Predicting Default Rate of Lending Club Loans: Analysis by Lasso Penalized Logistic Regression and Nonlinear Methods

ECO3080: Machine Learning for Business

Instructor: Prof. Qihui CHEN and Dr. Long MA

The Chinese University of Hong Kong, Shenzhen

Group Members:

120020054 Qinyang YU

120020062 Xian JU

120020095 Runxin ZHI

120020150 Jingnan ZHANG

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Predicting Default Rate of Lending Club Loans: Analysis by Lasso Penalized Logistic Regression and Nonlinear Methods

1. Introduction

P2P (Peer-to-peer) lending is an emerging form of financing that relies on the internet and financial technology. It provides a direct online transaction platform for borrowers and lenders, eliminating the need for traditional intermediaries such as commercial banks. However, while it improves the efficiency of social capital utilization, it also exposes investors to increased risks. Compared to liquidity risk and platform-collapse risk, the most significant risk comes from the default risk of borrowers. That is because many loans are not collateralized, and platforms cannot freeze other assets of borrowers when they default, which makes it very difficult to recover the defaulted loans (Chang et al., 2022, p. 2). Therefore, predicting and managing the risk from customer defaults is crucial in enhancing the prosperity of P2P lending platforms and safeguarding the interests of investors.

A straightforward way to manage the risk of loan default is assigning credit ratings to applicants, where applicants with low default risk should be assigned higher credit ratings, while applicants with high default risk have low credit ratings and need to pay more interest to compensate for the risk. However, a study by Ma et al. (2022) shows a mismatch between P2P loan default and credit rating based on the traditional banking system (p. 1). Therefore, credit rating is insufficient to manage the risk of loan default.

Lending Club, the world's first publicly traded P2P lending platform and industry leader, faces the same crisis. Table 1 shows the classification result directly based on credit grade of Lending Club and Figure 1 displays its ROC curve.

[Insert Figure 1]

[Insert Table 1]

As shown in Table 1, Lending Club's credit rating is not sufficient to reflect the actual loan default situation. Therefore, this paper hopes to investigate the effectiveness of different machine learning models to provide more accurate predictions of loan default based on Lending Club's credit ratings.

The focus of this study is to evaluate linear and nonlinear statistics method, compared to the baseline model of direct prediction based on credit rating. The linear method centered on logistic regression. We aim to improve the default prediction accuracy by optimizing the combination of features composition, regularization methods, and probability transformation. Linear method is preferred due to its interpretability, which facilitates the discovery of applicants' default reasons and captures clearer default identification. For nonlinear methods, this study employs two types of artificial intelligence methods (i.e., random forest and feedforward neural network) as comparison with linear statistics methods. Possessing their own advantages and disadvantages, these different models are compared in dimensions including of classification accuracy and running to determine the most suitable model for predicting loan defaults.

2. Data

2.1 Data Source

This study uses 140,837 loans from Lending Club incurred in 2015 as the sample. The raw data is cross-sectional and has information on 140,837 observations for 23 variables. These variables are part of the information collected by the platform about the applicant and the loan, including the applicant's income, credit history, the purpose of loan, etc.

We use the "loan_status" variable as the reference label for default. It takes two values, "Fully Paid" means the loan is not in default, and "Charged Off" means the loan is in default. The number of defaulted loans is 27,942, about 19.84% of the entire sample.

2.2 Data Cleaning

2.2.1. Removing redundant features

First, we have removed two types of data. The first category is information irrelevant to loan defaults, i.e., the user's ID on Lending Club. The second category is exceptionally singular descriptive information, the applicant's job title, and zip code. They are both unordered categorical variables, and the values differ across most of the observations, which makes it challenging to extract information from them.

2.2.2 Converting data type

In addition to the outcome variable, 5 of the 19 features selected for the study are categorical variables, which needs to be transformed before applying to the model.

We started by converting the outcome "loan_status" into a dummy variable, where "Fully Paid" was assigned the value of 1 and "Charged Off" was assigned the value of 0.

Among the five categorical regressors, credit rating "grade" and employment duration "emp_length" are ordered variables. We converted them into ordered factors according to credit rating from high to low and employment length from short to long.

The remaining three categorical features are unordered: loan duration "term", home ownership status "home_ownership", and purpose of the loan "purpose". These variables have no intrinsic order, so we transform them directly into unordered factor-type data.

2.2.3 Handling Missing Values

Before training the model, we have to deal with the missing values present in the dataset. Figure 1 shows that there is only one variable "bc_util" with missing values. The quantity of missing values is 1440, accounting for 1.02% in the total sample. Since the percentage of missing values is small and does not have a significant impact on the final results, we use the direct deletion method.

[Insert Figure 2]

2.2.4 Handling Extreme Values

After summarizing the data, we have found that only one observation has “purpose” as “wedding”, and only one observation has “home_ownership” is “ANY”. These two results are highly individualized and cannot exist in both the training and validation sets. Therefore, we remove these two observations.

2.3 Summary Statistics

According to loan_status, people can be divided into two groups: someone whose loan is defaulted or not. By analyzing numerical and ordinal features, we can observe that: compared to the “Fully Paid” people, those who have a loan in default also have higher loan amounts, ratio of total current balance to high credit, dti, number of open credit lines and lower “upper boundary range”, “lower boundary range”, number of mortgage account and possibility to have high credit grade (A, B, C, D).

From those unordered categorical features, we can see no significant difference in employment length in the year between the two groups of people. And compared to people whose home ownership is owned, those who mortgage to live a house have a lower percentage of having the loan in default, while people who rent their houses have a higher rate of having the loan in default. Besides, among all purposes of loans, people who loan for renewable energy have the most significant proportion to default, while those who borrow to pay credit card have the smallest proportion to default.

[Insert Figure 3-21]

3. Model

3.1 Baseline Model

The baseline model of this study is purely based on the borrowers’ loan grades assigned by Lending Club. Under every grade (A-G) of the training data set, we calculate the ratio of the number of borrowers with charged off loan to the total number of borrowers, which is the expected probability of the borrower under the corresponding grade to charge off the loan. The charge-off probability for each grade is shown in Table 1.

The baseline model then creates a random number between zero and one for each borrower in the test data set. If the random number is lower than the corresponding charged-off probability of the borrower’s grade, the baseline model classifies the borrower’s loan status to be “Charged Off.” Essentially, the baseline model is a random guess of the borrowers’ loan status, based on the average proportion of charged-off status under every grade.

3.2 Logistic regression

The linear method used in this study is based on logistic linear regression. Linear regression is preferred because it is easy to understand and interpret the correlation between the response variable and independent variables. To perform classification, the output should be limited with zero (charged-off) and one (fully-paid) representing the possibility of the borrower to charge off the loan. This paper applys the logistic regression to transform the linear output to a probability:

$$p(X) = \Pr(Y = 1 \mid X) = \frac{e^{X\beta}}{1 + e^{X\beta}} \quad (1),$$

where X includes the independent variables: the numeric variables and dummy variables generated from the categorical variables. Each categorical variable with n categories generates (n-1) dummy variables, leaving one category out for baseline.

The logistic regression uses maximum likelihood to estimate:

$$L(\beta) = \prod_{i:Y_i=1} p(x_i) \prod_{i:Y_i=0} (1 - p(x_i)) \quad (2),$$

where $p(x_i)$ calculates the probability of x_i being in category $Y_i = 1$ by equation (1). As the linear combination in $p(X)$ cannot model for nonlinear relationships, we further add interaction terms into the (generalized) linear regression. To avoid high dimensional problems, we use lasso regression to make the model self-selective. Figure 34 also shows there exists multicollinearity of some variables, which can be deleted by lasso.The lasso approach requires β to minimize:

$$-\ln(L(\beta)) + \lambda \sum_{j=1}^{q} | \beta_j | \quad (3),$$

which is equivalent to:

$$\begin{aligned}\beta &= \arg\min_{\beta} \ln[\prod_{i:Y_i=1} p(x_i) \prod_{i:Y_i=0} (1 - p(x_i))] + \lambda \sum_{j=1}^{q} | \beta_j | \\ &= \arg\min_{\beta} \ln[\prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}] + \lambda \sum_{j=1}^{q} | \beta_j | \qquad (4). \\ &= \arg\min_{\beta} [\sum_{i=1}^{n} y_i x_i \beta - \ln(1 + e^{x_i \beta})] + \lambda \sum_{j=1}^{q} | \beta_j |\end{aligned}$$

where n is the number of observations in the training data set, q is the number of independent variables, and λ is chosen by cross-validation.

This paper respectively estimates three models under the logistic regression: the basic logistic regression, the basic logistic regression, the logistic regression with lasso penalty term, and the logistic regression with lasso penalty term and interaction terms. The logistic regression is a common classifier, which intuitively limit the linear output to [0, 1] as a probability, but it can be overly simple and does not assign preference between sensitivity and specificity. In loan default management, companies care more about the false positive rate than the false negative rate when estimating if the borrower will fully pay the loan. Thus, this paper focus more on improving specificity than sensitivity, and we specifically determine a borrower's loan status to be "charged-off" if the probability estimation is less than 0.8 (instead of 0.5.)

3.3 Tree-based Methods

Tree-based methods, including bagging and random forests, are used in this study. The tree methods are simple and useful for interpretation.

Currently, logistic regression and neural network models are popular machine learning algorithms for classification tasks. However, the logistic model requires that the feature variables are independent of each other and is sensitive to multicollinearity, which is difficult to guarantee in the actual situation. In addition, the neural network is equivalent to the black box, which cannot be effectively explained in the sense of economics. The tree-based method can solve the above two problems well. It has excellent interpretability and can easily handle qualitative predictors.

Nevertheless, the trees can be easily overfitting and sensitive to a slight change in the data. In terms of predictive accuracy, trees are also generally considered not as good as classification approaches.

3.3.1 Simple tree method

First, for the simple classification tree method, we want to predict the loan status using other features. We built up a simple tree on the training set and predicted on the test set. We then calculated sensitivity and specificity according to the confusion matrix. Additionally, cross-validation (CV) was considered to prune the tree. We chose the best number of leaves, “2”, and plugged it into the function to get a pruned tree. However, the resulting confusion matrix shows no improvement because of the pruning. The reason might be that the size of the initially generated tree is small, and the room for improvement is minimal.

[Insert Figure 22]

[Insert Table 2]

3.3.2 Bagging

We conducted bagging on the training and test set and obtained the confusion matrix. We also changed the number of trees to 1000 and calculated the MSE to compare with the previous one. The result implies that the previous bagging is preferred for its smaller MSE.

[Insert Table 3]

3.3.3 Random Forest

The variable important scores of random forests are OOB and Gini. They are respectively listed for Mean Decrease Accuracy and Mean Decrease Gini. In the beginning, we set the number of trees large enough to establish a preliminary random forest model and obtained the variable importance score, as shown in Table 4.

[Insert Table 4]

In order to stabilize the results, five-fold cross-validation was adopted to determine the optimal feature subset. Finally, 16 features were obtained. We dropped “deling_amnt”, "ing_last_6mths", and "home_ownership".

We adjusted the number of split point candidate features (mtry) to 4, which is the root of the total number of variables. Then we predicted the model on the test set and calculated the MSE. We found that the MSE of random forest is the smallest. A confusion matrix was also calculated as well as the sensitivity, specificity, accuracy, and precision.

3.4 Neural Network

The second artificial intelligence method used in this study is the neural network, a nonlinear model proposed by McCulloch and Pitts in 1943 (Fitch, 1944). It mimics the structure and logic of the human brain and can learn complex nonlinear relationships. The capability of neural network to learn data features automatically also makes it excellent at handling high-dimensional data, eliminating the need for manual feature engineering. These advantages make it well suited to the task of this study.

However, it also has some disadvantages compared to linear statistical models. First, given the large amount of data in this dataset, the neural network requires a longer training time. Second, the interpretability of neural network is poor, so that we cannot directly derive which factors cause the default. Third, the hyper-parameters of the model such as the number of neurons and the compilation method need to be chosen artificially, and these choices lack an objective basis. Fourth, due to the high complexity of this model, overfitting can be a problem and needs to be avoided by regularization.

3.4.1 Model-Specific Data Process

Due to the neural network is sensitive to the scaling of regressors, the input matrix needs to be centralized. The study uses the *scale()* function, which divides each variable value by the standard deviation after subtracting the mean. Also, a new variable is generated for each value of the categorical variable. This allows direct comparison between values of different variables and provides a normalized input.

