ONLINE PEER-TO-PEER LOAN STATUS PREDICTION WITH BIG DATA AND DATA MINING

Yan Gao

GPIM 452 Big Data Analytics

University of California, San Diego

yanjessicagao@gmail.com

June 2018

TABLE OF CONTENT

TABLE OF FIGURES AND TABLES

1 Executive Summary

Because of simple and convenient applying procedure, online P2P lending has become an increasingly popular financing choice for individuals and some companies. Since loan in P2P platform usually has much higher interest rate than bank saving and treasury bonds, it is appeal to investors. But it also has s disadvantage of high credit risk because of borrows' delinquency.

Therefore, mainly based on data from Lending Club, this paper is written to two major group of people. Firstly, **P2P loans' investors** may use models in this paper to find "safe" loans (loans with less possibility to become charged off in the future) so as to obtain higher investment return ratio; Secondly, **workers in Lending Club** may use models in this paper to filter bad loans (loans that are like to be charged off in the future) so that more investors may be attracted to fund their money in the platform.

In this research, Lasso and Random forest are the two models used to classify loan status; Changing cutoff and oversampling are the two approaches applied to address the problem of unbalanced data. With the maximum average extra return ratio, Random forest with selected cutoff is the best model in this paper. Putting data of loan issued between August 2016 and September 2017 as the new testing data, using this best model **gives about 65 million dollars extra return and 5.5% higher return ratio.** Interest rate, average current balance, total credit limit and debt to income ratio are the most important features in determining loan status.

This paper has two major innovations. First is revising functions in R package "Glmnet" (created by Hastie T and Qian J) to combine cross validation with oversampling. Second is using "Average Extra Return Ratio" as the measure to select best model. Several limitations exist, like censoring data and low testing data accuracy. And more future works can be done, such as using more lag data, text mining and changing objective function to maximizing Average Extra Return Ratio.

2 Introduction

2.1 Background

Online P2P (Peer-to-Peer) Internet lending is a kind of loan settled on P2P platform. The intermediary platform is the important connector between borrowers and lenders. For example, as one of the biggest P2P platform in United States, Lending Club on one hand allows borrowers, mainly for individuals and SMEs (Small and Medium Enterprises), to apply loans as long as they qualify some basic regulations; on the other hand, allows individuals and institutions to invest loans on the platform; and meanwhile, charges a handling fee.

Two facts should be addressed here. Firstly, Lending Club performs only the role of intermediary and it is the lender who bears the credit risk. That is to say, the borrower needs to give Lending Club an origination fee at the starting point when receiving funds and after that, the payment for the loan will directly go to lenders. So if the borrower does not pay the principles and interests, it is the lender who losses profits.

Secondly, interest rate of a loan is mainly defined by Lending Club. Based on borrower's personal information and loan's information, this platform would give every loan a grade from A to F and a corresponding set of interest rate. Loan can only be applied within the given set of interest rate. The accurate calculation of how the grade as well as the interest rate be generated is not exhibited publicly by Lending Club.

