STAT GR5291

Final Project

Logistic Regression in Forecasting Loan Default

Group 16 Members:

Zhenhuan Xie (zx2241), Haofeng Chen (hc2962),
Xin Xia (xx2295), Zilin Chen (zc2423),
Yilun Luo (yl4032), Jian Su (js5397),
Yiyao Hu (yh3076), Yakun Wang (yw3211),
Jiameng Liu (jl5184), Ting Cai (tc2945),
Xinze Liu (xl2822), Fuhan Ma (fm2617),
Yunxiao Zhao (yz3380), Zhichao Liu (zl2686)

Instructor: Professor Demissie Alemayehu

Dec 6th, 2019

Logistic Regression in Forecasting Loan Default

Abstract

We evaluate different binary classification models in forecasting loan default of one year accepted loan data from LendingClub.com. Various machine learning techniques are applied after extensive data pre-processing and cleaning, including classical logistic regression, logistic regression with regularization, random forest, and decision trees ensembles in XGBoost. We find, first, there is no dramatic difference among overall prediction accuracy of models for the given data, though the computational effort range differs; Several variables including "inquiries in past 6 months", "installment", and "debt to interest ratio" are the most important features marked by multiple models; Also, we discuss pros and cons, and practical issues within each model; Finally, we compare and contrast all models based on certain evaluation metrics and average running time.

1 Introduction

1.1 Background

Credit risk is one of the most recurrent, crucial and difficult areas in insurance and banking. Investors provide loans to borrowers in exchange for the promise of repayment with interest. That means they only make a profit if the borrower fully pays off the loan. Since loan default will cause a loss for the investors, detecting and predicting whether a loan will default is of significant concern to both platforms and investors in an online peer-to-peer (P2P) lending business. The best way for investors to understand how likely a particular loan is to get repaid is credit modeling. In other words, it's a tool to understand the credit of a borrower. This is especially important because this credit profile keeps changing with time and circumstances. With the help of machine learning techniques, the data itself could provide more information while modeling loan status and lead to higher prediction accuracy than manual operations nowadays.

1.2 Objective

In this project, the main objective is to apply machine learning techniques on a loan dataset from LendingClub and predicting an individual's loan status (fully paid or default) through meticulously selected variables. We will implement several models to make the prediction and compare the outputs of these models under different criteria. Eventually, we want our final selected model to be relatively simple, and with a good interpretation. Ideally, the model can help provide a valuable reference for online investors at LendingClub, or similar platforms to make better decisions dealing with investment risk.

1.3 Data Description

The dataset we use is from the LendingClub website, and we combine the data of the 4 quarters of 2018 (i.e. 2018Q1, 2018Q2, 2018Q3 and 2018Q4) as our target dataset. The raw data has around 500,000 samples, with 151 predictor variables in total and the response variable, loan status. The variables include detailed information of each individual borrowings like income, homeownership, interest rate on the loan, FICO, LC assigned loan grade & sub-grade and so on. The dataset is large, and meanwhile, it is not clean because of the nature of raw data, therefore much work is needed in data pre-processing and exploratory data analysis.

2 Data Pre-processing

There are many different factors that affect a borrower's credit situation. This makes predicting a loan's status a highly complex task. As described above, our data has 500,000 observations with 151 predictor variables, which is a large dataset. In order to have a valid learning process and to generate interpretable models, preliminary feature selection is necessary. In this project, we only inspect loan applications made by individuals, not joint applicated ones because the variables generated from joint loans are much more complicated and harder to interpret. Also, we discarded observations whose loan status is 'Current', which means they have not yet reached due date and cannot be used in supervised learning models.

2.1 Visible Features

Since the purpose of this project is to provide a reliable model predicting individual loan default for LendingClub investors, any features not visible to investors should be excluded from our model. Some of the features like recovery rate and collection rate are only relevant after the loan is issued and therefore, not available at the moment of investing. Thus, we use 'Browse Notes' from the data dictionary downloaded from LendingClub website to filter the features and keep only the matched ones. 47 features are removed in this step.

2.2 Missing Values

Missing values occur when no data value is stored for the variable in an observation. It can have a significant effect on the conclusions drawn from the data. Although some statistical models used in this project can deal with missing values, we remove the features with high proportions of missing values because it is generally hard to provide accurate predictions and interpretations. Table 2.1 provides a table of features with a missing value ratio larger than 15%. 21 features are removed in this step.

Table 2.1: Features with a missing value larger than 15%

Features	Missing Values
member_id	100.0%
desc	100.0%
annual_inc_joint	100.0%
dti_joint	100.0%
revol_bal_joint	100.0%
sec_app_fico_range_low	100.0%
sec_app_fico_range_high	100.0%
sec_app_inq_last_6mths	100.0%
sec_app_mort_acc	100.0%
sec_app_open_acc	100.0%
sec_app_revol_util	100.0%
sec_app_open_act_il	100.0%
sec_app_num_rev_accts	100.0%
sec_app_chargeoff_within_12_mths	100.0%
sec_app_collections_12_mths_ex_med	100.0%
sec_app_mths_since_last_major_derog	100.0%
mths_since_last_record	83.7%
mths_since_last_major_derog	74.9%
mths_since_recent_revol_delinq	70.4%
mths_since_last_delinq	54.9%
il_util	16.1%

2.3 Multicollinearity

Multicollinearity is a phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy. In this situation, the coefficient estimates of the multiple regression may change erratically in response to small changes in the model or the data. With multicollinearity, results and interpretations about any individual feature are unreliable. Although some regularization techniques added in this project can handle moderate collinearity like Lasso, we check the pairwise correlation in our data and arbitrarily keep one feature within any highly correlated pair (> 0.9) to ensure a reliable regression model. 12 features are removed in this step. See section 2.7 correlation matrix also.

2.4 Irrelevant Variables

Next, we take a close look through summary statistics of all remaining features and remove some to make sure all features are relevant to our model and objective. Some of the features

like 'ID' and 'URL' have nothing to do with the result of the loan status. Also, some features have only one unique value throughout all observations and put them into our model only increases computational complexity. 10 features are removed in this step.

2.5 Seasonality

Finally, since the data of 2018 is divided into 4 quarters on LendingClub.com, we are able to examine whether seasonality of loan status exists, and decide a proper size of our data (i.e. whether we should include sample from all 4 quarters, or just 1 quarter is enough). After importing and cleaning the data of 4 quarters, we have the following 2×4 table exhibiting the relationship between loan status and time:

Table 2.2: Seasonality of loan status

	2018Q1	2018Q2	2018Q3	2018Q4
Default	9,677	10,949	7,836	6,062
Fully-paid	25,730	25,227	19,757	14,658

Pearson's Chi-Square Test gives that the p-value is smaller than 2.2e-16, which means there is significant association between loan status and time in a year. Therefore, we include data of all 4 quarters of 2018, and include season as one predictor variable.

2.6 Exploratory Data Analysis

After the data cleaning process, we have a processed dataset with 61 predictor variables, 1 response variable and 119,896 observations. The 119,896 observations are then split into training data (69,048 observations) and test data. See table A.1 in the appendix for the full list of features in cleaned data. To have a better comprehension of some important predictors, we perform exploratory data analysis on selected typical features and transform the feature to make it reliable in model fitting, if necessary.

2.6.1 Response Variable

The dependent variable in the data set is 'loan_status', which is the current status of the loan. After removing all 'Current' observations, the response variable has 6 categories, as shown in Figure 2.1 below.