3.4.2 Model Design

3.4.2.1 Model Structure

In light of previous research and the experimental results of this study, we used a multilayer neural network containing two hidden layers, with 128 nodes in each layer (Chang et al., 2022; Mijwel, 2018). The sigmoid function was chosen as the activation function for each hidden layer and the final output. It converts the input values to values between 0 and 1 and is ideally suited for binary classification problems.

$$S(x) = \frac{1}{1 + e^{-x}}$$

Because the neural network has a huge number of coefficients, it is important to avoid overfitting because it determines how the model performs on the test set. We carried out dropout regularization after each layer, which is especially suitable for the case with a large amount of data. This method randomly turns the output of some nodes to 0 to reduce the model complexity. In order to prevent the model from insufficient generalization ability, we chose 0.4 as the dropout rate of the first hidden layer and 0.3 as the dropout rate of the second hidden layer.

3.4.2.2 Model Compiling

After determining the model structure, we need to compile method with criteria to define the target of model training. Since we intend to solve the binary classification problem, binary cross-entropy was chosen as the loss function. According to its definition, it can measure the purity of the predicted values - if the distribution of predicted labels is close to 0 or 1, the cross-entropy will be smaller. We chose the adaptive movement estimation (adam) method as the optimizer, which is developed from stochastic gradient descent and can effectively adjust the learning rate to make the model converge faster. In terms of metrics, the study used a self-defined function, specificity, as the indicator. Because the percentage of defaulted loans in the dataset is relatively small, using accuracy alone as the metric can be biased. Moreover, the lending platforms are more concerned with the prediction of default cases than the overall situation.

3.4.2.3 Model Training

After completing the first two steps of neural network setup, the data from the training set were imported for model fitting. Because of the large amount of data, the epoch number was 128 to ensure that the model sufficiently learns data features. We chose a batch size of 512 in order to have more sample data to participate in each iteration the model parameters are updated. We also changed the relevant parameters several times and checked the results to ensure that the selected hyper-parameters were the most suitable for our dataset and model structure. The result of training is exhibited in Figure 23.

[Insert Figure 23]

4. Result

4.1 Obtaining Training Set and Validation Set

For the main estimation of the four methods, we randomly select 80% of the full data set observations as the training data set, and the rest as the validation data set. Theoretically, the ratio of fully-paid borrowers over charged-off borrowers should be similar in the training, validation, and the full data set.

Considering that there is an imbalance between the two kinds of borrowers because the number of charged-off borrowers is intrinsically small, we also repeat the model estimations on the adjusted training set and validation set. We randomly choose 10,000 observations from fully-paid observations and charged-off observation respectively, and use the obtained 20,000 observations as the training set and the rest 112,362 observations as the validation set. This approach can guarantee that the number of observations of fully-paid and charged-off loan statuses are equal in the training set, and thus solve the data imbalance problem. However, the drawback is that due to the much fewer observations of charged-off borrowers, the training set must be small to be balanced while keeping enough charged-off observations in the validation set. The small training set may prevent the models from capturing adequate data characteristics, while randomly selecting can obtain a large training set. Thus, we repeat the model estimation by both approaches and compare their results.

4.2 Model Estimation Criteria

This paper estimates the model by three criteria. First, we compare the accuracy, precision, sensitivity and specificity of the model estimation, where models with higher specificity is prior because it represents lower false positive rate. Second, we draw the receiver operating characteristics (ROC) graph and calculate the area under the ROC curve (AUC,) where models with higher AUC is preferred because it represents lower total error rate. Third, we consider the time used for the computer to perform the model estimation, and less time used is preferred because the model estimation needs to be convenient and easy to apply in reality.

4.3 Model Performance and Selection

4.3.1 Specificity and Other Indexes

The logistic-lasso method are estimated using the imbalanced and balanced training set respectively. For the imbalanced training set, we first estimate the baseline model and compare with the three logistic regressions. The specificities of the four models are 26.66%, 68.30%, 35.14%, 35.33% and the sensitivities of the four models are 82.04%, 64.97%, 87.44% 87.29%, respectively (Table 5, 6, 7, and 8). Compared to the baseline, the pure logistic regression greatly increased specificity at the sacrifice of sensitivity, while the logistic-lasso regression simultaneously raises both indicators, improving total prediction accuracy. Adding interaction terms has minor improvements on the lasso-logistic regression. Thus, the selection of pure-logistic and logistic-lasso faces a tradeoff between specificity and sensitivity. Since our aim is to reduce the default rate, a lower false positive rate and higher specificity is mainly pursued according to the risk aversion decision threshold, implying the pure logistic regression is relatively preferred.

To check for robustness, we use the balanced training set and re-estimate the above results. The specificities of the four models are 25.64%, 65.15%, 32.00%, and 32.10% (Table 9, 10, 11, and 12.) Similarly as the logistic regression model on the imbalanced training set, the basic logistic model improves specificity most, while the lasso-logistic model has some improvements compared with the baseline model but does not perform as well as basic logistic regression. Interaction terms still have a moderate effect.

For the two non-linear method, the specificities of the three tree-based models are 0%, 0.08%, 0.06% and the sensitivities are 100%, 97.87%, and 98.63% (Table 2, 3, and 4). Even though these tree-based method achieve a high accuracy and precision of around 81% and 82%, their low specificity emerges poor feasibility and fail to satisfy the study's need.

The neural network model, instead, presents a high and well-balanced predictive power, with specificity of 66.37% and sensitivity of 65.80% (Table 13).

4.3.2 AUC

The AUCs of the baseline model in both imbalanced and balanced training set are 0.538 (Figure 24 and 30). The AUCs of the three logistic regressions are 0.658, 0.613, 0.613 and 0.657, 0.606, 0.606 with little difference (Figure 24, 25, 26, 31, 32, and 33). The total error rates reflected by the AUCs confirms the four models' performance concluded above. The three tree-based models present AUCs of 0.500, 0.529, and 0.524 and the neural network model presents an AUC of 0.721 (Figure 38). Based on this criteria, the logistic and neural network models both possess more outstanding prediction performances than other methods

4.3.3 Time Complexity

Time complexity is another dimension to measure the practicality of these methods. In the imbalanced and balanced sample, the time complexity of the baseline model is 4 and 2 in seconds and that of pure-logistic and logistic-lasso models are 4, 105 and 2, 30. The index for the neural network model is 127, while all the tree methods and logistic-lasso with interaction terms experience time explosion. The result support the pure logistic method to the most highly time-efficient method under the premise of ensuring prediction accuracy.

5. Conclusion

This paper proposes three methods, namely logistic regression, tree method, and neural network. All methods make improvements in terms of specificity and AUC based on the baseline model, which merely classify loan status by the borrowers' grades. Specifically, in logistic regressions, the basic logistic regression performs best, as it improves specificity most and has relatively less time complexity. Lasso-Logistic regressions (with or without interaction terms) can improve the total error rate because lasso selects the best regressors and keeps the model simple enough to be applicable to different test sets. However, as the loan company cares more about the false positive rate when determining if a borrower will fully pay the loan, we select the basic logistic model that can best improve specificity.

We also found that tree-based methods did not do a good job in terms of specificity, AUC, and time. One possible reason is that tree-based methods are very sensitive to unbalanced categorical data, in which case the direction of the tree will grow towards the majority. Second, the decision tree tends to ignore the relationship between data. Thus, the trees may not perform well when dealing with large, high-dimensional, and highly correlated data.

The Neural Network outperforms other models in terms of AUC, which can be attributed to its strong capability in extracting data features. However, it has many hyperparameters that need to be manually adjusted, and the ones applied in this study may not be the optimal combination. The feedforward Neural Network is also unsatisfactory in terms of specificity. Therefore, it is recommended for future study to try various type of Neural Network such as Recurrent Neural Network (RNN) and select the most suitable type. Also, our study does not consider the data that are highly personalized (e.g., zip codes and titles,) and future studies and refine the model by taking these data and natural language processing into consideration.

References

Chang, A., Yang, L., Tsaih, R., & Lin, S. (2022). Machine learning and artificial neural networks to construct P2P lending credit-scoring model: A case using lending club data. *Quantitative Finance and Economics*, *6*(2), 303-325. https://doi.org/10.3934/qfe.2022013

Fitch, F. B. (1944). Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, Vol. 5 (1943), pp. 115–133. *Journal of Symbolic Logic*, *9*(2), 49-50. https://doi.org/10.2307/2268029

Ma, h., Li, G., Liu, R., Zhang, K., & Wang, Z. (2022). Research on credit scoring method matching the probability of default: evidence from Lending Club. *Applied Economics, (ahead-of-print)*, 1–14. https://doi.org/10.1080/00036846.2022.2140769

Mijwel MM (2018) Artificial Neural Networks Advantages and Disadvantages. Available from: https://www.linkedin.com/pulse/artificial-neural-networks-advantages-disadvantages-maad-m-mijwel/

Appendix

Table 1: Error rate of the baseline model on the imbalanced training set

Pred.\True.	0	1
0	1336	3820
1	3675	17446

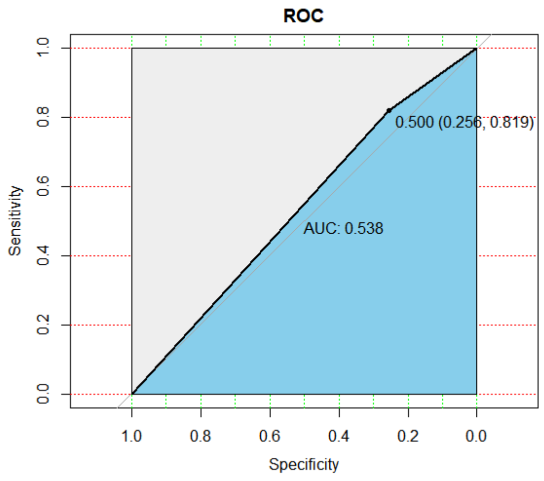


Figure 1: ROC of the baseline model on the imbalanced training set

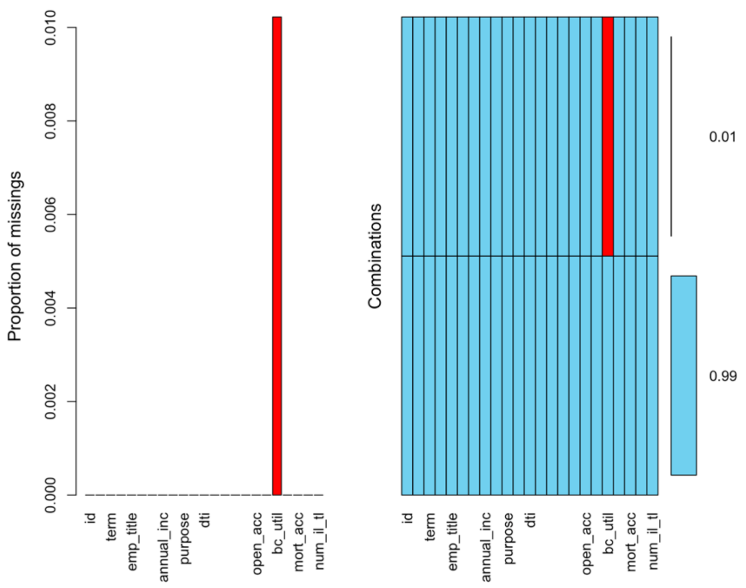


Figure 2: Display of Missing Values

Loan_amnt

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1000    8500   14000   15239   20000   35000
```

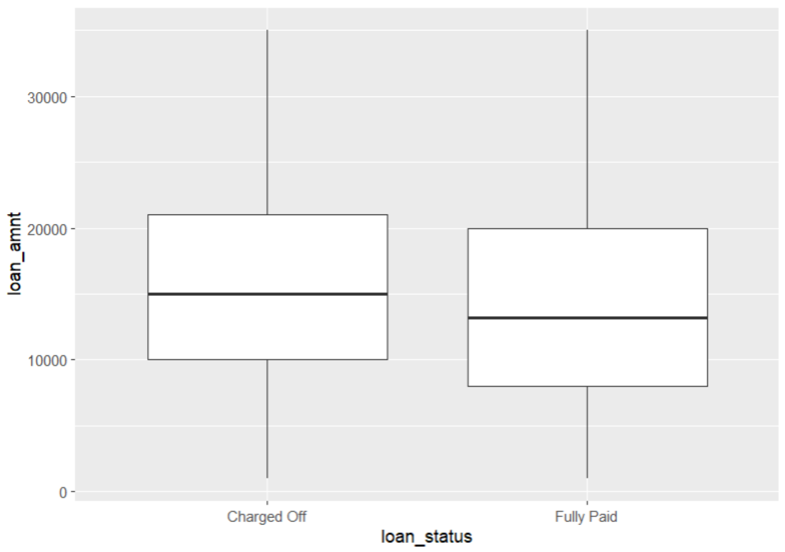


Figure 3. Loan_amnt and loan_status

annual_inc

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
5000   48000   65900   78443   94000 9500000
```