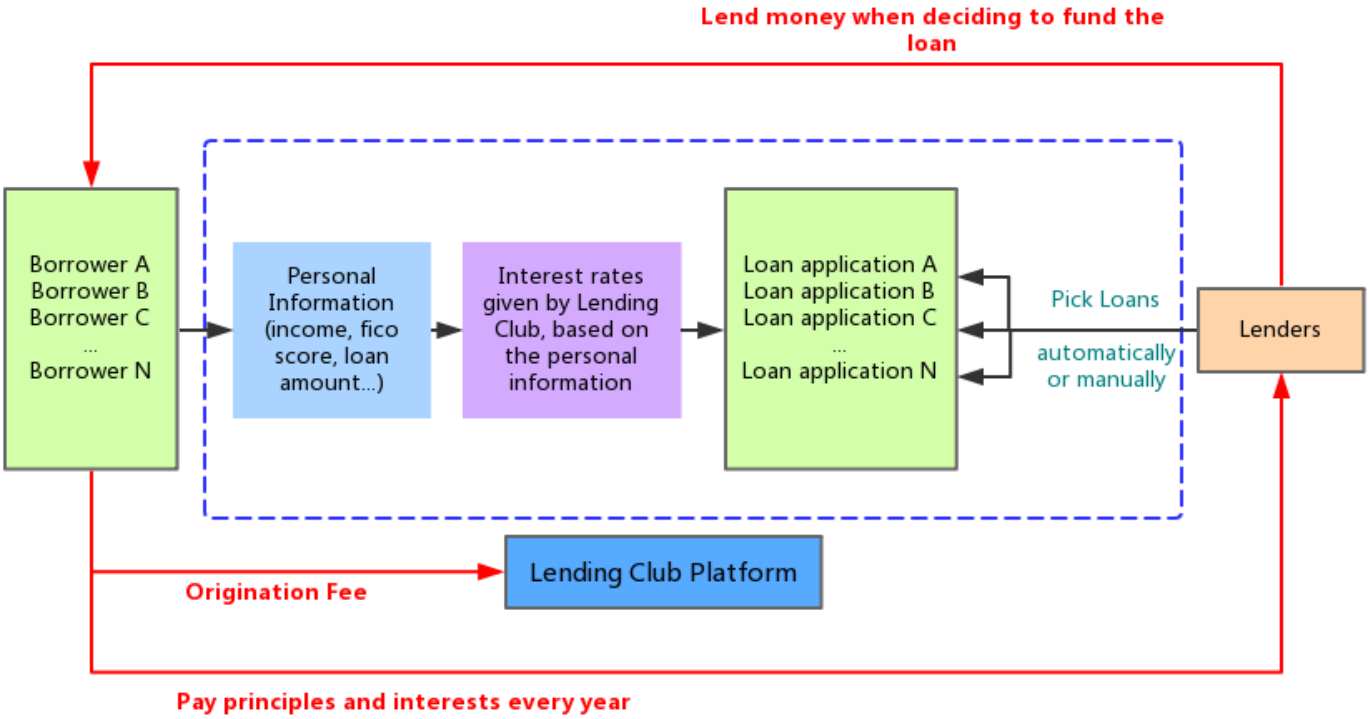

Figure 1. Structure of borrowing & lending process in Lending Club

P2P network lending has eased the current state of financing difficulties for SMEs and individuals. Compared with formal financial institutions, it has unparalleled competitive advantages such as low loan threshold, convenient transaction procedures, and small amounts involved. However, due to P2P's low entry barriers, lack of supervision, and imperfect platform risk control measures, the credit risk of the P2P network lending market is high, and credit risk events like loan defaults frequently occur.

2.2 Project Importance

According to Lending Club [36], from 2017Q1 to 2018 Q1, almost 2 billion dollars were default which is as 3 times as the amount issued in 2017. These huge amount of default (charged off) loans definitely decreased investor's return ratio and also brought bad influence to the platform Lending Club.

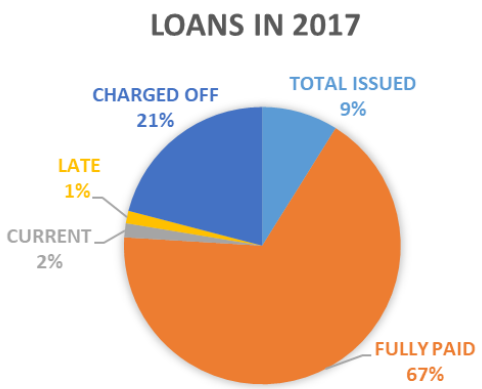

Figure 2. Loans status in 2017

Therefore, when someone applies a loan, it is of great value for investors and Lending Club's worker to predict whether this loan will be default or not in the future. With such a goal, this project's domain is Finance. Accurate prediction of future default loans will not only increase investor's investment return, but also will enlarge Lending Club's reputation so that more investors will join in it.

Overall, based on the data mining methods, this paper studies a reasonable and effective loan default prediction model, provides inspiration to the risk control mechanism of the online P2P lending platform, and provides support for solving the risk control issues of the P2P network lending platform construction.

2.3 Literature Review

Research on forecasting credit risk and scoring credit can be dated from 1940s and has developed greatly from then on. According to method's principles, commonly used credit risk prediction techniques can be classified into statistical methods like discriminant analysis, and non-statistical methods like data mining.

Discriminant analysis is one of the most popular models in statistical methods. Selecting variables that provide more information to establishing a discriminant function, it is a statistical analysis method for discriminating the category to which the object belongs. The pioneer who first applied this method to financial crisis, corporate bankruptcy, and default risk analysis was Dr. Edward I. Altman in 1968 [2]. He adopted 22 financial ratios, and established the famous 5-variable Z-score model and the improved "Ze-ta" discriminant analysis based on mathematical statistics screening model. The Zeta model has been commercialized due to its simplicity and low cost.