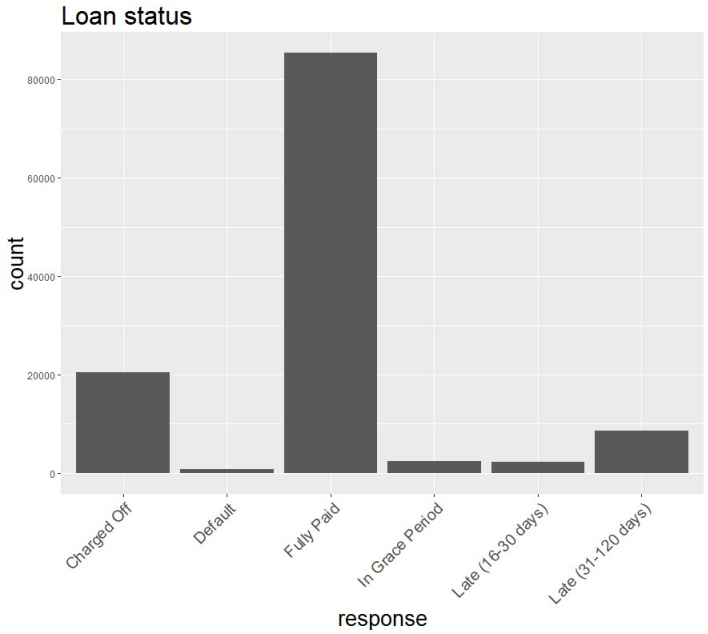

Figure 2.1: Loan Status

Since the objective of this project is to predict whether the loan is fully paid or not at the due date, we simply combine all delayed categories other than 'Fully Paid' into one class, named as 'Default'. After this procedure, the response variable has two categories, and the problem becomes a binary classification problem.

2.6.2 Annual Income

One commonly known important feature in predicting loan defaults is the borrower's income. The summary statistic is shown in Table 2.2.

Table 2.3: Annual Income Summary

Min	1^{st} Quarter	Median	Mean	3^{rd} Quarter	Max
2,500	48,000	68,000	80,540	96,064	2,300,000

The table shows that borrower's annual income ranges from 2.5 thousand to 2.3 million, with a median of 68 thousand. Because of the large range and scale, we take a log transformation of this feature. The transformation makes the data less skewed, easy to be interpreted and helps in meeting the assumptions of statistical inference.

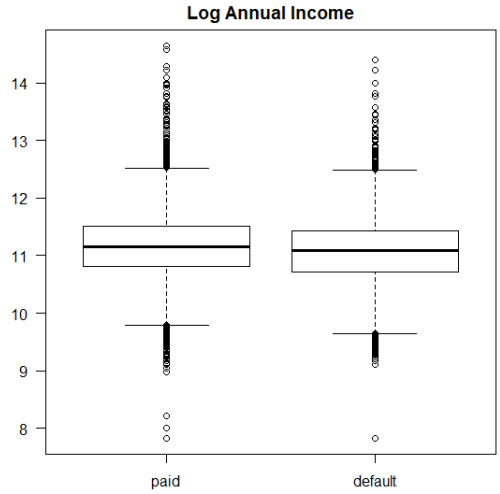

Figure 2.2: Log Annual Income

Next, to find out whether annual income is different between two different loan status groups, we apply a Welch two-sample t-test. The result $p - value = 2.8e^{-14}$ shows that the null hypothesis gets rejected and we should conclude annual income is indeed different between two groups. Figure 2.2 also shows that individuals with higher income are more likely to pay off their loans.

2.6.3 Home Ownership Status

Home ownership status might be another important feature in predicting loan status. In our dataset, this information is provided by the borrower during registration or obtained from the credit report.

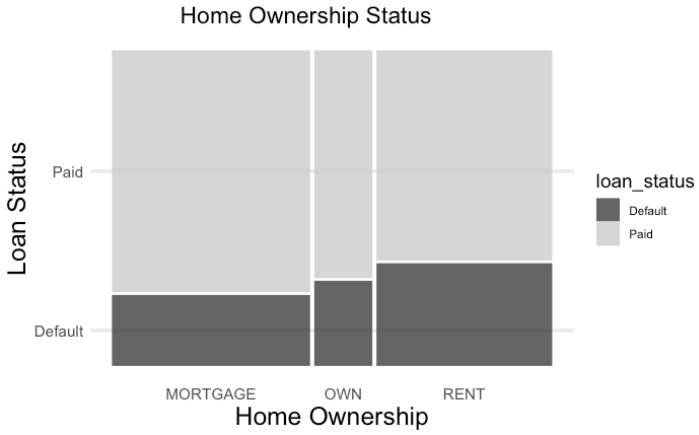

Figure 2.3: Home Owner Status

Figure 2.3 shows that homeowners are less likely to borrow money than those who mortgage or rent in 2018. In addition, people who take a mortgage have lower default rate, while home renters have higher default rate. This makes an intuitive sense because if people take out a mortgage, they have to perform well, otherwise the lender has the right to foreclose on their properties.

Table 2.4: Home Ownership Status, 2018Q1

	Default	Paid
MORTGAGE	13,227	42,836
OWN	4,490	10,988
RENT	16,770	31,484

To inspect whether there is an association between loan status and home ownership status, we use Pearson's chi-squared test. The result $p - value < 2.2e^{-16}$ shows that the null hypothesis gets rejected, so we conclude there is an association between home ownership status and loan status.

2.6.4 FICO

FICO score is a measure of individual credit risk, first introduced in 1989 by FICO, then called Fair, Isaac, and Company. The FICO model is used by the vast majority of banks and credit grantors, and is based on consumer credit files of the three national credit bureaus: Experian, Equifax, and TransUnion. The FICO score in our dataset is provided by LendingClub.com. Based on the FICO score ranges table, we convert numeric FICO scores to categorical rating scores which are fair, good, very good and exceptional. FICO score below 579 is assigned "poor", above 670 and below 739 for "good", above 740 and below 799 for "very good", and above 800 for "exceptional".

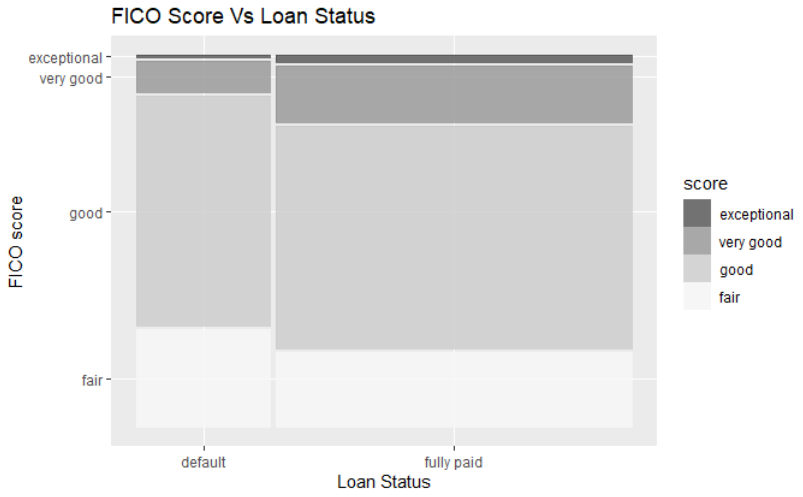

Figure 2.4: FICO Scores

From Figure 2.4 we see that people who fully paid the loan on time tend to have higher FICO scores than those people who did not fully paid the loan.

2.7 Correlation Matrix

2.7.1 Correlation Matrix for Numeric Variables

We use the Pearson correlation coefficient to graph the correlation matrix for numerical variables to detect multicollinearity. See detailed correlation matrix output in the appendix (Figure A.1). Each cell in figure is a Pearson correlation coefficient of a pair of variables. The darker the cells, the higher correlation the two variables experience. The matrix suggests weak or moderate multicollinearity exists among most predictors, which is acceptable for prediction use.

2.7.2 Correlation Matrix for Categorical Variables

For categorical variables, we use Cramér's V to measure correlations. Cramér's V (sometimes referred to as Cramér's phi and denoted as ϕ_c) is a measure of association between two nominal variables, giving a value between 0 and +1. An interesting finding is that:, interest rate, grade, and subgrade are all highly correlated with each other. From the LendingClub's website, we find out the reason why they are highly correlated: subgrade and interest rate are exactly the same variable (they have one-to-one relationship). After consideration, we decide to only keep subgrade. See detailed correlation matrix output in the appendix (Figure A.2).