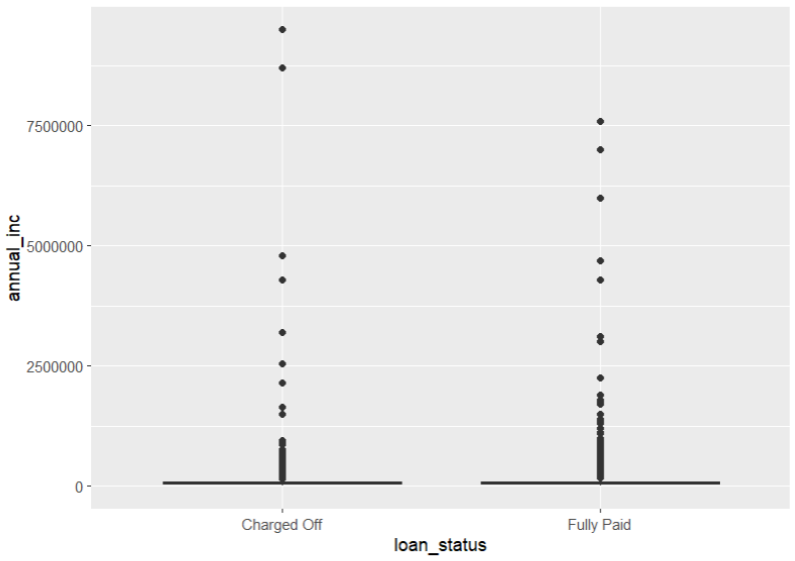


Figure 4. Annual_inc and Loan_status

bc_util

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00   41.30   65.40   61.96   86.10  243.80
```

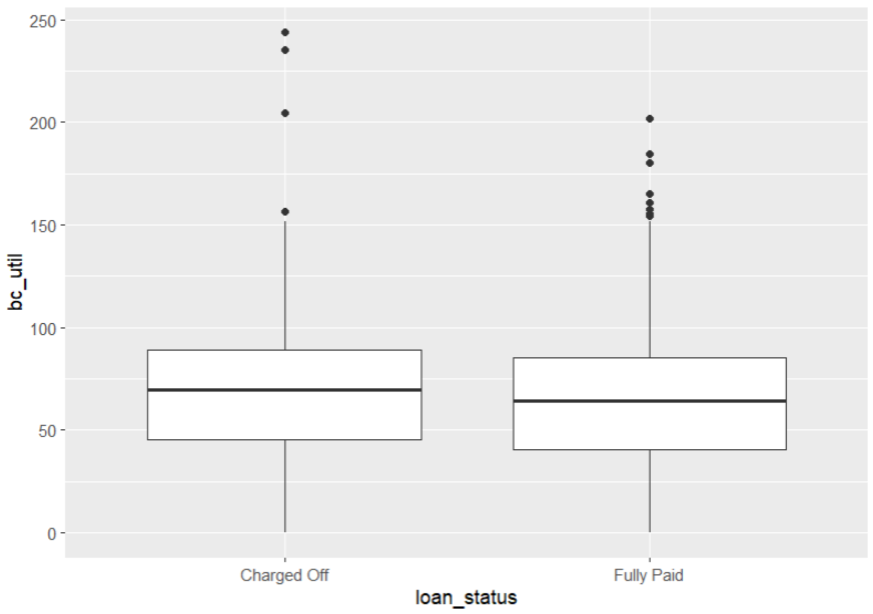


Figure 5. Bc_util and Loan_status

delinq_2yrs

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0000  0.0000  0.0000  0.3484  0.0000 27.0000
```

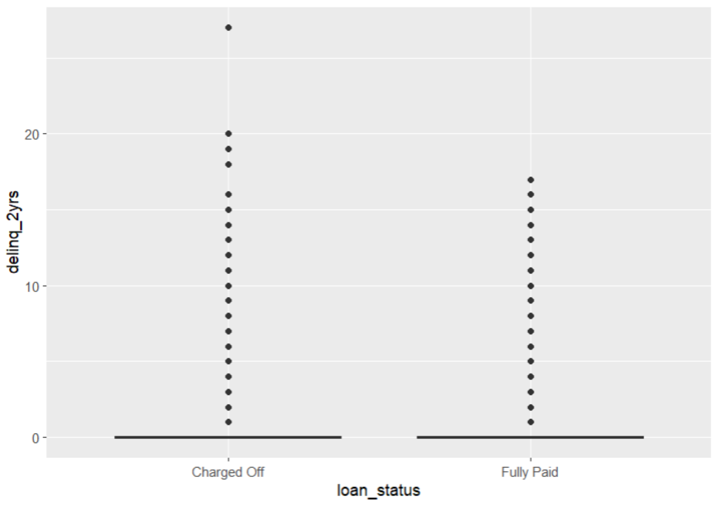


Figure 6. Delinq_2yrs and Loan_status

delinq_amnt

```
Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.00     0.00     0.00    16.35     0.00 88216.00
```

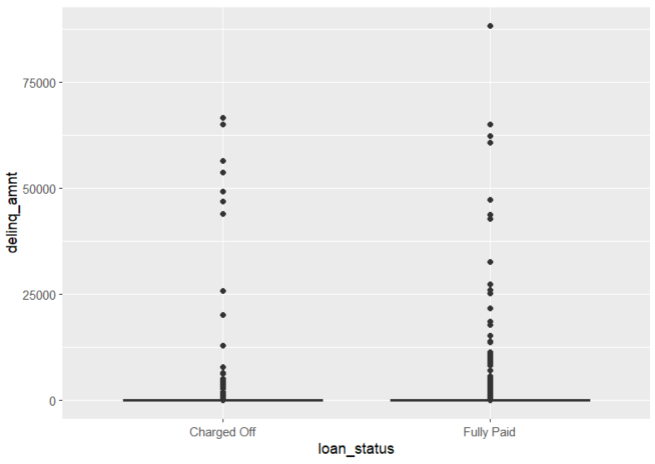


Figure 7. Delinq_amnt and Loan_status

dti

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00   12.54   18.41   18.99   25.07   39.99
```

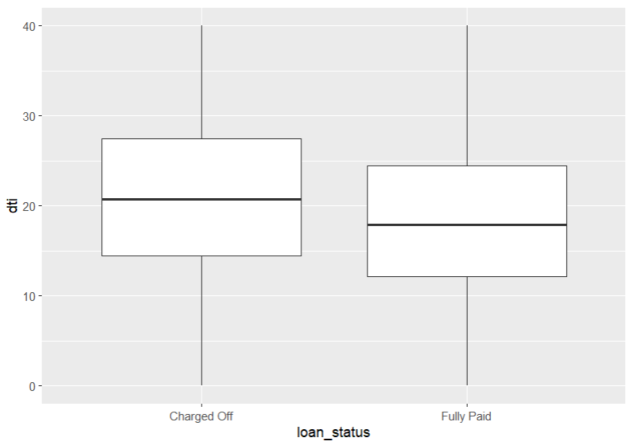


Figure 8. Dti and Loan_status

fico_range_high

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
664.0   674.0   689.0   697.6   714.0   850.0 
```

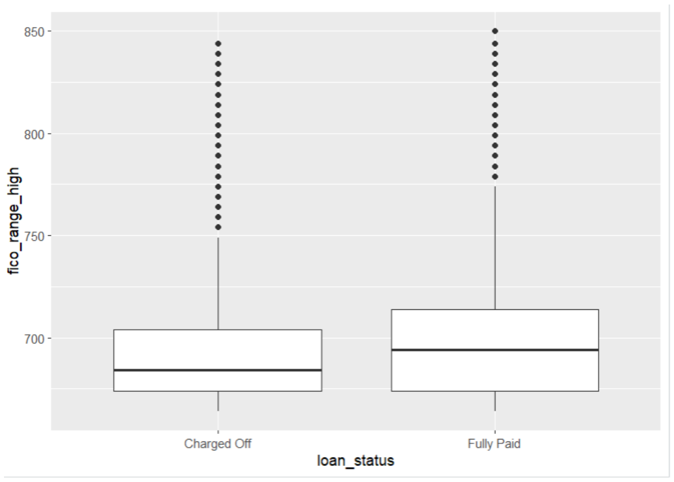


Figure 9. Fico_range_high and Loan_status

fico_range_low

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
660.0   670.0   685.0   693.6   710.0   845.0 
```

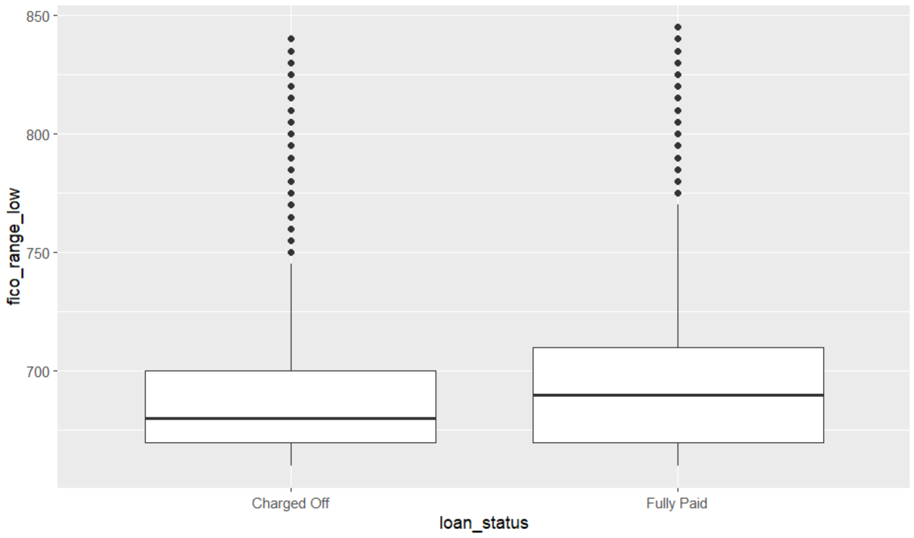


Figure 10. Fico_range_high and Loan_status

inq_last_6mths

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.000   0.000   0.000   0.577   1.000   6.000
```

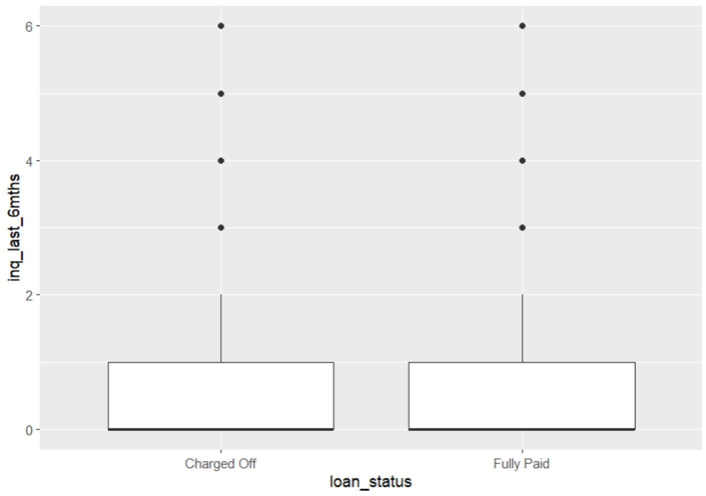


Figure 11. Inq_last_6mths and Loan_status

mort_acc

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.000   0.000   1.000   1.673   3.000  47.000
```

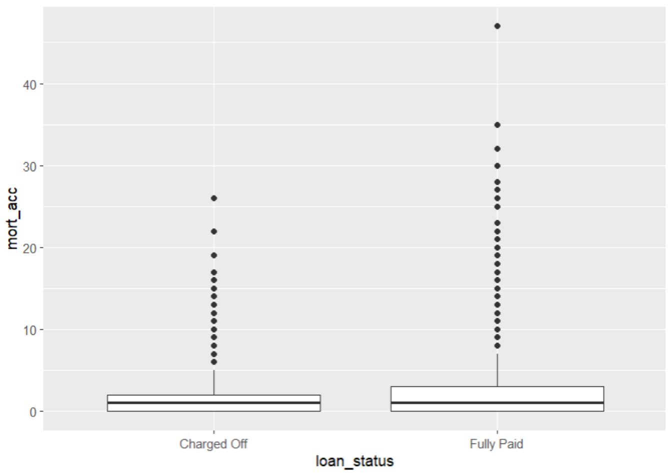


Figure 12. Mort_acc and Loan_status

num_bc_tl

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.000   5.000   7.000   8.171  11.000  57.000
```