Because the statistical model has more rigorous assumptions, linear techniques cannot distinguish between random noise and nonlinear relationships. Recent years have witnessed the prosperity of computer science and as a result, data storage and analysis methods have been increasingly improved. With the continuous emergence of this phenomenon, data mining methods are more commonly applied in credit scoring. Odom (1990) [39] introduced the neural network into the bankruptcy domain for the first time and used the BP neural network to predict financial distress. The results show that the neural network model is superior to the discriminant analysis model.

As for P2P loans. Hulme and Wright (2006) [29] conclude the emergence of Social Lending as a significant development within the financial sector is a direct response to social trends and a demand for new forms of relationship. Michael Klafft (2008) [35] explores investment return of P2P loans and gives some simple investment rules to improve profitability of a portfolio. Herrero (2009) [26] measures the influence of social interactions in the risk evaluation based on data provided by Prosper.com. Berkovich (2011) [4] also uses Prosper dataset and finds that high-priced loans provide excess returns even after accounting for risk-aversion. Gonzalez et al. (2014) [16] examines the effect of both borrower and lender personal characteristics, including perceived attractiveness, age and gender, on peer-to-peer lending decisions by regression models. Riza and Yanbin (2015) [44] implement the nonparametric tests to examine if there is a significant difference in the variables between defaulted loans and good status loans and then model the default risk of the loan applicants by employing a binary logit regression. Fatemeh et al. (2015) [10] employ a new hybrid credit scoring model of FS algorithms and ensemble learning classifiers in order to attain the suitable feature subset with higher classification performance. Jin and Zhu (2015) [10] compare five data mining models: two decision trees (DTs), two neural networks (NNs) and one support vector machine (SVM) to predict loan status with Lending Club datasets and find several important variables in loan defaults.

3 Data

3.1 Dataset: Lending Club

3.1.1 Data Source

This paper mainly uses data provided by Lending Club, which derived from transactions in its own network. The original datasets, which are provided as .csv files (Comma Separated Value), contain complete loan data for all loans issued through 2007 to 2017, including the current loan status (Current, Late, Fully Paid, etc.) and latest payment information. The raw data could be retrieved from:

Lending Club at https://www.lendingclub.com/info/download-data.action

or in Kaggle at https://www.kaggle.com/wordsforthewise/lending-club/version/4/data.

3.1.2 Data Description

About 1.5 million loan records with more than 150 variables are included. Each instance contains four kinds of variables: First is the client's personal information when applying the loan, such as fico score, annual income and so on; Second is the loan's information given by the client when applying the loan, such as loan amount, issued date and so on; Third is information given by Lending Club when after the loan be applied and before the loan has funded, including grade and interest rate; Fourth is information after the successfully funded, such as loan status, payments received to date, flags whether or not the borrower is on a hardship plan and so on.

3.1.3 Data Manipulation

Among two kinds of application type, i.e. individual application and joint application (applied and paid by two co-borrowers), the latter one takes up only about 0.6% of the whole dataset, thus this paper simply eliminates instances of joint loan and variables related with the second co-borrower. Also, some other variables that have too many missing values are also be dropped. Loans that did not fully funded (fund amount less than apply amount) are excluded.

As this paper's goal is to determine the loan's status when it has applied but has not funded, loan status is the output variable in the paper and other variables in the fourth kind described in 4.1.2 should be dropped. The raw data has 9 kinds of loan status. To simplify the problem, this paper deletes loans in 3 unsure statuses, including loans in current (we won't know whether these loans will be default or not.) and loans that does not meet the credit policy; then idealizes other 6 kinds into 3 general kinds, shown in Figure 3. Since loans need attention takes up only of the whole dataset, this paper dropped instances in this status. Finally, the dependent variable is status which contains 2 classes, i.e., good loans and bad loans.