3 Models & Evaluations

Given the nature of our processed dataset has a binary response variable, various machine learning techniques in statistical binary classification are suggested for our study objective. In this report, we mainly focus on four commonly used binary classification methods. They are logistic regression model, probit model, random forest, and decision trees ensembles in XGBoost.

3.1 Logistic Regression

Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. At the center of the logistic regression analysis is the task of estimating the log odds of an event. Mathematically, the logistic regression model is a generalized linear model (GLM) with

$$\text{Mean:} \quad \mu = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p.$$

$$\text{Link function:} \quad g(\mu) = log\frac{\mu}{(1-\mu)} = logit(\mu).$$

$$\text{Full model:} \quad logit(\mu) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p.$$

One advantage of logistic regression model is that it does not make many of the key assumptions of linear regression and general linear models that are based on ordinary least squares except the independence of observations. However, without further regularization techniques, the result from logistic regression is not reliable.

3.1.1 Regularization

Ridge Ridge regression shrinks the regression coefficients by imposing a penalty on model size. The ridge coefficients minimize a penalized residual sum of squares.

$$\max_{\beta_0,\beta}\{\sum_{i=1}^{N}[y_i(\beta_0 + \beta^T x_i) - log(1 + e^{\beta_0+\beta^T x_i})] - \lambda \sum_{j=1}^{p} \beta_j^2\}.$$

Ridge regularization is useful to mitigate the problem of multicollinearity, which commonly occurs in models with large numbers of parameters.

Lasso The Lasso is a shrinkage method like Ridge, with subtle but important differences. The lasso estimate is defined by:

$$\max_{\beta_0,\beta}\{\sum_{i=1}^{N}[y_i(\beta_0+\beta^T x_i)-log(1+e^{\beta_0+\beta^T x_i})]-\lambda\sum_{j=1}^{p}|\beta_j|\}.$$

Lasso can handle moderate multicollinearity and perform some variable selection. However, there is no closed-form expression for Lasso and cost for computing the lasso solution is computationally higher.

3.2 Probit Regression

There are several alternatives to logistic regression for binary responses. One of them is probit regression. The probit regression is defined by the probit link function:

$$g(\mu)=\Phi^{-1}(\mu),$$

where Φ is the cumulative distribution function (CDF) of the standard normal distribution.

Probit regression model and a logistic regression model are often similar, the result will be different only when the response is concentrated in the tail.

3.3 Random Forest

Random forest is a modification of bagging that builds a large collection of de-correlated trees and then averages them. When growing a tree, random forests select m variables at random from the total p variables which perform the job as regularization. Random forests are simpler to train and tune and are implemented in a variety of packages.

3.4 XGBoost

XGBoost (eXtreme Gradient Boosting), is an optimized distributed gradient boosting library using the decision tree ensembles model. Similar to the random forest, it is also an ensemble learning method and predicts by combining the outputs from individual trees. Rather than training each tree independently, gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction, based on a gradient descent algorithm.

3.5 Evaluation Metrics

3.5.1 Confusion Matrix

To evaluate the performance of our models, we introduce several commonly used metrics in statistical machine learning. One is the confusion matrix, also known as an error matrix, which is a specific table layout that allows visualization of the performance of a model.

Table 3.1: Confusion Matrix

	True Class	
Predicted Class	0	1
0	TN	FN
1	FP	TP

The confusion matrix has two rows and two columns that report the number of false positives, false negatives, true positives, and true negatives. See table 3.1. This allows a more detailed analysis than mere proportion of correct classifications or overall accuracy. In our study, we want the overall accuracy to be high with a lower false-positive rate (FPR = FP / N) since the cost of classifying a defaulted loan to a riskless one to investors is generally high.

3.5.2 ROC & AUC

ROC curve, or known as receiver operating characteristic curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR, or sensitivity) against the false positive rate (FPR, or 1 - specificity) at various threshold settings. With a ROC space, we can evaluate model performance by curve above the diagonal represent good classification results (better than random) and curve below the line represent bad results (worse than random). See Figure 3.1.

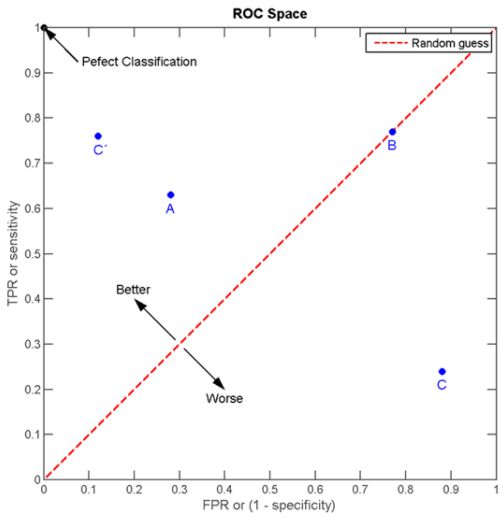

Figure 3.1: ROC Curve

The area under the ROC curve, or AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. It is often used together with ROC in statistical machine learning for model comparison. Furthermore, ROC curve can also help decide an optimal threshold (with the highest specificity + sensitivity) for 0-1 prediction.

4 Analysis of Results

4.1 SMOTE

A dataset is imbalanced if the classes are not approximately equally represented. For example, in our loan status prediction, most clients can fully pay their loans on time, so default is the minority class. When the data is imbalanced and/or the costs of different errors vary markedly, the evaluation of machine learning algorithms using predictive accuracy is not appropriate. Over years, the machine learning community has come up with techniques including over-sampling and under-sampling trying to deal with the negative effects caused by data imbalance.

Synthetic Minority Over-sampling Technique (SMOTE) algorithm, is an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples, rather than by over-sampling with replacement, which doesn't significantly improve minority class recognition, according to previous research. In this algorithm, the minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen (See the pseudo code of this algorithm in the appendix Figure A.3). SMOTE can also be further extended to be combined with under-sampling of the majority class, which is the case of its application in our project.

In preliminary steps (before our presentation), we find that the prediction is "biased" towards the majority class if we use imbalanced training data: it tends to predict most cases as the majority class, which is problematic in our case, because we can allow for some error in the majority class (fully paid), while require a reasonably high accuracy in identifying the minority class (default). In other words, we require that the specificity (correctly identifying default) should not be too low.

Table 4.1: Preliminary model fitting results using imbalanced training data

Model	Specificity	Sensitivity	Overall Accuracy
Logistic Regression	0.218	0.938	0.741
Probit Regression	0.207	0.940	0.740
L1 Logistic Regression	0.454	0.820	0.720
L2 Logistic Regression	0.451	0.822	0.721

Clearly in Table 4.1, no matter what model we use, the prediction results cannot meet our needs. Therefore, we apply SMOTE algorithm to our data, and all the following analysis are conducted based on the processed balanced data.

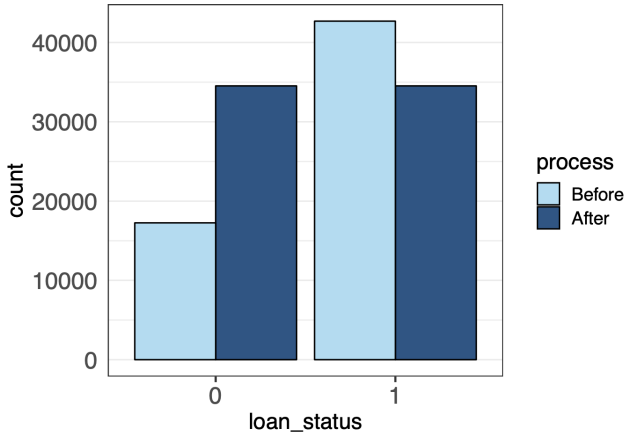

Figure 4.1: Training Data Before & After SMOTE

For details in preliminary model training and data processing, please refer to the corresponding contents in appendix. In the following analysis, you will see that using balanced data proves to be helpful in improving specificity.