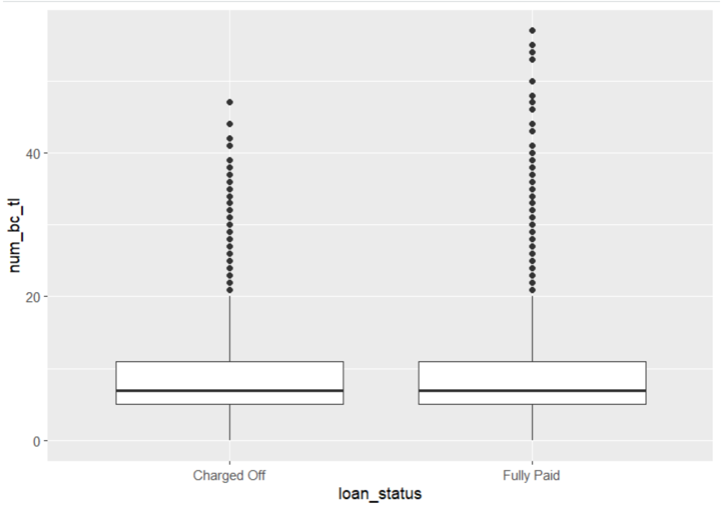


Figure 13. Num_bc_tl and Loan_status

num_il_tl

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.000   4.000   7.000   8.775  12.000 131.000
```

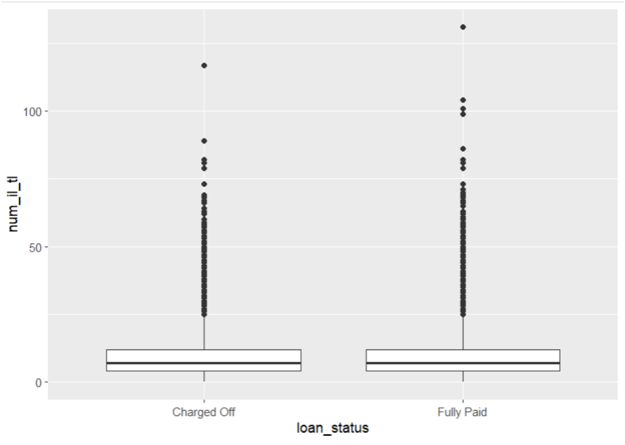


Figure 14. Num_il_tl and Loan_status

open_acc

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.00    8.00   11.00   12.05   15.00   79.00
```

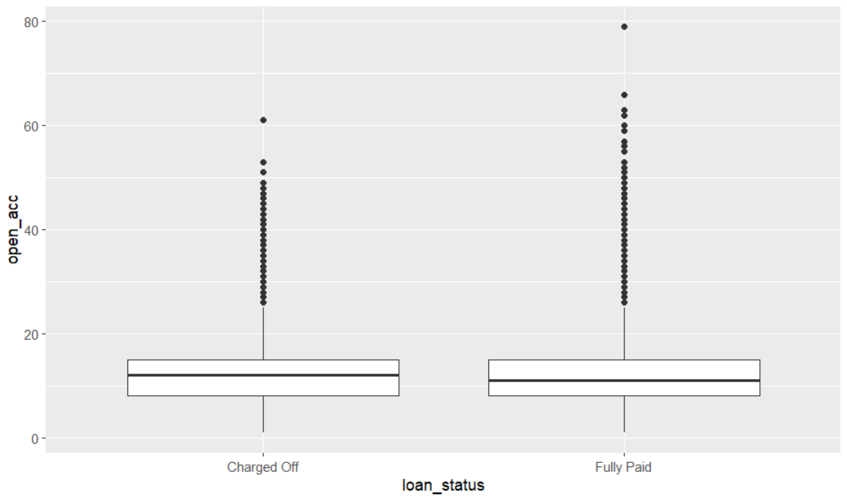


Figure 15. Open_acc and Loan_status

total_acc

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 4.0    17.0    24.0    25.6    32.0   162.0
```

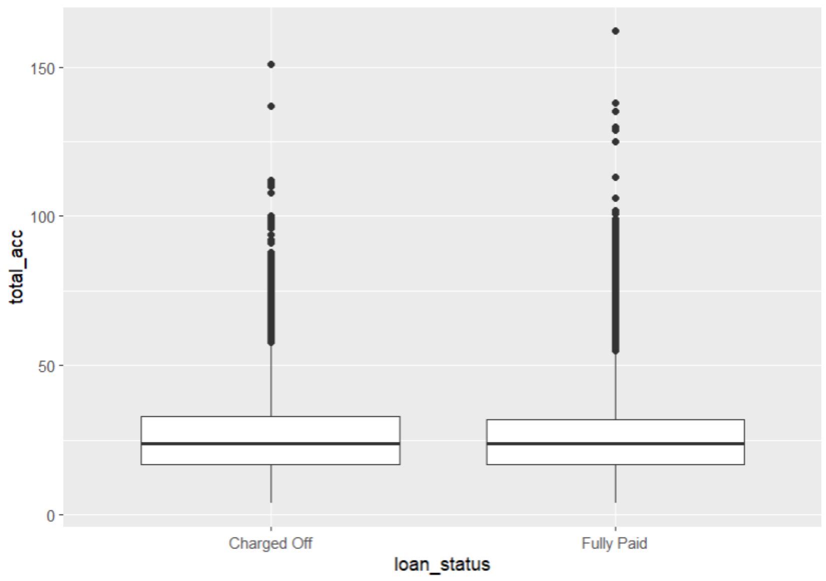


Figure 16. Total_acc and Loan_status

emp_length

```
< 1 year    1 year   2 years   3 years   4 years   5 years
   11599      9072     12372     11108      8114      8339
 6 years   7 years   8 years   9 years 10+ years
    5484      6032      7143      5544     46574
```

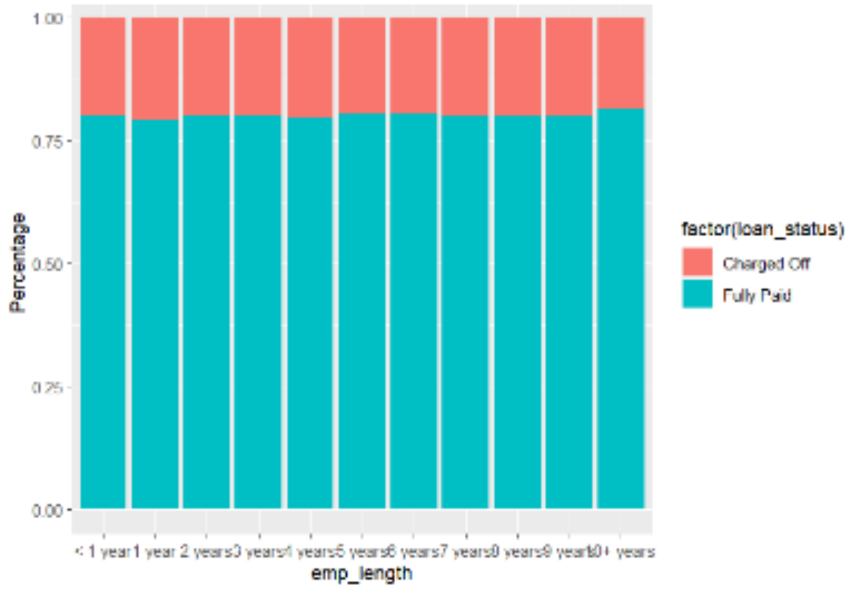


Figure 17. Emp_length and Loan_status

Grade

```
    A     B     C     D     E     F     G
23839 37261 37186 19016 10450  2968   661
```

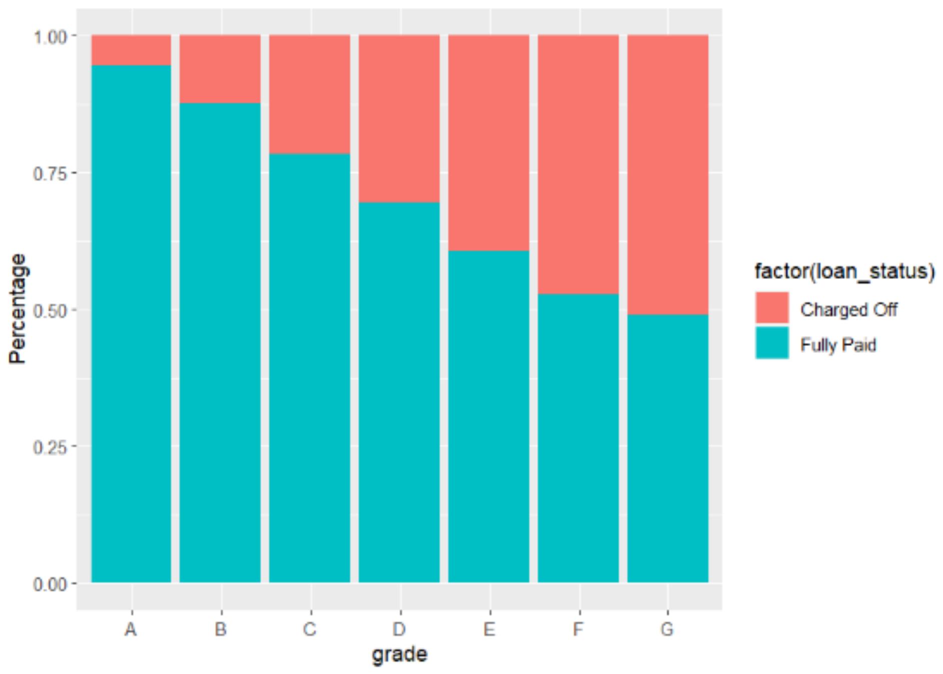


Figure 18. Grade and Loan_status

home_ownership

```
MORTGAGE       ANY       OWN      RENT
   64744         0     13422     53215
```

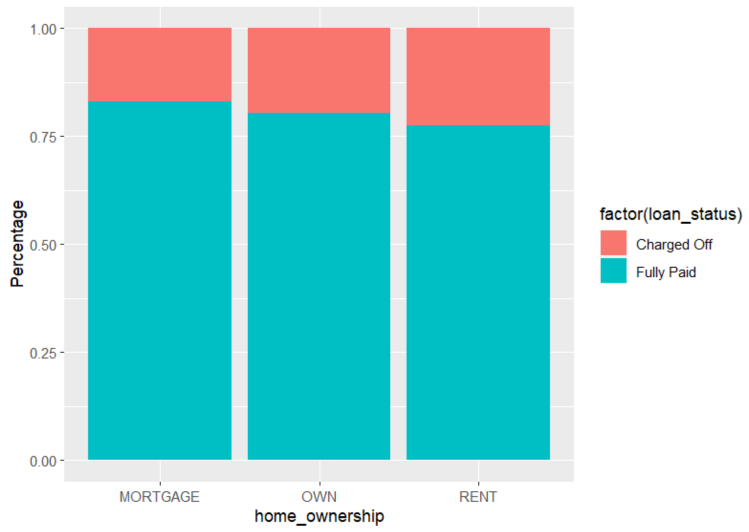


Figure 19. Home_ownership and Loan_status

purpose