Original status	Frequency	Generalized status	Desity
Current	749145	Not sure	(deleted)
Fully Paid	744230	Good loan	0.76
In Grace Period	9346	Need attention	0.03
Late (16-30 days)	5207		
Late (31-120 days)	19402		
Default	78		
Charged Off	195479	Bad loan	0.20

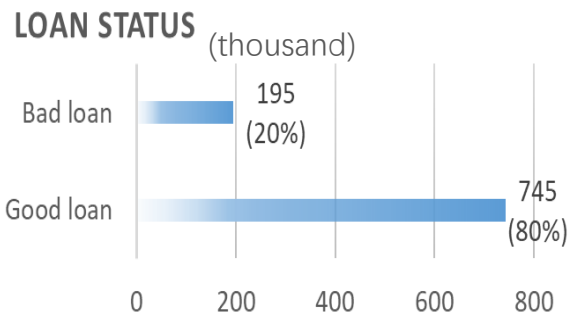

Figure 3. Loan Status Sorts

3.1.4 Feature Engineering

The dataset so far still has problems, such as exists of many missing values, uneven data distribution and too many categories of some nominal variables. Therefore, basically, the following four principles are applied when creating or transforming variables and the detailed process of feature engineering shows in Appendix 1.

1) Combining and generalizing some variables with similar meaning.

For example, in original dataset, “Months since most recent 90-day or worse rating”, “Months since most recent bankcard delinquency”, “Months since most recent revolving delinquency” and “The number of months since the borrower's last delinquency” all record the number of month since the last bad performance. Thus for each loan, the new variable “months since last bad record” could be created by computing the minimum number of the former 4 variables, ignoring the missing value.

2) Using ratio instead of number for some variables.

For example, if person A has 10 accounts and 3 of them are satisfactory (i.e. 70% delinquent rate), while person B has 4 accounts and 3 of them are satisfactory (i.e. 25% delinquent rate). Although the A's number of satisfactory account are larger, he is more likely to default a loan than B. Thus the ratio can do a better work.

3) Applying non-linear transformation, like log or square root, to variables whose cumulative distribution are not linear at all.

Annual income, for example, is transformed into log(income). This creation can reduce the extreme of outliers and make data distribute more even.

4) Transforming or substituting categorical variables with too many classes into numerical ones.

If a categorical variable has too many classes, like "address states" which contains 50 classes, when putting it into a model, 49 dummy variables would be generated which increase model's complexity uselessly. But we cannot simply drop this variable, because it may contain important information. Let good ratio be computed by number of good loans over number of all loans, issued in specific year and states, thus deep color stands for low credit and vice versa. Then by drawing the GIS plot of good ratio of loans issued between 2013 to 2016, grouped by year and state, as Figure 4 shows, two important observations should be attached attention to.

Firstly, as time goes by, the overall color is deeper and deeper and the number of grey space is fewer and fewer. Grey means that this state has very few (less than 10) loans issued in the mentioned year. This indicates the fact that during 2013-2016, Lending Club is more and more popular but the portion of default loans increases. Therefore, Lending Club needs to control loan's quality, to reduce the number of default loans and so as to hold its investors. Otherwise, investors may go to its competitor, i.e. other P2P platform, because of the high defaulting rate (which leads to low return rate) of loans in Lending Club.

Secondly, some states like ME, NH, VT and CO, always have high good ratio, while some other like OK, LA, AL and NY have low good ratio in all years. In the plot, top 3 worst states every year are marked in yellow letters. Therefore, default probability is not only related with year, but also related with states. USA economic data, like GDP, CPI, interest rate, are good indicators of year; States economic data, like states unemployment rate and states average annual income, are good indicator of states.