Since there are quite a few available predictors, we present the formula (coefficients) of all the models in the appendix part, and we only present and compare the goodness of fit of models here. Goodness of fit is measured by confusion matrix and AUC (area under the Receiver Operating Characteristic curve).

4.2 Logistic Regression

For this classical 0-1 binary classification problem, the first model that comes to mind is the simple logistic regression model. In our implementation, we include in the model all the predictors variables that we decide to keep after data processing. This model has the performance described by the following confusion matrix and ROC curve:

Table 4.2: Confusion Matrix of Simple Logistic Regression

	True Class	
Predicted Class	0	1
0	12,318	16,415
1	4,944	26,271

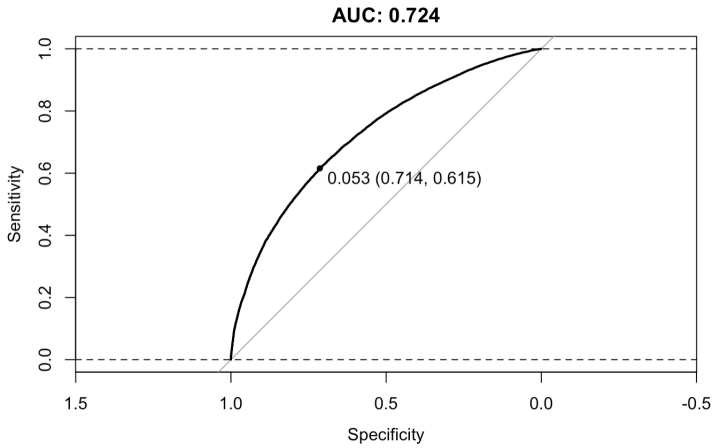

Figure 4.2: ROC Curve of Simple Logistic Regression

4.3 Lasso & Ridge Regression

The logistic regression model can then be further extended to L1 regularized and L2 regularized models. The reason we consider these two models is that, considering the number of available predictors and the size of data, stepwise feature selection procedure is too cumbersome, and thus L1 and L2 regularized models are more appropriate.

In the L1 regularized logistic model, the best regularization parameter λ (with minimum mean cross-validated error) is chosen to be 0.000267. The model gives the following confusion matrix and ROC curve:

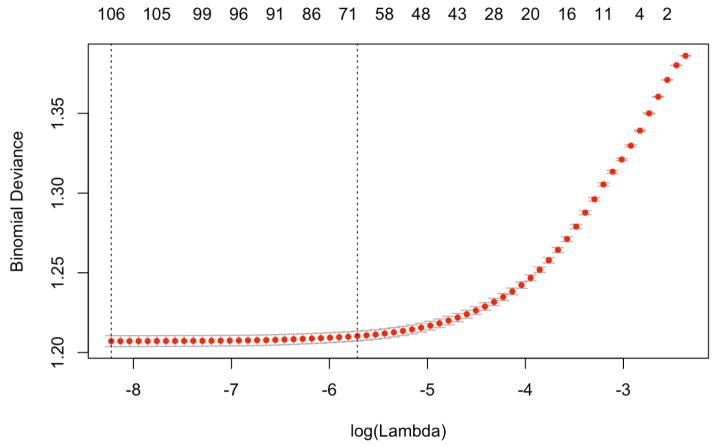

Figure 4.3: Cross Validation Plot of L1 Regularized Logistic Regression Model

Table 4.3: Confusion Matrix of L1 Regularized Logistic Regression Model

	True Class	
Predicted Class	0	1
0	12,182	16,096
1	5,080	26,590

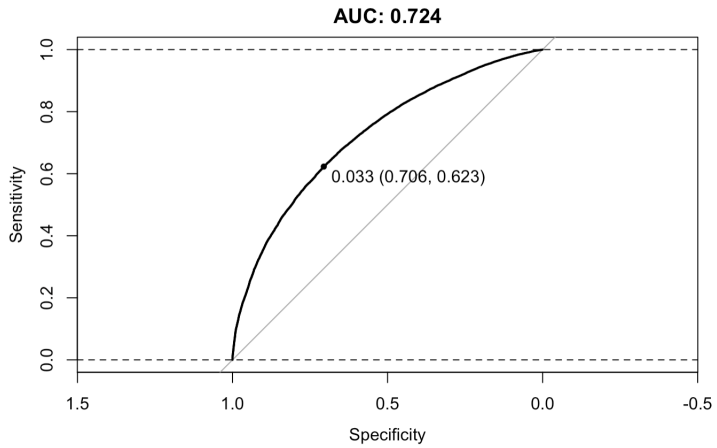

Figure 4.4: ROC Curve of L1 Regularized Logistic Regression Model

In the L2 regularized logistic model, the best λ is chosen to be 0.0103. The model gives the following confusion matrix and ROC curve:

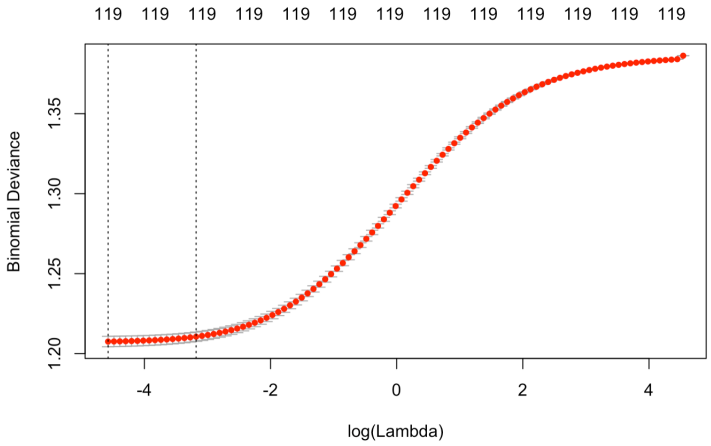

Figure 4.5: Cross Validation Plot of L2 Regularized Logistic Regression Model

Table 4.4: Confusion Matrix of L2 Regularized Logistic Regression Model

	True Class	
Predicted Class	0	1
0	11,799	15,118
1	5,463	27,568

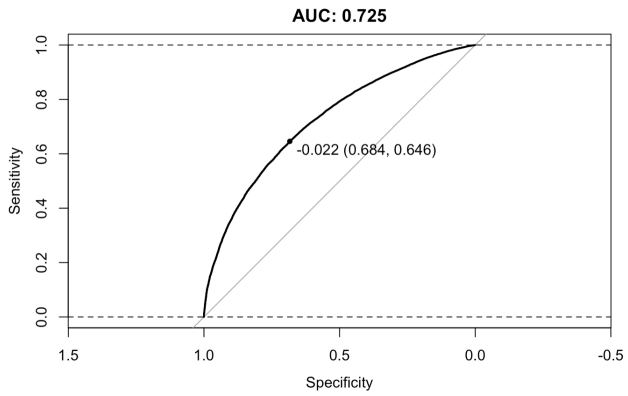

Figure 4.6: ROC Curve of L2 Regularized Logistic Regression Model

By inspecting the coefficients of L1 and L2 regularized logistic regression models, we notice that the two models actually behave similarly. For the predictors whose coefficients are shrunk to 0 in L1 model, they also have a correspondingly small coefficient in the L2 model: all of them have magnitude below 10^{-2}. More generally, the coefficients of all predictors in the two models have a high correlation of 0.995. This can help explain why the two models give similar predictions.

Table 4.5: Lasso and Ridge Coefficient comparison

Variable	open_acc	pub_rec	revol_bal	tot_cur_bal	total_bal_il	chargeoff_within_12_mths	mo_sin_rcnt_tl
LASSO	Dropped (coefficient=0)						
Ridge	-1.218320e-03	1.833899e-02	8.420496e-03	2.487521e-02	8.059681e-04	9.213951e-03	8.306611e-05
Variable	num_bc_sats	num_il_tl	pct_tl_nvr_dlq	tax_liens	total_bc_limit	purpose_vacation	
LASSO	Dropped (coefficient=0)						
Ridge	-2.343662e-04	8.832774e-03	2.620544e-02	-1.834383e-02	6.882862e-03	2.202742e-02	

For comparison purpose, we also consider other types of models.