```
       credit_card                car debt_consolidation   home_improvement              house
             31554               1102              78343               7801                434
    major_purchase            medical             moving              other   renewable_energy
              2284               1241                742               6005                 71
    small_business           vacation            wedding
              1088                716                  0
```

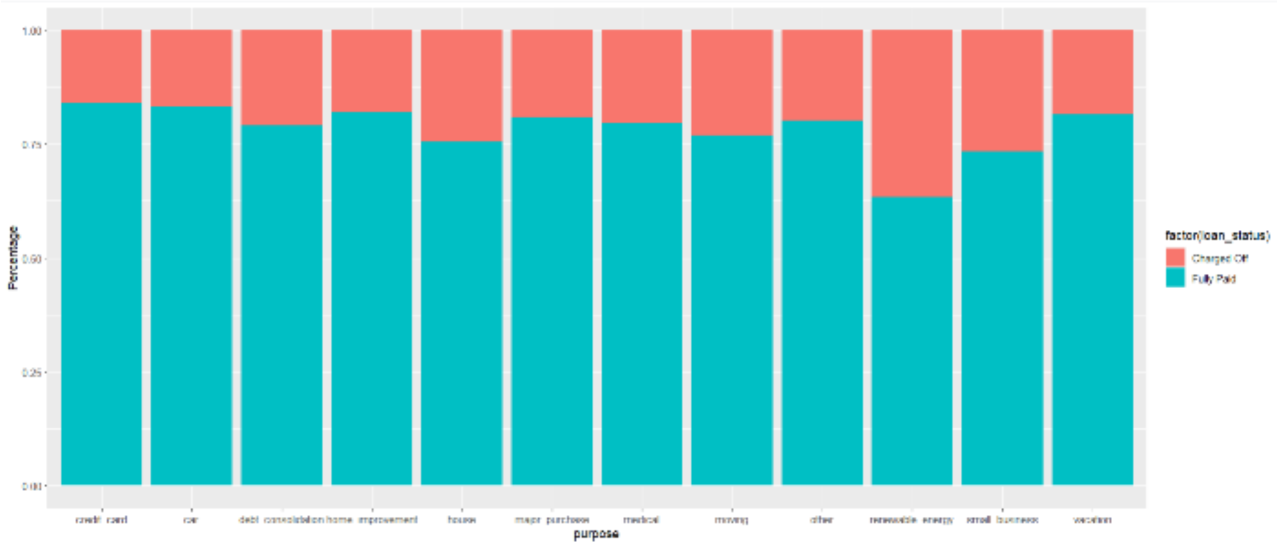
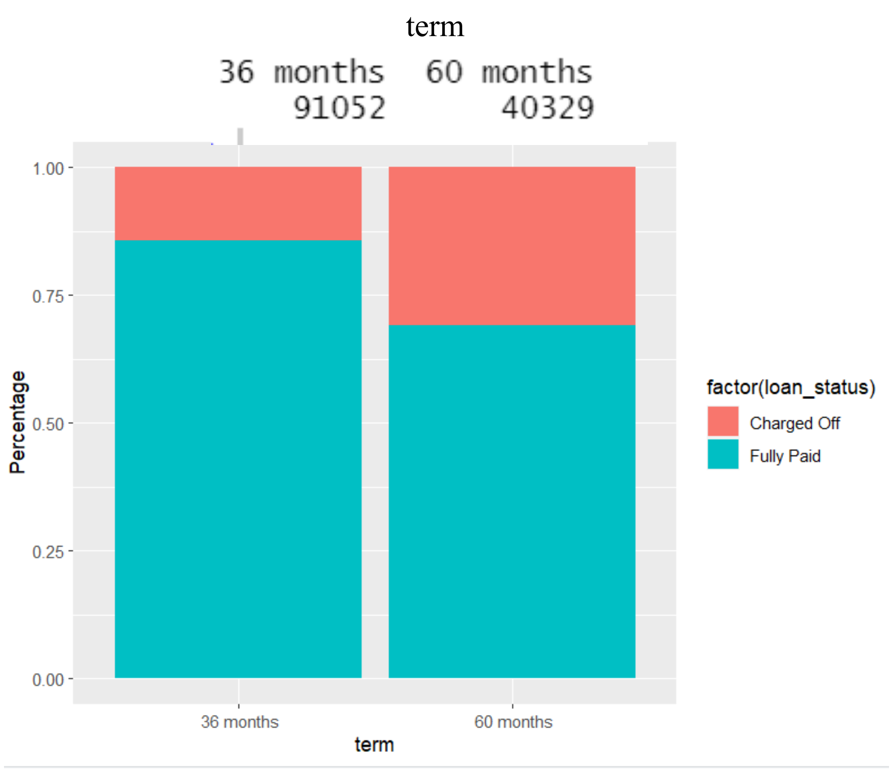


Figure 20. Purpose and Loan_status