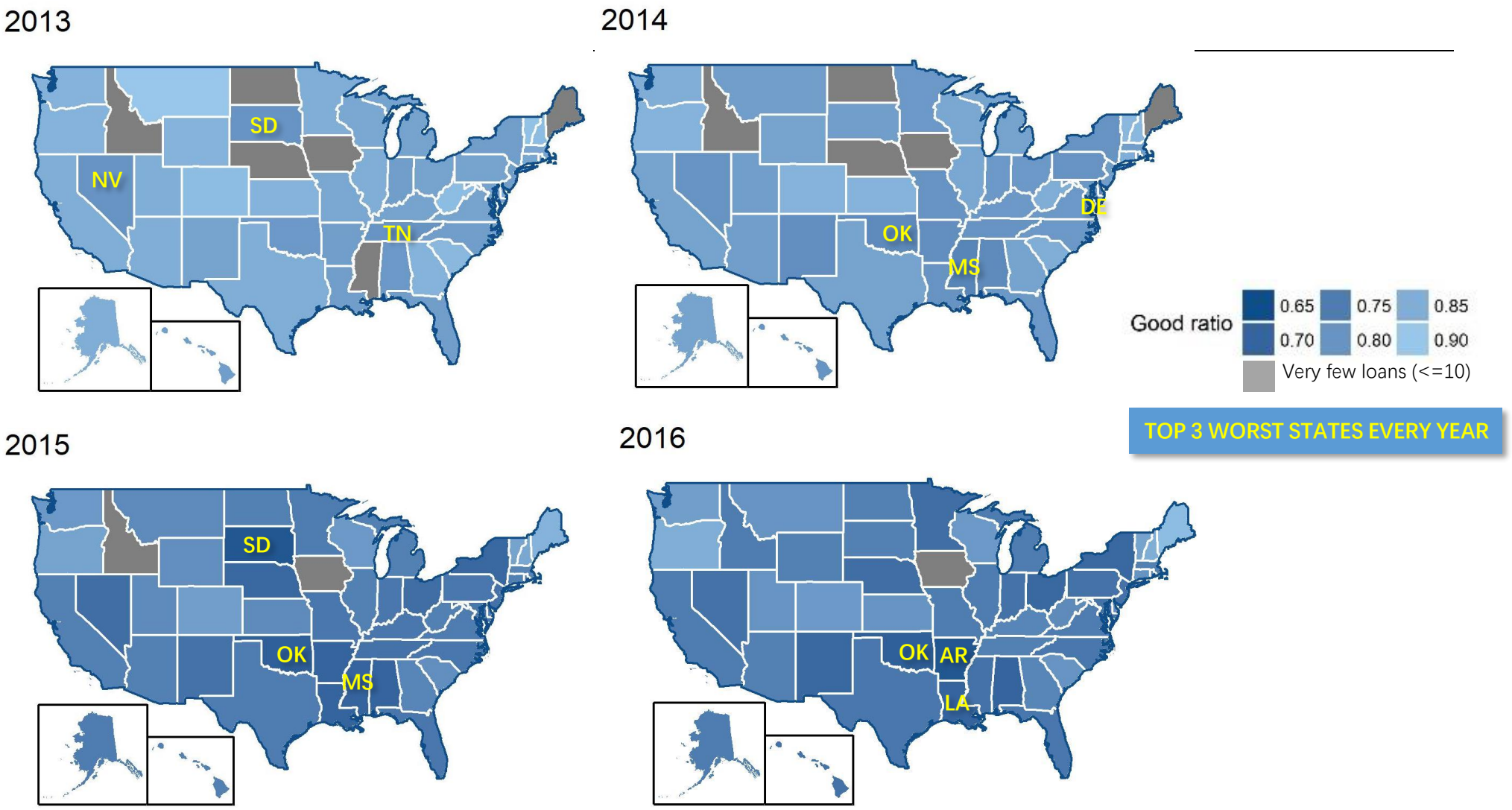

Figure 4. Good Ratio of loans issued between 2013 to 2016, grouped by year and state.

3.2 Dataset: WIND Economic Database

As it is mentioned in the formal section, weather a loan will be well-paid or not is a problem affected by multiple factors. It not only depends on the borrower's personal information like annual income and FICO score, the loan's information, like issue date, loan amount and term, but also depends on the economic situation of whole country and the states where the loan is issued. If some accidents happen and lead to economy decline, the income of the loan borrower may decrease, or in a worse situation, he or she may even be fired, so the default probability of this loan may increase.

Therefore, 9 USA economic variables (like interest rates, GDP and so on) and 2 states economic variables (state average income and state unemployment rate) are derived from Wind [54]. Very similar to Bloomberg in USA, Wind is an economic database in China.

Note that it is the lag 1 values of these variables are introduced.

Issued date and states are not good variables to be included in models since they are nominal variables with so many categories that many dummy variables would be generated when using models like Lasso. In Figure 5, the left column shows monthly economic data in the United States; the middle column points the monthly state unemployment rate of different states; and the right column describes the yearly state average annual income of different states. Due to high correlation seen in Figure 5, Dates and states, can be well substituted by some numerical economic data.