4.4 Probit Regression

Probit regression is another member in the generalized linear model family. We are taught in lecture that this model gives similar results as logistic regression for most of the time, and thus we are interested in whether this is still the case in our task, and it turns out that it indeed is: Not only are their confusion matrix and ROC curve similar, but they also give 99.7% same 0-1 prediction result.

Table 4.6: Confusion Matrix of Probit Regression Model

	True Class	
Predicted Class	0	1
0	12,289	16,355
1	4,973	26,331

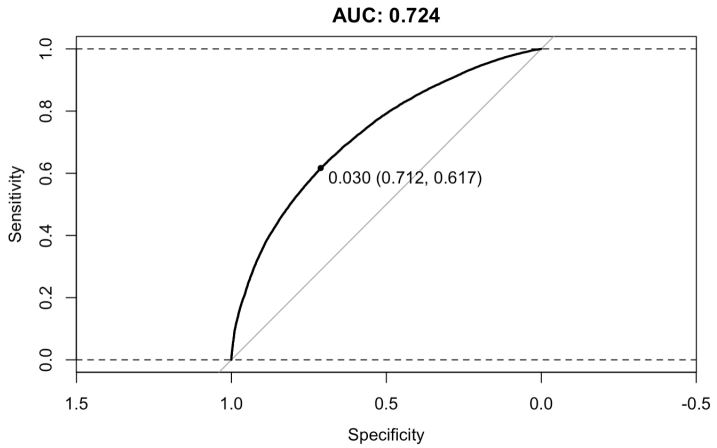

Figure 4.7: ROC Curve of Probit Regression Model

4.5 Random Forest

Another model is random forest. This model has one available tuning parameter: mtry, which is the number of variables randomly sampled as candidates at each split. By cross validation, the best parameter is chosen to be 60.

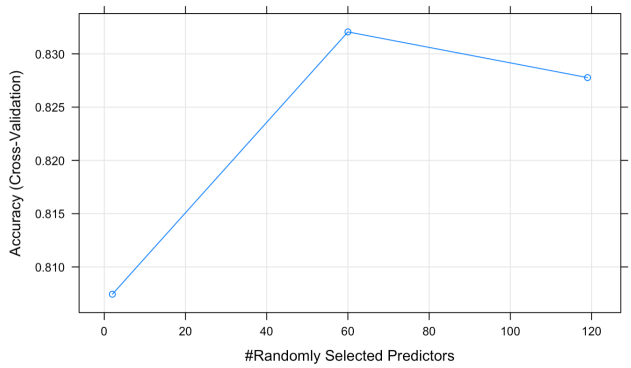

Figure 4.8: Tuning Parameter "mtry" of Random Forest Model

Although they have a totally different mechanism, the random forest model still gives a relatively similar result as logistic regression.

Table 4.7: Confusion Matrix of Random Forest Model

	True Class	
Predicted Class	0	1
0	11,937	16,254
1	5,325	26,432

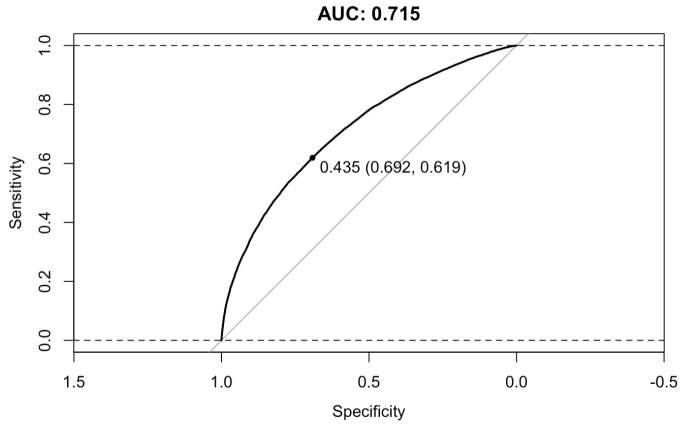

Figure 4.9: ROC Curve of Random Forest Model

One good thing about the random forest model is that it can evaluate feature importance, by computing how much each feature decreases the weighted impurity in a tree and then taking average among all the trees in the forest. The features are ranked according to this measure. In our random forest model, the feature importance is given in Figure 4.10.

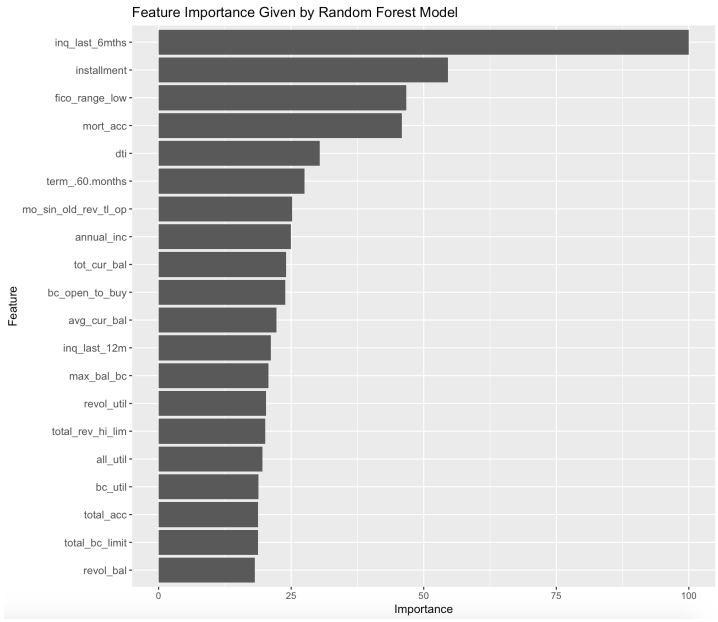

Figure 4.10: Feature Importance Given by Random Forest Model

In future work, we can consider fitting a more parsimonious model using the most important features and dropping unimportant ones.

4.6 XGBoost

The last model we try is XGBoost. For this model, there are several parameters can be tuned, which will be explained in the order of the actual tuning process.

booster the booster type to use. For our classification problem, we can use gbtree or dart. Compared with the default booster gbtree, dart (Dropouts meet Multiple Additive Regression Trees) costs longer training time but holds a higher True Negative rate. Therefore, the booster is chosen to be dart.

max_depth and gamma the max depth of the tree and regularization degree. It's better to tune them together for their high dependence. Larger the depth, more complex the model and higher chances of overfitting, while higher the regularization, heavier penalization on coefficients which don't improve the model's performance and lower chances of overfitting. By cross-validation, the best two combinations are max_depth = 6, gamma = 2, max_depth = 4, gamma = 3. The latter one is chosen finally due to a trade-off with relatively lower AUC but higher Specificity.

subsample the number of observations supplied to a tree. Its optimal value is 0.85.

colsample_bytree the number of features supplied to a tree. Its optimal value is 0.9.

min_child_weight the minimum sum of instance weight to stop splitting tree. It blocks the potential feature interactions to prevent overfitting. Its optimal value is 2.

sample_type type of the sampling algorithm to drop trees. It's tuned from uniform to weighted, which selects dropped trees in proportion to weight rather than uniformly.

eta learning rate. Balancing the speed and accuracy of the model, its optimal value is 0.15.

early_stopping_rounds the maximum acceptable rounds without improvement on the accuracy of validation. Due to the randomness of the booster dart, a relatively larger value 30 is chosen.

The result has the highest Specificity and similar AUC compared with all other models, shown in Table 4.8 and Figure 4.11.

