the similar data cleaning steps as what we do for the first dataset. However, there exists ‘Current’ loan status in the new testing dataset, and we use the features of these data to do the predictions but remove them when calculate the log-loss. We now don’t know these data’s status in the future, so the loan status cannot be divided into the ‘Fully Paid’ or ‘Default/ Charged Off’. Finally, we predict the fitted value for two new datasets.

4.2.1 Evaluation for New Data

For prediction on the new data from 2018Q3 and 2018Q4, the log-loss is 0.6493325 and 0.6576236 respectively.

We randomly select 10 samples of prediction results from 2018Q3 and 2018Q4. The xgboost model we selected above can tell us the potential probability for a customer in “current” status to pay for the load fully or to default/ charge off. For example, if a fitted value for an observation is close to 0, the customer is more likely to pay off the load fully. In contrast, if a fitted value is closer to 1, the customer might be failed to pay for his or her load. In this bad case, the lender should be aware of this, and take some actions to follow it.

*Table2: 10 samples from 2018Q3*

|  |  |  |
| --- | --- | --- |
| Sample | ID | prediction |
| 1 | 140386332 | 0.5829706 |
| 2 | 137596254 | 0.4814080 |
| 3 | 139042421 | 0.4033672 |
| 4 | 137958160 | 0.5278149 |
| 5 | 136676909 | 0.6138587 |
| 6 | 138950684 | 0.5264910 |
| 7 | 139213780 | 0.4660960 |
| 8 | 140888573 | 0.6700576 |
| 9 | 137212149 | 0.5338041 |
| 10 | 138249446 | 0.6581378 |

As table2 shows, observations with small prediction value, such as the #3 sample, are more likely to pay for load on time, while #8 and #10 sample are more likely to not pay for load on time, since the prediction values are larger.

*Table3: 10 samples from 2018Q4*

|  |  |  |
| --- | --- | --- |
| sample | ID | prediction |
| 1 | 142346158 | 0.6397886 |
| 2 | 142963300 | 0.4858047 |
| 3 | 139943868 | 0.6108240 |
| 4 | 145151814 | 0.6805556 |
| 5 | 145429487 | 0.5425312 |
| 6 | 143193025 | 0.5229566 |
| 7 | 141936844 | 0.5506626 |
| 8 | 145409123 | 0.5857298 |
| 9 | 144149074 | 0.5734055 |
| 10 | 144109839 | 0.6852481 |

As table3 shows, observations with small prediction value, such as the #2 sample, are more likely to pay for load on time, while #4 and #10 sample are more likely to not pay for load on time, since the prediction values are larger.

5 Other Information

• Computer system: MacBook Pro, 2.53 GHz, 4GB memory

• Running time: 2.25023