# 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

We use R for coding, and all packages we use are listed as below: xgboost, glmnet, randomForest, kernlab, caret, tidyr, tidyverse, lubridate.

### 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.1$^{st}$, 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

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Logistic regression transforms its output using the logistic sigmoid function to return a probability value. 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

Lasso regression uses the L1 penalty term and stands for Least Absolute Shrinkage and Selection Operator. 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

Random forest is a very popular ensemble method that can be used to build predictive models for both classification and regression problems. Ensemble methods use multiple learning models to gain better predictive results. This model creates an entire forest of random uncorrelated decision trees to arrive at the best possible answer. When we build a function of a random forest model of 500 trees, 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

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. We use max.depth = 6, iterations = 300, learning rate = 0.09, subsampling =1 and loss function L2 regularization term. Also, objective = "binary:logistic" means we train a binary classification model. 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. So this is the model we are going to use.

### 3.5 Model 5: SVM

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. 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:

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

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

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.

### 4.2.2 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 | 137985446 | 0.6268709 |
| 2 | 139365050 | 0.6604961 |
| 3 | 137380219 | 0.5168082 |
| 4 | 140426852 | 0.5371727 |
| 5 | 140665677 | 0.5172512 |
| 6 | 137708109 | 0.6225411 |
| 7 | 138622266 | 0.5482422 |
| 8 | 140939644 | 0.5142216 |
| 9 | 139258182 | 0.6096013 |
| 10 | 139215476 | 0.7224717 |

*Table3: 10 samples from 2018Q4*

| sample | id | prediction |
|---|---|---|
| 1 | 141572384 | 0.5431699 |
| 2 | 145244575 | 0.6845984 |
| 3 | 144872725 | 0.5838404 |
| 4 | 143449686 | 0.7694326 |
| 5 | 142802916 | 0.5289983 |
| 6 | 143218129 | 0.5634968 |
| 7 | 141199223 | 0.6495010 |
| 8 | 142964379 | 0.5110934 |
| 9 | 143270644 | 0.5923718 |
| 10 | 141931467 | 0.5769194 |

Explanation: We use the whole 2018Q2 as the train set and apply to xgboost model. Then we create the importance matrix to understand how the model gives the predictions based on 2018Q3 and 2018Q4 samples. From table4, we know that the int_rate, dti, annual_inc occpy the top 3 importance and these features influence on the predictions most. Therefore, the xgboost model gives the sample predictions based on the importance matrix and it splits the node at the proper value of the features we selected from the root.

*Table4: importance matrix from xgboost model*

| Importance rank | Feature | Frequency |
|---|---|---|
| 1 | int_rate | 0.0786321354 |
| 2 | dti | 0.0976376899 |
| 3 | annual_inc | 0.0967677173 |
| 4 | revol_bal | 0.1009168172 |
| 5 | installment | 0.0809743693 |
| 6 | term | 0.0078966740 |
| 7 | earliest_cr_line | 0.0831158402 |
| 8 | revol_util | 0.0834504450 |
| 9 | fico_avg | 0.0472462022 |
| 10 | mort_acc | 0.0288429365 |

## 5 Other Information

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

• Running time: 2.25023 hours