Table 4.8: Confusion Matrix of XGBoost Model

	True Class	
Predicted Class	0	1
0	12,593	17,165
1	4,699	25,521

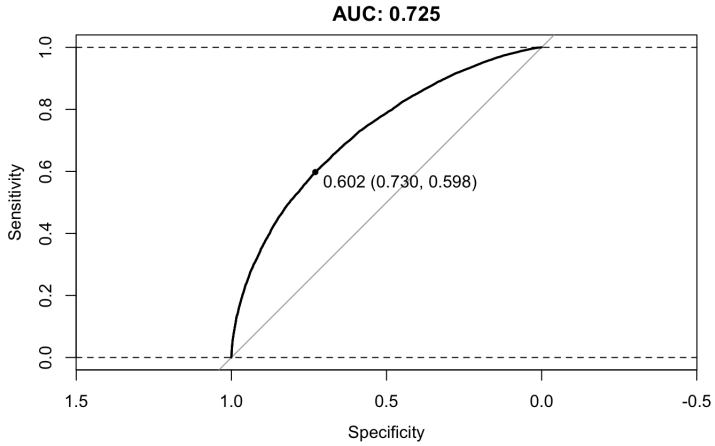

Figure 4.11: ROC Curve of XGBoost Model

XGBoost model can also evaluate feature importance. The importance sequence based on XGBoost has a few differences with that given by random forest. The top 20 ranked features are given in Figure 4.12.

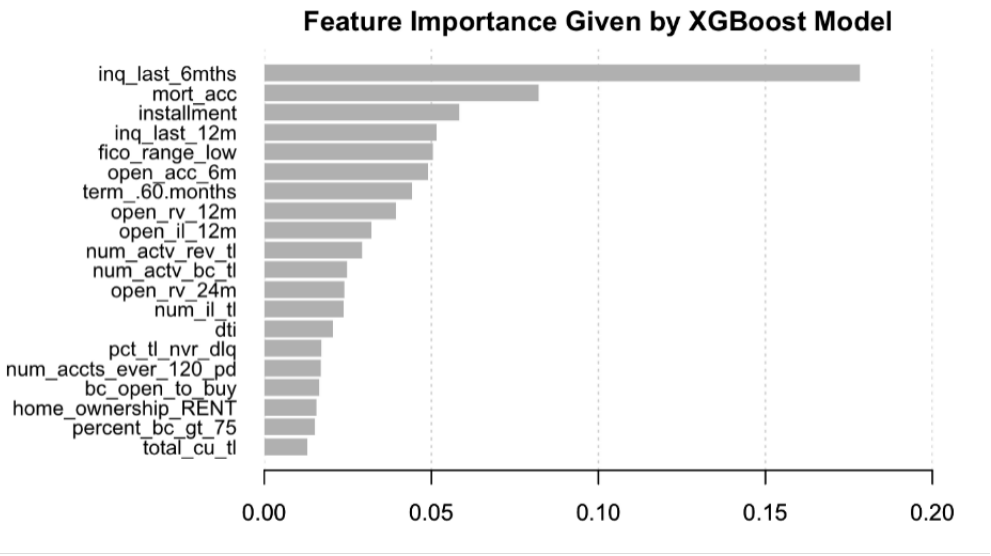

Figure 4.12: Feature Importance Given by XGBoost Model

In general, the different models seem to perform comparably well in our loan prediction task, with the AUC of all models being close and XGBoost having the (slightly) best specificity. In such case, what model would we prefer? We need to take into consideration the time it takes to train these models.

Table 4.9: Training Time of Different Models

Model	Logistic Regression	L1 Logistic Regression	L2 Logistic Regression	Probit Regression	Random Forest	XGBoost
Time	15 seconds	1.7 minutes	2.9 minutes	15 seconds	7.5 hours	4.7 minutes

We see that random forest model takes too long to train, and it is also not easy to interpret, so it may not be feasible in terms of further analysis. XGBoost is fast in each single run, while it requires much more effort in tuning parameters in order to achieve comparable (and slightly better) performance. Nevertheless, the feature importance function of random forest and XGBoost can serve as a guide on feature selection in preliminary steps.

Considering its relatively good prediction performance, fast training speed and simplicity in interpretation, Logistic regression, together with L1 and L2 regularized logistic regression, is appropriate enough to deal with our task.

5 Conclusion

In this project, we propose a brand new raw dataset, define our research problem to be the popular topic: loan default prediction, and successfully apply what we have learned in this class to work out this problem. During this process, we have interesting findings about the dataset through data processing and EDA; we try to fit different models to the data and evaluate their goodness of fit, cost of training the model, and come up with suggestions and comments on different models.

The obtained results from extreme gradient boosting(XGBoost) model outperforms other algorithms with AUC of 0.725 and the highest specificity, though its edge is not huge. To this end, we may consider using XGBoost as the method of preliminary research, because it can provide a feature importance plot, which is useful when we would like to select most important features to fit a parsimonious model and still keep the goodness of fit at approximately the same level. Meanwhile, the result also tells us that, logistic regression, though a simple and traditional generalized linear model, is still not out of date. It has its special properties to remain a classic and popular solution to binary classification problems: It has relatively simple functional form (and thus easy to train), easy-to-interpret coefficients, and doesn't make assumptions on data that are too restrictive. In our case, even with a bunch of predictor variables, it can give reasonably good prediction result (even a little better than more complex model such as random forest) within seconds.

Last but not least, the phenomenon that all models have similar performance may indicate that they are approaching the prediction accuracy limit of available predictor variables, and they can extract useful information contained in data in a equally good manner.

References

[1] Fred L. Ramsey, Daniel W. Schafer. The Statistical Sleuth: A Course in Methods of Data Analysis, Third Edition, Oregon State University. Brooks/Cole, Cengage Learning, 2013. Print.

[2] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, An Introduction to Statistical Learning, Springer, 2013. Print.

[3] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.

[4] Fawcett, Tom (2006). "An Introduction to ROC Analysis" (PDF). Pattern Recognition Letters. 27 (8): 861–874. doi:10.1016/j.patrec.2005.10.010.

[5] Chen, Tianqi; Guestrin, Carlos (2016). "XGBoost: A Scalable Tree Boosting System". In Krishnapuram, Balaji; Shah, Mohak; Smola, Alexander J.; Aggarwal, Charu C.; Shen, Dou; Rastogi, Rajeev (eds.). Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. ACM. pp. 785–794. arXiv:1603.02754. doi:10.1145/2939672.2939785.

[6] Breiman L (2001). "Random Forests". Machine Learning. 45 (1): 5–32. doi:10.1023/A:1010933404324.

[7] Oxford English Dictionary, 3rd ed. s.v. probit (article dated June 2007): Bliss, C. I. (1934). "The Method of Probits". Science. 79 (2037): 38–39. doi:10.1126/science.79.2037.38. PMID 17813446.

[8] Sasaki, Y. (2007). "The truth of the F-measure" (PDF). Teach Tutor mater 1 (5), 1-5.

[9] "How FICO became 'the' credit score". finance.yahoo.com. 2013-12-12. Retrieved 2018-08-25.

A Tables & Results

Table A.1: Selected Features

LoanStatNew	Description
acc_now_delinq	The number of accounts on which the borrower is now delinquent
acc_open_past_24mths	Number of trades opened in past 24 months
all_util	Balance to credit limit on all trades
annual_inc	The self-reported annual income provided by the borrower during registration
avg_cur_bal	Average current balance of all accounts
bc_open_to_buy	Total open to buy on revolving bankcards
bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts
chargeoff_within_12_mths	Number of charge-offs within 12 months
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years
fico_range_low	The lower boundary range the borrower's FICO at loan origination belongs to
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
initial_list_status	The initial listing status of the loan. Possible values are: W, F
inq_fi	Number of personal finance inquiries
inq_last_12m	Number of credit inquiries in past 12 months
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
installment	The monthly payment owed by the borrower if the loan originates
loan_status	Current status of the loan
max_bal_bc	Maximum current balance owed on all revolving accounts
mo_sin_old_rev_tl_op	Months since oldest revolving account opened
mo_sin_rcnt_rev_tl_op	Months since most recent revolving account opened
mo_sin_rcnt_tl	Months since most recent account opened
mort_acc	Number of mortgage accounts
mths_since_rcnt_il	Months since most recent installment accounts opened
mths_since_recent_bc	Months since most recent bankcard account opened
num_accts_ever_120_pd	Number of accounts ever 120 or more days past due
num_actv_bc_tl	Number of currently active bankcard accounts
num_actv_rev_tl	Number of currently active revolving trades
num_bc_sats	Number of satisfactory bankcard accounts