```
term
36 months  60 months
    91052      40329
```

Figure 21. Term and Loan_status

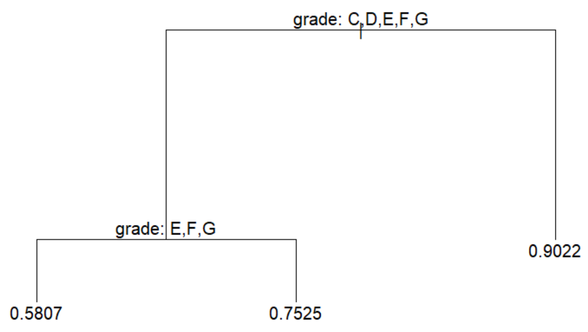


Figure 22: Simple classification Tree on the Training Set

Table 2: Confusion Matrix of Simple Classification Tree

		Prediction	
		0	1
Truth	0	0	0
	1	5011	21266

sensitivity=	0	specificity=	100%
accuracy=	0.809301	precision=	0

Table 3: Confusion Matrix of Bagging

		Prediction	
		0	1
Truth	0	404	453
	1	4607	20813

sensitivity=	0.978698	specificity=	0.080623
accuracy=	0.807436	precision=	0.818765

Table 4: Confusion Matrix of Random Forest

		Prediction	
		0	1
Truth	0	294	291
	1	4717	20975

sensitivity=	0.986316	specificity=	0.058706
accuracy=	0.809415	precision=	0.816402

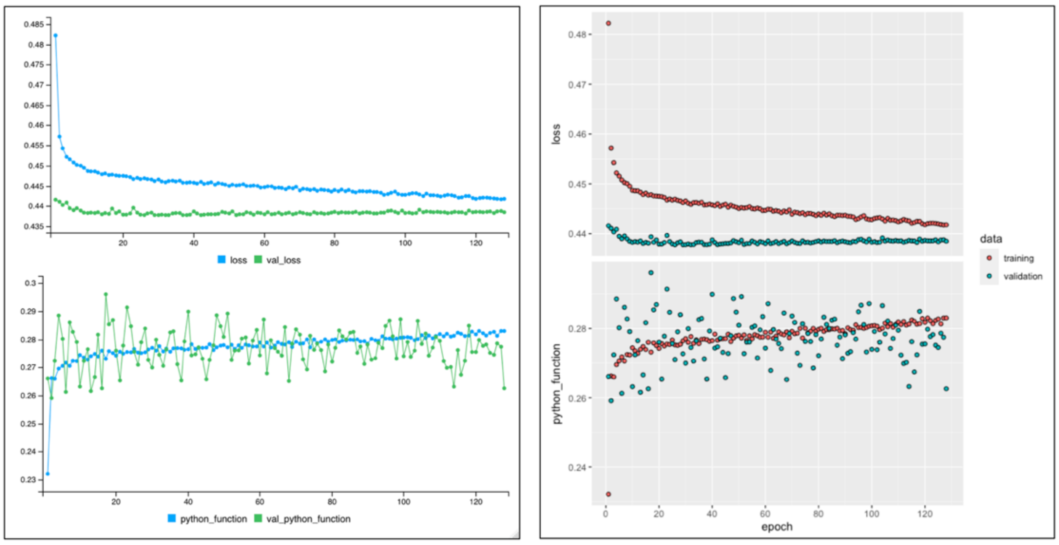


Figure 23: Training Result for Neural Network

Table 5: Error rate of the baseline model on the imbalanced training set

Pred.\True.	0	1
0	1336	3820
1	3675	17446

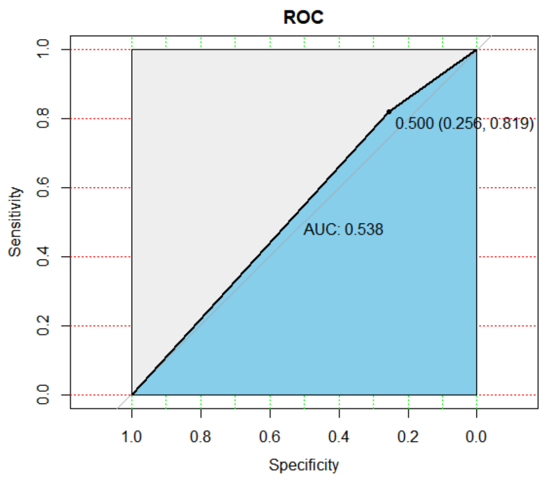


Figure 24: ROC of the baseline model on the imbalanced training set

Result of Logistic Regression with Imbalanced Training Set

Table 6: Error rate of the basic logistic model on the imbalanced training set

Pred.\True.	0	1
0	3359	7449
1	1652	13817

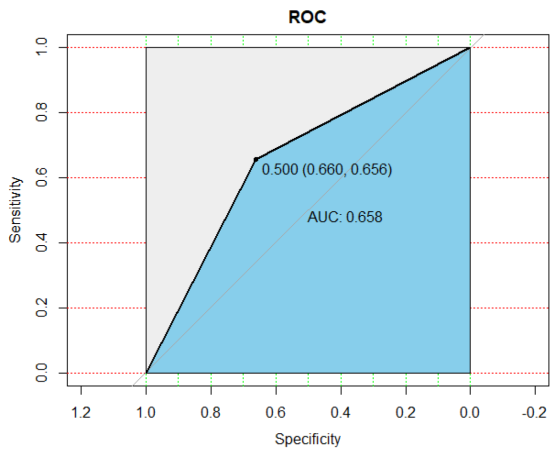


Figure 25: ROC of the basic logistic model on the imbalanced training set

Table 7: Error rate of the lasso model on the imbalanced training set

Pred.\True.	0	1
0	1838	2643
1	3393	18403

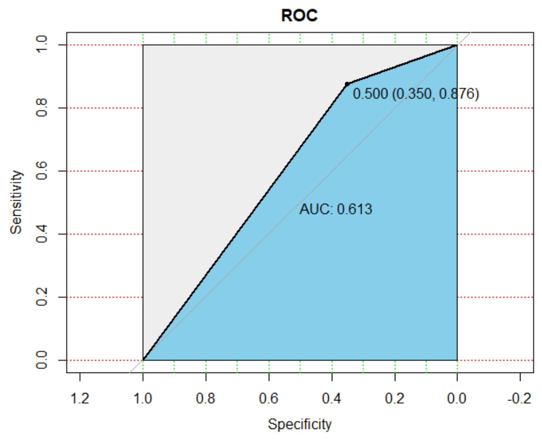


Figure 26: ROC of the lasso-logistic model on the imbalanced training set

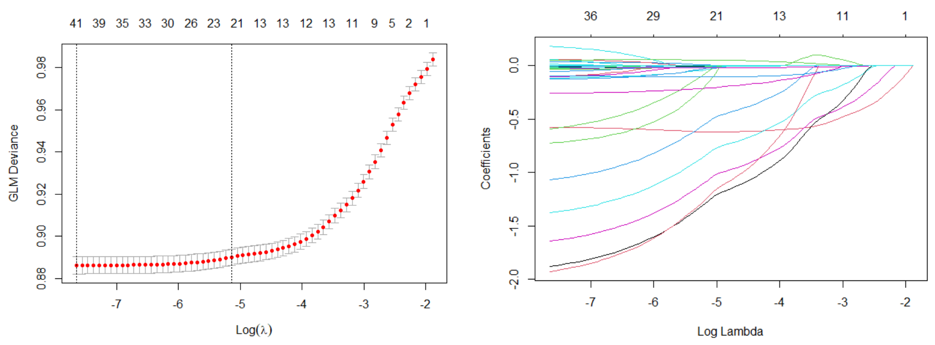


Figure 27: ROC of the lasso-logistic model with interaction terms on the imbalanced training set

Table 8: Error rate of the lasso-logistic model with interaction terms on the imbalanced training set

Pred.\True.	0	1
0	1848	2673
1	3383	18373

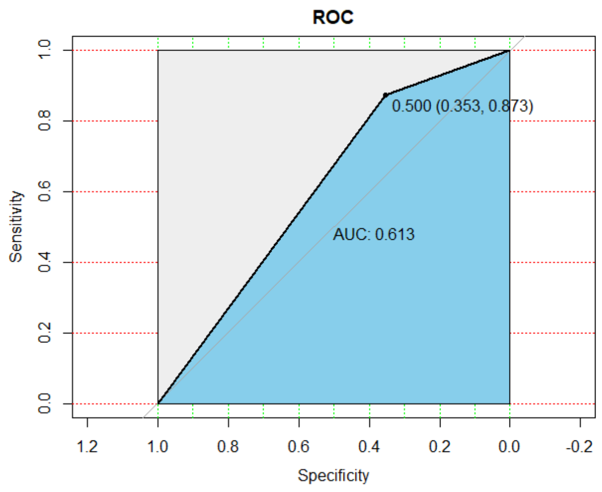


Figure 28: ROC of the lasso-logistic model with interaction terms on the imbalanced training set

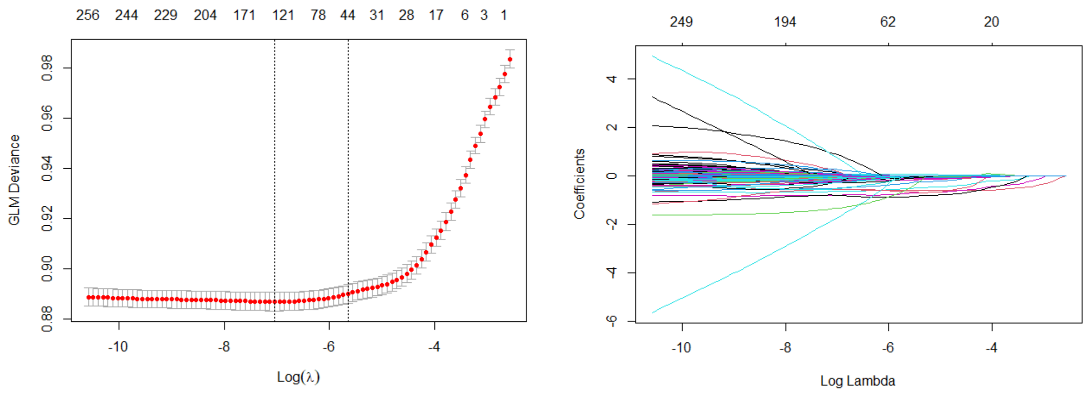


Figure 29: Lambda and shrinkage of lasso-logistic model with interaction terms on the imbalanced training set

Result of Logistic Regression with Balanced Training Set

Table 9: Error rate of the baseline model on the balanced training set

Pred.\True.	0	1
0	5623	16385
1	16308	74046

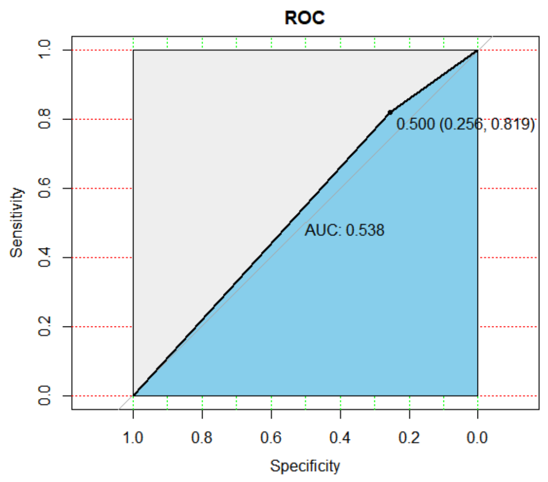


Figure 30: ROC of the baseline model on the balanced training set

Table 10: Error rate of the basic logistic model on the balanced training set

Pred.\True.	0	1
0	14288	30471
1	7643	59960

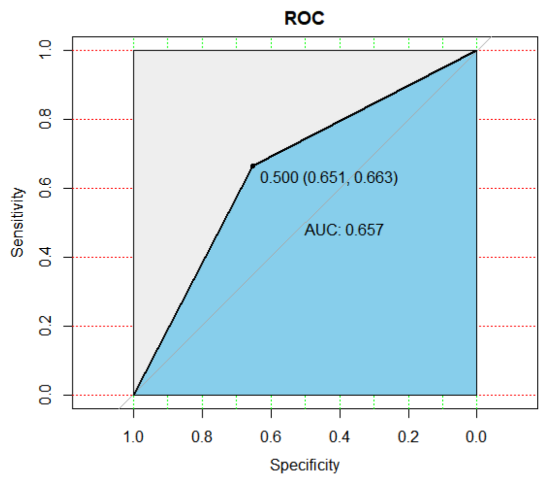


Figure 31: ROC of the basic logistic model on the balanced training set

Table 11: Error rate of the lasso-logistic model on the balanced training set

Pred.\True.	0	1
0	7018	9819
1	14913	80612

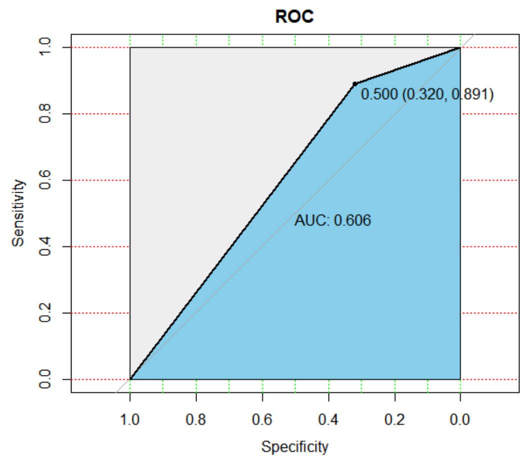


Figure 32: ROC of the lasso-logistic model on the balanced training set

Table 12: Error rate of the lasso-logistic model with interaction terms on the balanced training set

Pred.\True.	0	1
0	7039	9897
1	14892	80534

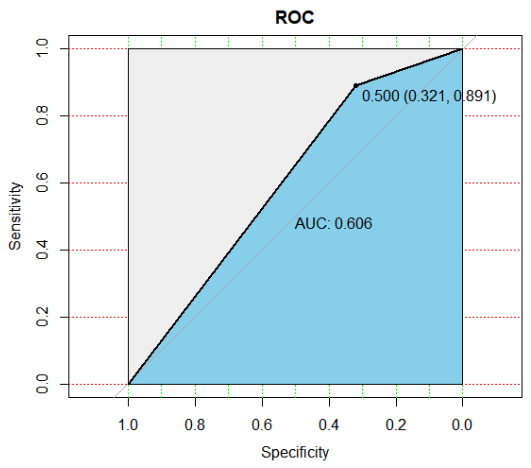


Figure 33: ROC of the lasso-logistic model with interaction terms on the balanced training set

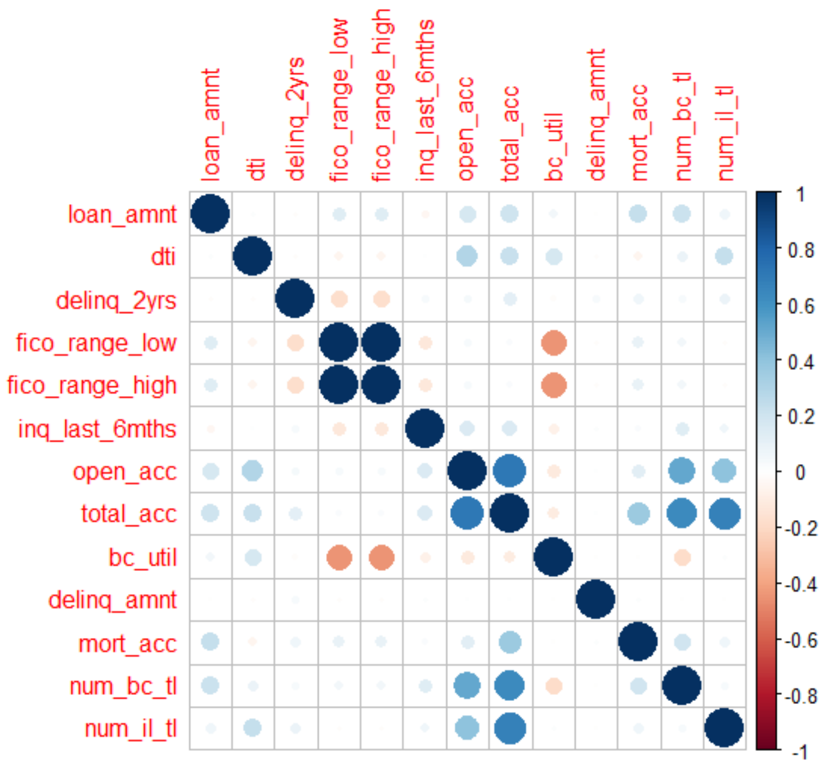


Figure 34: Multicollinearity of numeric variavles

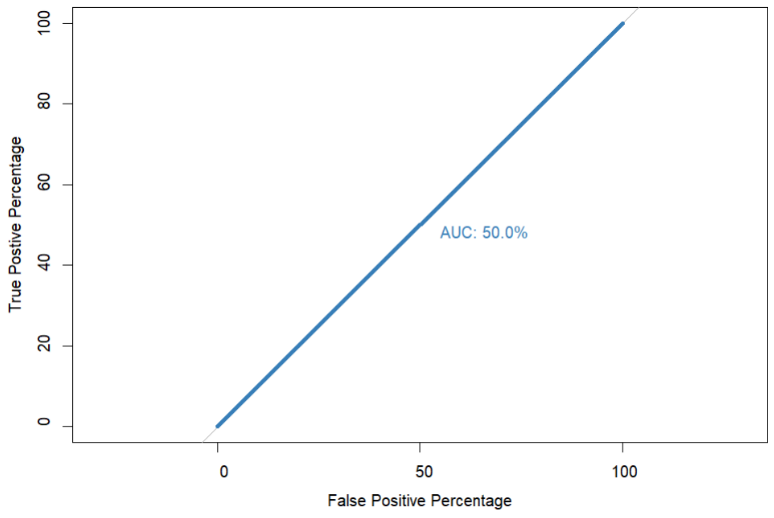


Figure 35: ROC of the Simple Tree on the imbalanced training set

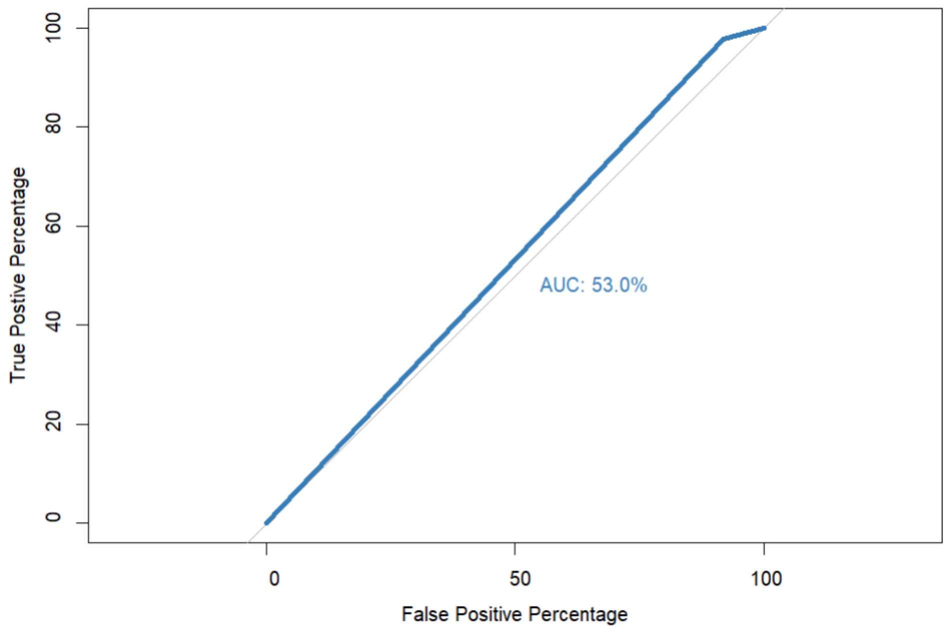


Figure 36: ROC of the Bagging Tree on the imbalanced training set

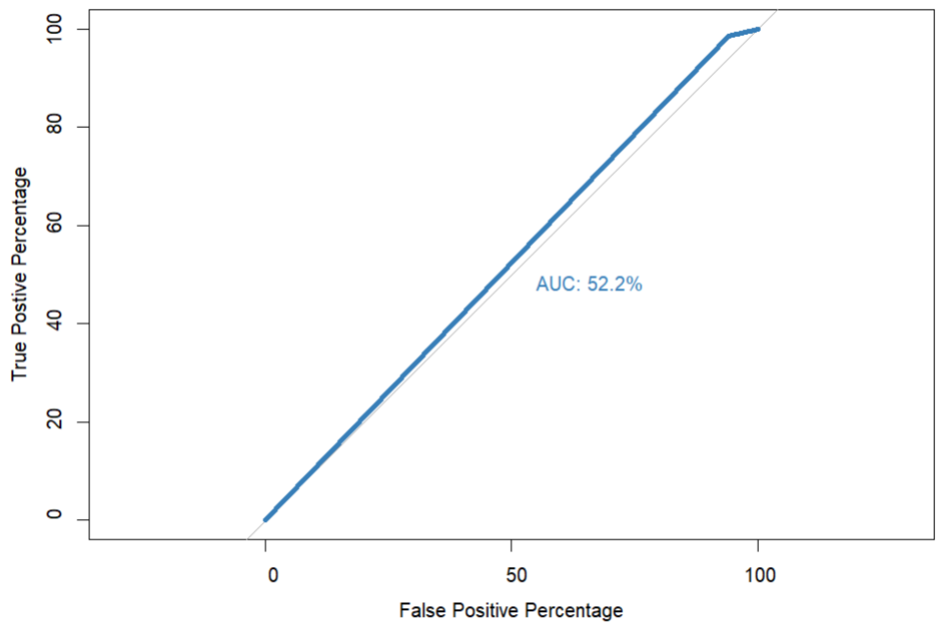


Figure 37: ROC of the Random Forest on the imbalanced training set

Table 13: Error rate of the Neural Network on the imbalanced training set

Pred.\True.	0	1
0	3326	7272
1	1685	13994

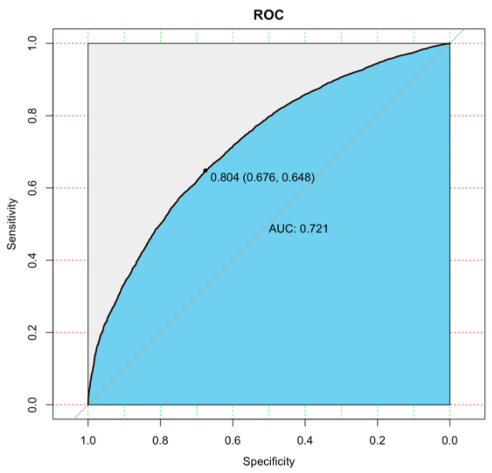


Figure 38: ROC of the Neural Network model on the imbalanced training set