num_bc_tl	Number of bankcard accounts
num_il_tl	Number of installment accounts
num_op_rev_tl	Number of open revolving accounts
num_rev_accts	Number of revolving accounts
num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months
num_tl_op_past_12m	Number of accounts opened in past 12 months
open_acc	The number of open credit lines in the borrower's credit file
open_acc_6m	Number of open trades in last 6 months
open_il_12m	Number of installment accounts opened in past 12 months
open_il_24m	Number of installment accounts opened in past 24 months
open_act_il	Number of currently active installment trades
open_rv_12m	Number of revolving trades opened in past 12 months
open_rv_24m	Number of revolving trades opened in past 24 months
pct_tl_nvr_dlq	Percent of trades never delinquent
percent_bc_gt_75	Percentage of all bankcard accounts ¿ 75% of limit
pub_rec	Number of derogatory public records
pub_rec_bankruptcies	Number of public record bankruptcies
purpose	A category provided by the borrower for the loan request
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit
sub_grade	LC assigned loan subgrade
tax_liens	Number of tax liens
term	The number of payments on the loan. Values are in months and can be either 36 or 60
tot_coll_amt	Total collection amounts ever owed
tot_cur_bal	Total current balance of all accounts
total_acc	The total number of credit lines currently in the borrower's credit file
total_bal_il	Total current balance of all installment accounts
total_bc_limit	Total bankcard high credit/credit limit
total_cu_tl	Number of finance trades
total_il_high_credit_limit	Total installment high credit/credit limit
total_rev_hi_lim	Total revolving high credit/credit limit

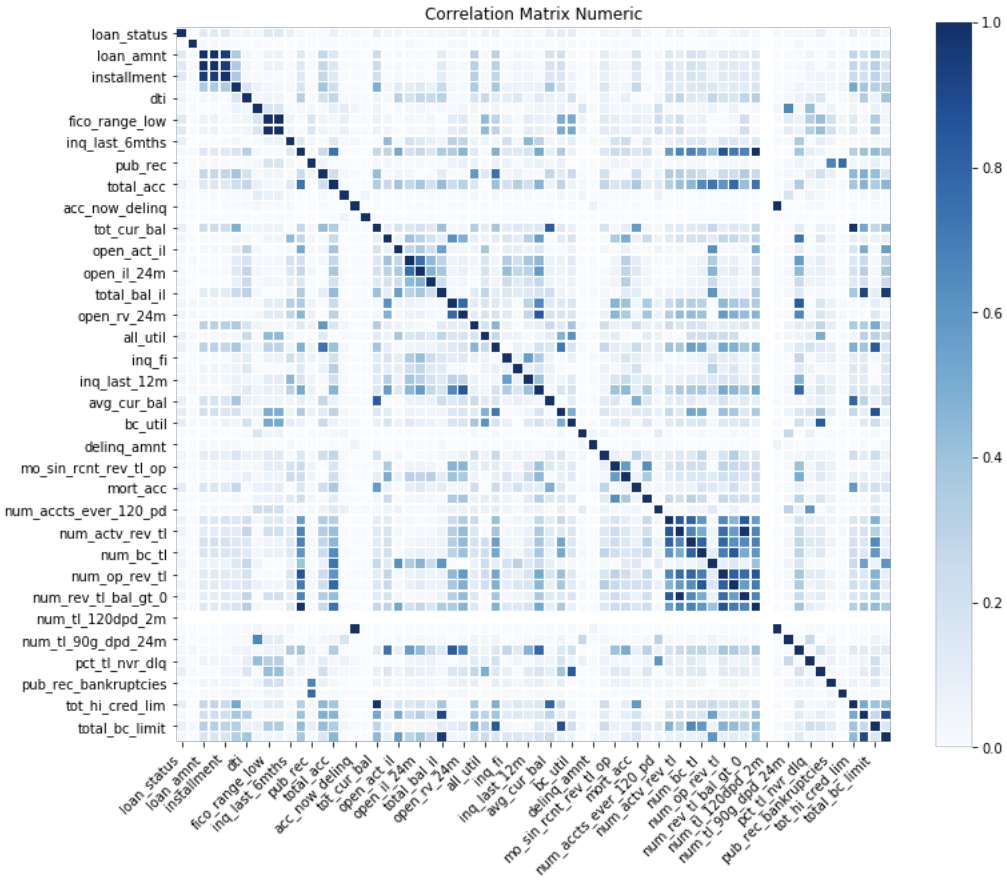

Figure A.1: Correlation Matrix for Numeric Variables

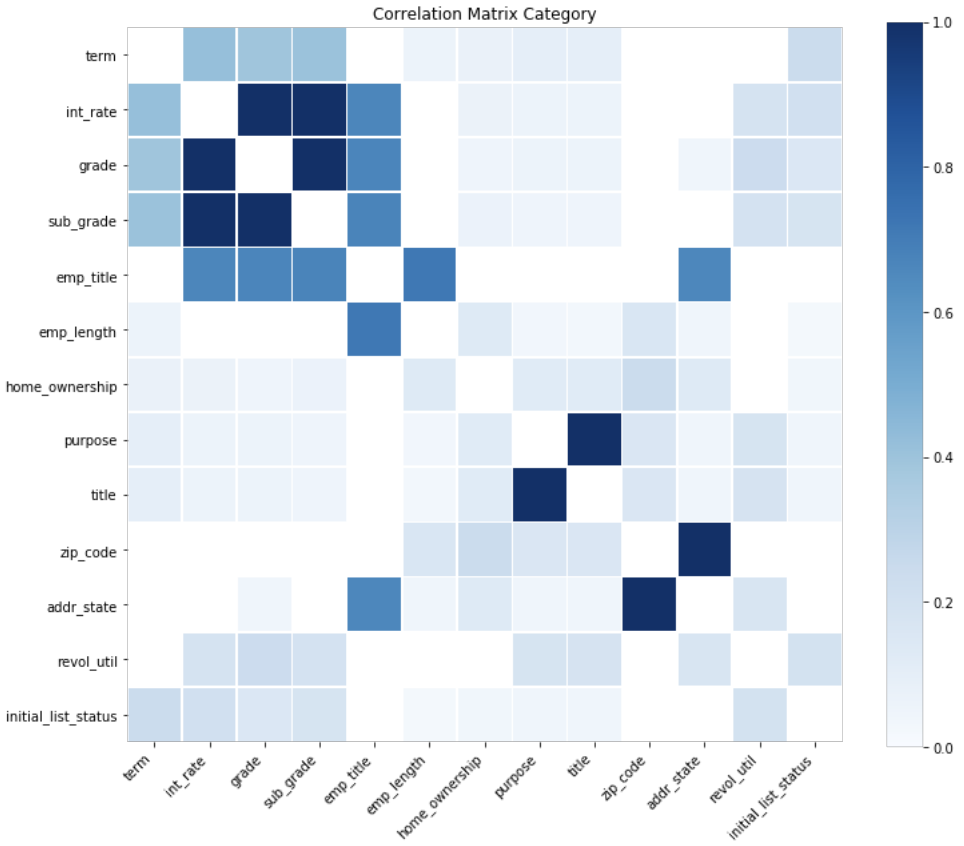

Figure A.2: Correlation Matrix for Categorical Variables

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100)$ * T synthetic minority class samples

(* *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* *)
if $N < 100$
  then Randomize the T minority class samples
    $T = (N/100) * T$
    $N = 100$
endif
$N = (int)(N/100)$ (* *The amount of SMOTE is assumed to be in integral multiples of 100.* *)
k = Number of nearest neighbors
$numattrs$ = Number of attributes
$Sample[\,][\,]$: array for original minority class samples
$newindex$: keeps a count of number of synthetic samples generated, initialized to 0
$Synthetic[\,][\,]$: array for synthetic samples
(* *Compute k nearest neighbors for each minority class sample only.* *)
for $i \leftarrow 1$ **to** T
    Compute k nearest neighbors for i, and save the indices in the $nnarray$
    Populate(N, i, $nnarray$)
endfor

Populate(N, i, $nnarray$) (* *Function to generate the synthetic samples.* *)
while $N \neq 0$
    Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
    for $attr \leftarrow 1$ **to** $numattrs$
      Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
      Compute: gap = random number between 0 and 1
      $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
    endfor
    $newindex{+}{+}$
    $N = N - 1$
endwhile
return (* *End of Populate.* *)

Figure A.3: SMOTE Algorithm

B Codes

```
## Numeric variables correlation matrix plot - Python
import pandas as pd
import seaborn as sns
%matplotlib inline
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import cm

df_new = pd.read_csv("2018Q1_correlation.csv")
df_new.loan_status = df_new.loan_status.map(lambda x: 1 if x ==
                                            'Fully␣Paid' else 0)
df_num = df_new.select_dtypes(['int64','float64'])
loan_status = df_num.loan_status
df_num.drop('loan_status', axis =1 , inplace = True)
df_num.insert(0,'loan_status', loan_status)
df_num = df_num.corr().abs()
ccmap = sns.palplot(sns.light_palette('deep␣sky␣blue', input = 'xkcd',
                                      n_colors = 10))

plt.figure(figsize=(12, 10))

corr = df_num

ax = sns.heatmap(
    corr,
    vmin = 0, vmax=1, center=0.5,
    cmap= "Blues",
    square=True,  linewidths=.5
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);

plt.title('Correlation␣Matrix␣Numeric')
plt.show()

## Categorical variables correlation matrix plot - Python
df_obj = df_new.select_dtypes('object')
df_plot_category = pd.read_csv("cramerV_matrix_1208.csv", index_col=0)
```

```
plt.figure(figsize=(12, 10))

ax = sns.heatmap(
    df_plot_category,
    vmin = 0, vmax=1, center=0.5,
    cmap= "Blues",
    square=True,  linewidths=.5
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);

plt.title('Correlation␣Matrix␣Category')
plt.show()

## Test the correlation between features - R
test_cor <- function(data){
  n = ncol(data)
  features <- colnames(data)
  types <- sapply(data, class)
  nodes <- data.frame(id = features, label = features)
  edges <- data.frame(from = c(), to = c(), label = c(),
                      title = c(), coefficient = c())
  for(i in 1:(n-1)){
    print(i)
    print(features[i])
    for(j in (i+1):n){
      print(features[j])
      flush.console()
      if(types[i] == 'factor'){
        if(types[j] == 'factor'){
          print('f+f')
          flush.console()
          table <- data %>% group_by(get(features[i]),
                                     get(features[j])) %>%
            summarise(n = n()) %>%
            spread(key = `get(features[i])`, value = n) %>%
            column_to_rownames('get(features[j])')
          table[is.na(table)] <- 0
          res <- chisq.test(table)
          p.value <- res$p.value
```

```
coefficient <- cramerV(as.matrix(table))
message <- paste("<p␣style='font-size:␣8pt;'><b>",
                  toupper(features[i]), 'with',
                 toupper(features[j]),
                 ':</b><br>Test␣method␣=␣',res$method,
                 '<br>chi-squared␣=␣', res$statistic,
                 ';␣df␣=␣', res$parameter,
                 ';␣P-value␣=␣', p.value,
                 '<br>Cramer␣V␣=␣', coefficient,'</p>')
if(p.value < 0.01){
  edges <- rbind(edges, data.frame(from = c(features[i]),
                                   to = c(features[j]),
                                   label = c(sprintf("%.2e",
                                   p.value)),
                                   title = message,
                                   coefficient = coefficient))
}
  else{
    print('f+n')
    flush.console()
    # Anova
    ano = my_anova(data,features[i], features[j])
    if(ano$valid == TRUE){
      message = ano$message
      p.value = ano$p.value
    }else{
      res = kruskal.test(as.formula(paste(features[j],
                         features[i], sep="␣~␣")),
                         data = data)
      p.value <- res$p.value
      message <- paste("<p␣style='font-size:␣8pt;'><b>",
                       toupper(features[i]), 'with',
                       toupper(features[j]), ':</b>␣Anova␣Unvalid:',
                       ano$message,
                       '<br>Test␣method␣=␣',res$method,
                       '<br>chi-squared␣=␣', res$statistic,
                       ';␣df␣=␣', res$parameter,
                       ';␣P-value␣=␣', p.value,'</p>')
    }
  coefficient <- NA
  }
}else{
  if(types[j] == 'factor'){
```

```
  print('n+f')
  flush.console()
  # Anova
  ano = my_anova(data,features[j], features[i])
  if(ano$valid == TRUE){
    message = ano$message
    p.value = ano$p.value
  }else{
    res = kruskal.test(as.formula(paste(features[i],
                        features[j], sep="␣~␣")),
            data = data)
    p.value <- res$p.value
    message <- paste("<p␣style='font-size:␣8pt;'><b>",
                  toupper(features[i]), 'with',
                  toupper(features[j]),
                  ':</b>␣Anova␣Unvalid:', ano$message,
                  '<br>Test␣method␣=␣', res$method,
                  '<br>chi-squared␣=␣', res$statistic,
                  ';␣df␣=␣', res$parameter,
                  ';␣P-value␣=␣', p.value,'</p>')
  }
  coefficient <- NA
}else{
  print('n+n')
  flush.console()
  # Pearson
  pea <- my_pearson_test(data, features[i], features[j])
  if(pea$valid == TRUE){
    message <- pea$message
    p.value <- pea$p.value
  }else{
    if(length(boxplot.stats(data[,features[i]])$out) < 500 &
       length(boxplot.stats(data[,features[j]])$out) < 500){
      res_s <- cor.test(data[,features[i]],
                        data[,features[j]], method = "sp")
      p.value <- res_s$p.value
      coefficient <- res_s$estimate
      message <- paste("<p␣style='font-size:␣8pt;'><b>",
                  toupper(features[i]), 'with',
                  toupper(features[j]),
                  ':</b>␣Pearson␣Unvalid:␣', pea$message,
                  '<br>Test␣method␣=␣', res_s$method,
                  '<br>S␣=␣', res_s$statistic,
```

```
                                ';␣rho␣=␣', res_s$estimate,
                                ';␣P-value␣=␣', p.value,'</p>')
                  }else{
                    res_k <- cor.test(data[,features[i]], data[,features[j]],
                                      method = "kendall")
                    p.value <- res_k$p.value
                    coefficient <- res_k$estimate
                    message <- paste("<p␣style='font-size:␣8pt;'><b>",
                                toupper(features[i]), 'with',
                                toupper(features[j]),
                                ':</b>␣Pearson␣Unvalid:␣', pea$message,
                                '<br>Test␣method␣=␣', res_k$method,
                                '<br>z␣=␣', res_k$statistic,
                                ';␣tau␣=␣', res_k$estimate,
                                ';␣P-value␣=␣', p.value,'</p>')
                   }
                }
              }
          }
          # if(p.value < 0.01){
          #   edges <- rbind(edges, data.frame(from = c(features[i]),
          #                                    to = c(features[j]),
          #                                    label = c(sprintf("%.2e",
                                                         p.value)),
          #                                    title = message,
          #                                    coefficient = coefficient))
        }
      }
    }
    return(list(nodes = nodes, edges = edges))
}
# data = read.csv('./Data/2018Q1_2.csv')
data = read.csv('./Data/2018Q1_categorical.csv')
network_category <- test_cor(data)
cor_cat <- network_category$edges
# network <- test_cor(data_net)
# network_try <- test_cor(int_grad)
# edges <- network_try$edges
# nodes <- network_try$nodes
try <- cor_cat[c("from", "to", "coefficient")] %>%
  spread(key = to, value = coefficient) %>%
  column_to_rownames("from")
write.csv(try, './Data/cramerV_matrix.csv')
```