```
################################################################################
######################## ECO3080 Term Project Group 9
##########################
################################################################################



######################## 1. Setup
##############################################
################################################################################
## (1) Set wd & Loading Packages
#setwd()
library(haven)
library(ggplot2)
library(keras)
library(pROC)
library('VIM')
library(glmnet)
library(corrplot)
library(tree)
library(randomForest)
## (2) import the dataset
Lending_Club <- read_dta("/Users/ZhangJingnan/Desktop/Data.dta")
## (3) Check data structure
# View(Lending_Club)
# check the type
str(Lending_Club)



########################  2. Data Cleaning
#####################################
################################################################################
## (1) Data conversion
Lending_Club$term <- as.factor(Lending_Club$term)
Lending_Club$emp_length <- factor(Lending_Club$emp_length, levels=c("< 1
year","1 year","2 years","3 years","4 years","5 years","6 years","7
years","8 years","9 years","10+ years"))
Lending_Club$grade <- factor(Lending_Club$grade, levels =
c("A","B","C","D","E","F","G"))

ggplot(Lending_Club, aes(x = grade, fill = factor(loan_status))) +
  geom_bar(position='fill', aes(y = ..count..*100/sum(..count..) ) ) +
  xlab("Grade") +
  ylab("Percentage")

Lending_Club$loan_status[Lending_Club$loan_status=="Charged Off"] = 0
Lending_Club$loan_status[Lending_Club$loan_status=="Fully Paid"] = 1
Lending_Club$loan_status <- as.numeric(Lending_Club$loan_status)
str(Lending_Club$loan_status)
Lending_Club$home_ownership <-
relevel(as.factor(Lending_Club$home_ownership), ref = "MORTGAGE")
Lending_Club$purpose <- relevel(as.factor(Lending_Club$purpose), ref =
"credit_card")

## (2) Handling missing values
apply(Lending_Club, 2, function(x){sum(is.na(x))})
```

```
aggr(Lending_Club, prop=TRUE, numbers=TRUE)
## bc_util contains 1440 missing values
# drop the missing value
Lending_Club <- Lending_Club[complete.cases(Lending_Club),]

## (3) handle extreme values
summary(Lending_Club$home_ownership)
Lending_Club <- subset(Lending_Club, home_ownership!="ANY")
summary(Lending_Club$purpose)
Lending_Club <- subset(Lending_Club, purpose!="wedding")
summary(Lending_Club$emp_title)
summary(Lending_Club$emp_length)
summary(Lending_Club$zip_code)

## (4) Remove redundant features
# drop id, emp_title, zip_code
colnames(Lending_Club) [-1]
myvariable <- c( "loan_amnt", "term", "grade", "emp_length",
"home_ownership",
                 "annual_inc", "loan_status", "dti", "delinq_2yrs",
"purpose",
                 "fico_range_low", "fico_range_high", "inq_last_6mths",
"open_acc",
                 "total_acc", "bc_util", "delinq_amnt", "mort_acc",
"num_bc_tl", "num_il_tl")
regdata <- Lending_Club[, myvariable]

## (5) Standardize for NN
x <- scale(model.matrix(loan_status ~ . - 1, data = regdata))
x <- subset(x, select = -purposewedding)
x <- subset(x, select = -home_ownershipANY)



######################## 3. Summary Statistics
#################################
################################################################################
## loan_amnt
qplot(loan_status, loan_amnt, data=Lending_Club,
      fill =loan_amnt,
      geom = c("boxplot"))
summary(regdata$loan_amnt)

## annual_inc
qplot(loan_status, annual_inc, data=Lending_Club,
      fill =annual_inc,
      geom = c("boxplot"))
summary(regdata$annual_inc)

## bc_util
qplot(loan_status, bc_util, data=Lending_Club,
      fill =bc_util,
      geom = c("boxplot"))
summary(regdata$bc_util)

##  delinq_2yrs
qplot(loan_status, delinq_2yrs, data=Lending_Club,
      fill =delinq_2yrs,
      geom = c("boxplot"))
summary(regdata$delinq_2yrs)

##  delinq_amnt
qplot(loan_status, delinq_amnt, data=Lending_Club,
      fill =delinq_amnt,
      geom = c("boxplot"))
summary(regdata$delinq_amnt)

##  dti
qplot(loan_status, dti, data=Lending_Club,
      fill =dti,
      geom = c("boxplot"))
summary(regdata$dti)

##  fico_range_high
qplot(loan_status, fico_range_high, data=Lending_Club,
      fill =fico_range_high,
      geom = c("boxplot"))
summary(regdata$fico_range_high)

##  fico_range_low
qplot(loan_status, fico_range_low, data=Lending_Club,
      fill =fico_range_low,
      geom = c("boxplot"))
summary(regdata$fico_range_low)

##  inq_last_6mths
qplot(loan_status, inq_last_6mths, data=Lending_Club,
      fill =inq_last_6mths,
      geom = c("boxplot"))
summary(regdata$inq_last_6mths)

##  mort_acc
qplot(loan_status, mort_acc, data=Lending_Club,
      fill =mort_acc,
      geom = c("boxplot"))
summary(regdata$mort_acc)

##  num_bc_tl
qplot(loan_status, num_bc_tl, data=Lending_Club,
      fill =num_bc_tl,
      geom = c("boxplot"))
summary(regdata$num_bc_tl)

##  num_il_tl
qplot(loan_status, num_il_tl, data=Lending_Club,
      fill =num_il_tl,
      geom = c("boxplot"))
summary(regdata$num_il_tl)

##  open_acc
qplot(loan_status, open_acc, data=Lending_Club,
      fill =open_acc,
      geom = c("boxplot"))
summary(regdata$open_acc)

##  total_acc
```

```
qplot(loan_status, total_acc, data=Lending_Club,
      fill =total_acc,
      geom = c("boxplot"))
summary(regdata$total_acc)

##  emp_length
ggplot(Lending_Club, aes(x = emp_length, fill = factor(loan_status))) +
  geom_bar(position='fill', aes(y = ..count..*100/sum(..count..) ) ) +
  xlab("emp_length") +
  ylab("Percentage")
summary(regdata$emp_length)

##  grade
ggplot(Lending_Club, aes(x = grade, fill = factor(loan_status))) +
  geom_bar(position='fill', aes(y = ..count..*100/sum(..count..) ) ) +
  xlab("grade") +
  ylab("Percentage")
summary(regdata$grade)

##  home_ownership
ggplot(Lending_Club, aes(x = home_ownership, fill = factor(loan_status)))
+
  geom_bar(position='fill', aes(y = ..count..*100/sum(..count..) ) ) +
  xlab("home_ownership") +
  ylab("Percentage")
summary(regdata$home_ownership)

##  purpose
ggplot(data = regdata, mapping = aes(x = home_ownership)) +
  geom_bar(color = "red", fill = "yellow")
summary(regdata$purpose)

ggplot(Lending_Club, aes(x = purpose, fill = factor(loan_status))) +
  geom_bar(position='fill', aes(y = ..count..*100/sum(..count..) ) ) +
  xlab("purpose") +
  ylab("Percentage")

##  term
ggplot(data = regdata, mapping = aes(x = term)) +
  geom_bar(color = "red", fill = "yellow")
summary(regdata$term)

ggplot(Lending_Club, aes(x = term, fill = factor(loan_status))) +
  geom_bar(position='fill', aes(y = ..count..*100/sum(..count..) ) ) +
  xlab("term") +
  ylab("Percentage")



######################## 4. Training & test set
################################
################################################################################
set.seed(114514)
train <- sample(1:nrow(regdata), nrow(regdata)*4/5)
test <- (-train)
trainset <- regdata[train, ]
testset <- regdata[test, ]
loan_status.test <- regdata$loan_status[-train]
```

```
xtest <- as.matrix(x[test, ])
xtrain <- as.matrix(x[train, ])
ytest <- as.matrix(testset$loan_status)
ytrain <- as.matrix(trainset$loan_status)



######################## 5. Baseline
###########################################
################################################################################
b_A <- sum(trainset$loan_status == '1' & trainset$grade == 'A')/
sum(trainset$grade == 'A')
b_B <- sum(trainset$loan_status == '1' & trainset$grade == 'B')/
sum(trainset$grade == 'B')
b_C <- sum(trainset$loan_status == '1' & trainset$grade == 'C')/
sum(trainset$grade == 'C')
b_D <- sum(trainset$loan_status == '1' & trainset$grade == 'D')/
sum(trainset$grade == 'D')
b_E <- sum(trainset$loan_status == '1' & trainset$grade == 'E')/
sum(trainset$grade == 'E')
b_F <- sum(trainset$loan_status == '1' & trainset$grade == 'F')/
sum(trainset$grade == 'F')
b_G <- sum(trainset$loan_status == '1' & trainset$grade == 'G')/
sum(trainset$grade == 'G')

set.seed(0002)
baseline_pred <- rep("0", 26277)
for (i in 1:26277) {
  baseline_prob = runif(1,min = 0, max = 1)
  if (testset$grade[i]=="A")
  {
    if (baseline_prob < b_A)
    {
      baseline_pred[i] <- 1
    }
  }
  if (testset$grade[i]=="B")
  {
    if (baseline_prob < b_B)
    {
      baseline_pred[i] <- 1
    }
  }
  if (testset$grade[i]=="C")
  {
    if (baseline_prob < b_C)
    {
      baseline_pred[i] <- 1
    }
  }
  if (testset$grade[i]=="D")
  {
    if (baseline_prob < b_D)
    {
      baseline_pred[i] <- 1
    }
  }
  if (testset$grade[i]=="E")
  {
    if (baseline_prob < b_E)
    {
      baseline_pred[i] <- 1
    }
  }
  if (testset$grade[i]=="F")
  {
    if (baseline_prob < b_F)
    {
      baseline_pred[i] <- 1
    }
  }
  if (testset$grade[i]=="G")
  {
    if (baseline_prob < b_G)
    {
      baseline_pred[i] <- 1
    }
  }
}
baseline_pred
table(baseline_pred, loan_status.test)


baseline_roc <- roc(regdata$loan_status[test], as.numeric(baseline_pred))
plot(baseline_roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')


#loan_status.test
#baseline_pred     0     1
#0  1336  3820
#1  3675 17446



######################## 6. Linear Method (Logit + lasso)
######################
################################################################################

######################## 6.1 pure logit (without lasso)
########################
logitreg <- glm(loan_status ~ .,
                data = regdata,
                family = binomial(link = logit),
                subset = train)
summary(logitreg)
logit_probs <- predict(logitreg, regdata[test, ], type = "response")
logit_pred <- rep(0 , 26277)
logit_pred[logit_probs > 0.8] <- 1
table(logit_pred, loan_status.test)


logit_roc <- roc(regdata$loan_status[test], as.numeric(logit_pred))
plot(logit_roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')


#loan_status.test
#logit_pred     0     1
#0  3359  7449
#1  1652 13817


######################## 6.2 logistic lasso
####################################

# check the multicollinearity
#install.packages("corrplot")
correlations <- cor(regdata[,-c(2,3,4,5,6,7,8)])
corrplot(correlations)

# lasso regression
X <- model.matrix(loan_status ~ ., regdata)
Y <- regdata$loan_status
Y.test <- Y[test]

cv.out <- cv.glmnet(X[train,], Y[train], family = binomial(link = logit),
alpha = 1)
plot(cv.out)
bestlambda <- cv.out$lambda.1se   # can also use min
bestlambda

lasso.mod <- glmnet(X, Y, family = binomial(link = logit), alpha = 1,
lambda = bestlambda)
summary(lasso.mod)
plot(lasso.mod)
lasso.mod$beta
lasso.mod$a0

lasso.pred <- predict(lasso.mod, s = bestlambda, newx = X[test,])
summary(lasso.pred)
#lasso.pred

lasso_pred <- rep(0, 26277)
lasso_pred[lasso.pred > 0.8] <- 1
table(lasso_pred, regdata$loan_status[test])

out <- glmnet(X, Y, family = binomial(link = logit), alpha = 1)
predict(out, type = "coefficients", s = bestlambda)[1:20,]

plot(cv.out$glmnet.fit, xvar="lambda")

coef(cv.out, cv.out$lambda.1se)

lasso_roc <- roc(regdata$loan_status[test], as.numeric(lasso_pred))
plot(lasso_roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')


#lasso_pred     0     1
```

```
#0  1838  2643
#1  3393 18403



######################## 6.3 net elastis regression
#############################

cv.out2 <- cv.glmnet(X[train,], Y[train], family = binomial(link = logit),
alpha = 0.5)
plot(cv.out2)
bestlambda2 <- cv.out2$lambda.1se   # can also use min
bestlambda2

lasso.mod2 <- glmnet(X, Y, family = binomial(link = logit), alpha = 0.5,
lambda = bestlambda2)
summary(lasso.mod2)
plot(lasso.mod2)
lasso.mod2$beta
lasso.mod2$a0

lasso.pred2 <- predict(lasso.mod2, s = bestlambda, newx = X[test,])
summary(lasso.pred2)
#lasso.pred

lasso_pred2 <- rep(0, 26277)
lasso_pred2[lasso.pred2 > 0.9] <- 1
table(lasso_pred2, regdata$loan_status[test])

out2 <- glmnet(X, Y, family = binomial(link = logit), alpha = 0.5)
predict(out2, type = "coefficients", s = bestlambda)[1:20,]

plot(cv.out2$glmnet.fit, xvar="lambda")

coef(cv.out2, cv.out$lambda.1se)

lasso_roc2 <- roc(regdata$loan_status[test], as.numeric(lasso_pred2))
plot(lasso_roc2, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')


#lasso_pred2     0     1
#0  1831  2618
#1  3400 18428



######################## 6.4 interaction term
##################################
X3 <- model.matrix(loan_status ~ . +
grade*loan_amnt+grade*term+grade*emp_length
                   +grade*home_ownership+grade*annual_inc+grade*purpose
                   +dti+grade*delinq_2yrs+grade*fico_range_low+grade*fico_range_high
                   +grade*inq_last_6mths+grade*open_acc+grade*total_acc+grade*bc_util
                   +grade*delinq_amnt+grade*mort_acc+grade*num_bc_tl+grade*num_il_tl,
regdata)
```

```
cv.out3 <- cv.glmnet(X3[train,], Y[train], family = binomial(link =
logit), alpha = 1)
plot(cv.out3)
bestlambda3 <- cv.out$lambda.1se   # can also use min
bestlambda3

lasso.mod3 <- glmnet(X3, Y, family = binomial(link = logit), alpha = 1,
lambda = bestlambda3)
summary(lasso.mod3)
plot(lasso.mod3)
lasso.mod3$beta
lasso.mod3$a0

lasso.pred3 <- predict(lasso.mod3, s = bestlambda3, newx = X3[test,])
summary(lasso.pred3)
#lasso.pred3

lasso_pred3 <- rep(0, 26277)
lasso_pred3[lasso.pred3 > 0.8] <- 1
table(lasso_pred3, regdata$loan_status[test])

out3 <- glmnet(X3, Y, family = binomial(link = logit), alpha = 1)
predict(out3, type = "coefficients", s = bestlambda3)[1:20,]

plot(cv.out3$glmnet.fit, xvar="lambda")

coef(cv.out3, cv.out3$lambda.1se)


lasso_roc3 <- roc(regdata$loan_status[test], as.numeric(lasso_pred3))
plot(lasso_roc3, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')



#lasso_pred3     0     1
#0  1848  2673
#1  3383 18373



####################### 7. Tree-based Methods
#################################
###############################################################################

######################## 7.1 Simple Decision Tree
##############################
TrainTree <- tree(loan_status ~ ., data=regdata, subset = train)
plot(TrainTree)
text(TrainTree, pretty = 0)

y_pred_tree <- predict(TrainTree, testset, type = "class")
table(y_pred_tree, loan_status.test)

#              loan_status.test
# y_pred_tree       Charged Off     Fully Paid
#       Charged Off           0           0
#       Fully Paid         5011       21266
#
# Sensitivity = TP/(TP+FN)=21266/(21266+0)=100%
# Specificity = TN/(TN + FP)=0
# accuracy = (TN+TP)/N=(21266+0)/(21266+5011)=0.8093009
# precision =TP/(TP+FP)=21266/(21266+5011)=0.8093009

regdata$loan_status<-relevel(as.factor(regdata$loan_status),ref="Charged
Off")
regdata$loan_status<-as.numeric(regdata$loan_status)-1

baseline_roc <- roc(loan_status.test, y_pred_tree)
plot(baseline_roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')


#### Then, consider the pruning of the tree
set.seed(114514)
cv.Loan <- cv.tree(TrainTree, FUN = prune.misclass)
cv.Loan

par(mfrow = c(1, 2))
plot(cv.Loan$size, cv.Loan$dev, type = "b")
plot(cv.Loan$k, cv.Loan$dev, type = "b")

prune.Loan <- prune.misclass(TrainTree, best = 2)

par(mfrow = c(1, 1))
plot(prune.Loan)
text(prune.Loan, pretty = 0)

Pred002 <- predict(prune.Loan, testset, type = "class")
table(Pred002, loan_status.test)

#        loan_status.test
# Pred001       Charged Off Fully Paid
# Charged Off           0         0
# Fully Paid         5011      21266
#
# Sensitivity = TP/(TP+FN)=0
# Specificity = TN/(TN + FP)=21266/(21266+0)=100%
# accuracy = (TN+TP)/N=(21266+0)/(21266+5011)=0.8093009
# precision =TP/(TP+FP)=0


######################## 7.2 Bagging
############################################
High <- as.factor(ifelse(Lending_Club$loan_status =="Fully Paid", "1",
"0"))
Data002 <- data.frame(Lending_Club, High)
myvariable2 <- c( "loan_amnt", "term", "grade", "emp_length",
"home_ownership",
                  "annual_inc", "High", "purpose", "dti", "delinq_2yrs",
                  "fico_range_low", "fico_range_high", "inq_last_6mths",
"open_acc",
                  "total_acc", "bc_util", "delinq_amnt", "mort_acc",
"num_bc_tl", "num_il_tl")

Data002 <- Data002[, myvariable2]

set.seed(114514)
bag.Data002 <- randomForest(High ~ ., data = Data002, subset = train, mtry
= 20,
                            importance = TRUE)
bag.Data002

####  make predictions on test set
Data002.test<-Data002[-train,"High"]
Data002.test<-as.character(Data002.test)
Data002.test<-as.numeric(Data002.test)
yhat.bag <- predict(bag.Data002, newdata = Data002[-train, ])
yhat.bag<-as.character(yhat.bag)
yhat.bag<-as.numeric(yhat.bag)

par(mfrow = c(1, 1))
plot(yhat.bag, Data002.test)
abline(0, 1)
mean((yhat.bag - Data002.test)^2)
# [1] 0.1925638

TN_bag <- sum((1-Data002.test)*(1-yhat.bag))
# [1] 404
TP_bag <- sum((Data002.test)*(yhat.bag))
# [1] 20813
FN_bag <- sum((Data002.test)*(1-yhat.bag))
# [1] 453
FP_bag <- sum((1-Data002.test)*(yhat.bag))
# [1] 4607

#             loan_status.test
#   y_pred           0     1
#             0     404   453
#             1    4607   20813

# Sensitivity = TP/(TP+FN)=20813/(20813+453)=0.9786984
# Specificity = TN/(TN + FP)=404/(404+4607)=0.08062263
# accuracy = (TN+TP)/N=(404+20813)/(404+20813+453+4607)=0.8074362
# precision =TP/(TP+FP)=20813/(20813+4607)=0.8187648

baseline_roc <- roc(Data002.test, yhat.bag)
plot(baseline_roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')


####  Change the number of trees
bag.Data002 <- randomForest(High ~., data = Data002, subset = train,
                            mtry = 20, ntree = 1000)
yhat.bag <- predict(bag.Data002, newdata = Data002[-train, ])
yhat.bag<-as.numeric(yhat.bag)
mean((yhat.bag - Data002.test)^2)
# [1] 1.51037 不考虑
```

```
######################## 7.3 Random Forest
#####################################
importance(rf.Data003)
varImpPlot(rf.Data003)

myvariable3 <- c( "loan_amnt", "term", "grade", "emp_length",
                  "annual_inc", "High", "purpose", "dti",
                  "fico_range_low", "fico_range_high", "open_acc",
                  "total_acc", "bc_util", "mort_acc", "num_bc_tl",
"num_il_tl")
# delete "delinq_amnt", "inq_last_6mths","home_ownership"
Data003 <- Data002[, myvariable3]

set.seed(114514)
rf.Data003 <- randomForest(High ~., data = Data003, subset = train,
                           mtry = 4, importance = TRUE)
yhat.rf <- predict(rf.Data003, newdata = Data003[-train, ])
yhat.rf<-as.character(yhat.rf)
yhat.rf<-as.numeric(yhat.rf)
Data003.test<-Data003[-train,"High"]
Data003.test<-as.character(Data003.test)
Data003.test<-as.numeric(Data003.test)

mean((yhat.rf - Data003.test)^2)
#[1] 0.1905849
#[1] 0.1905088

TN_rf <- sum((1-Data003.test)*(1-yhat.rf))
# [1] 294
TP_rf <- sum((Data003.test)*(yhat.rf))
# [1] 20977
FN_rf <- sum((Data003.test)*(1-yhat.rf))
# [1] 289
FP_rf <- sum((1-Data003.test)*(yhat.rf))
# [1] 4717

#             loan_status.test
#       y_pred       0     1
#            0     294   289
#            1    4717   20977

# Sensitivity = TP/(TP+FN)=20977/(20977+289)=0.9864102
# Specificity = TN/(TN + FP)=323/(289+4717)=0.06452257
# accuracy = (TN+TP)/N=(323+20977)/(4717+289+323+20977)=0.8097012
# precision =TP/(TP+FP)=20977/(20977+4717)=0.8164163

baseline_roc <- roc(Data003.test, yhat.rf)
plot(baseline_roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')



######################## 8. Neural Network
#####################################
################################################################################
```

```
######################## 8.1 Structure of NN
###################################
modnn <- keras_model_sequential() %>%
  layer_dense(units = 128, activation = "sigmoid", input_shape = ncol(x))
%>%
  layer_dropout(rate = 0.4) %>% ## regularization to avoid overfitting
  layer_dense(units = 128,activation = "sigmoid")%>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 1, activation = "sigmoid")

######################## 8.2 Compile Model with Criteria
#######################
specificity <- function(y_true, y_pred) {
  true_negatives <- sum((1-y_true) * (1-y_pred))
  false_positives <- sum((1-y_true) * y_pred)
  return(true_negatives / (true_negatives + false_positives))
}
### step 2. announce with method to use
modnn %>% compile(
  loss = 'binary_crossentropy',
  optimizer = 'adam',
  metrics = list(specificity)
)

######################## 8.3 Parameter Estimation
##############################
system.time(
  history <- modnn %>% fit(xtrain, ytrain,
              epochs = 128, batch_size = 512, verbose = 2,
              validation_data = list(xtest, ytest)))

plot(history, smooth = FALSE)

modnn %>% evaluate(xtest, ytest)

y_pred = modnn %>% predict(xtest)

y_pred_c <- c(y_pred)
y_test_c <- c(ytest)

rocnn <- roc(y_test_c, y_pred_c)
plot(rocnn, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col = c("green", "red"), max.auc.polygon = TRUE,
     auc.polygon.col = "skyblue", print.thres = TRUE, main = 'ROC')

y_pred <- ifelse(y_pred > 0.7, 1, 0)

true_neg <- sum((1-ytest)*(1-y_pred))
true_pos <- sum((ytest)*(y_pred))
false_neg <- sum((ytest)*(1-y_pred))
false_pos <- sum((1-ytest)*(y_pred))

accuracy <- (true_neg+true_pos)/(true_neg+true_pos+false_neg+false_pos)
sensitivity <- (true_pos)/(true_pos+false_neg)
specificity <- (true_neg)/(true_neg+false_pos)
```