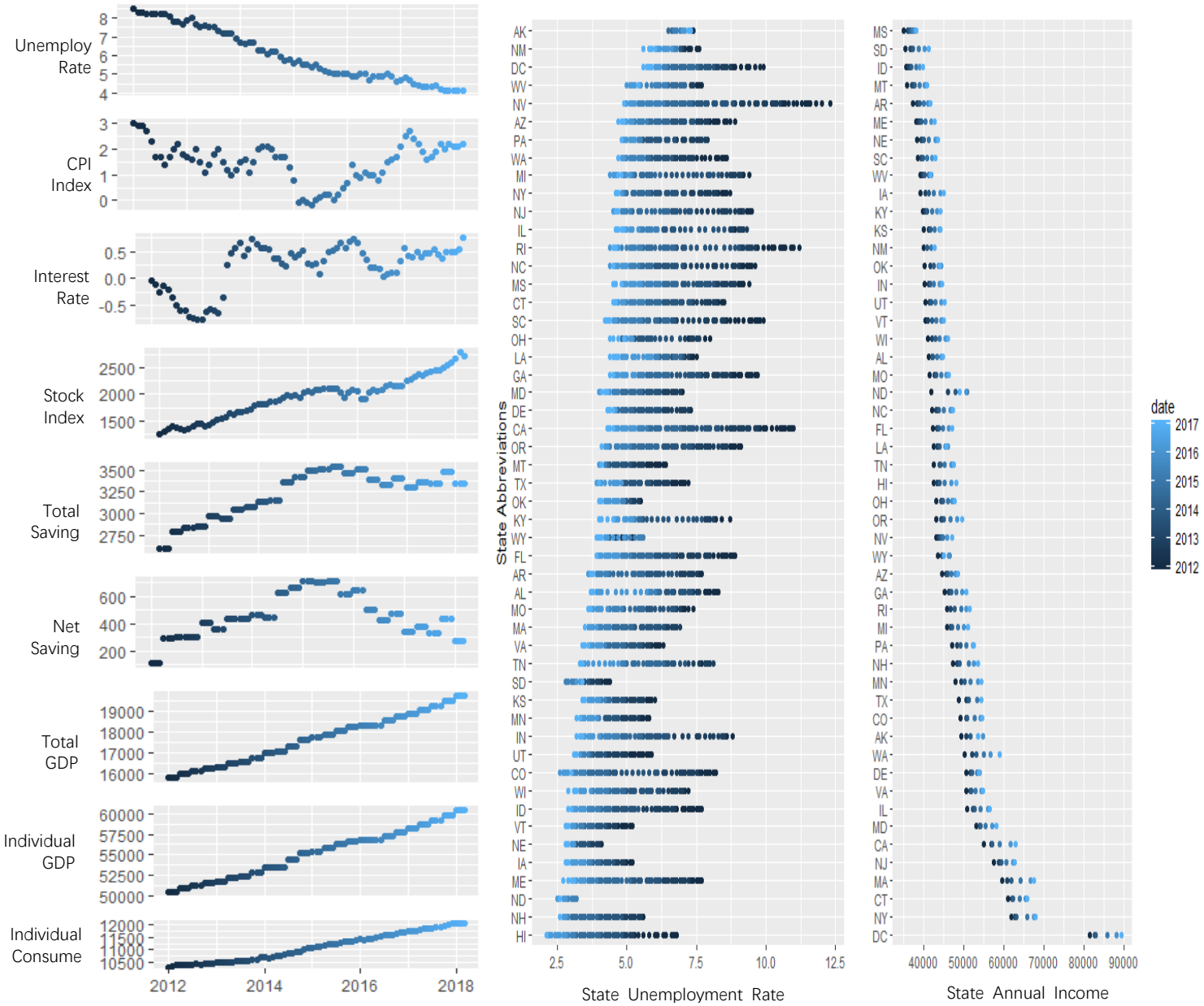

Figure 5. Economic data plots

3.3 Final Dataset and Data Exploration

After merging the two dataset above, creating new variables and dropping missing values, the final dataset has about 600 thousand instances with about 100 variables. No missing value exists. Samples with all variables and variables' description are attached in Appendix 1. The dependent variable is "status", which has two classes: Good and Bad. This paper assumes that good loans have already been well paid while bad loans are default and have no possibility to be paid in the future. This is a reasonable assumption, because according to Lending Club, a loan becomes "Charged Off" when there is no longer a reasonable expectation of further payments, and in this dataset, charged off loans take up 99.6% of bad loans (the rest 0.4% are default loans).

Figure 6 gives us a glimpse of seven variables in the dataset. For each variable x and each status y, the number in the radar plot is calculated by the following Formula (1).

$$\frac{(mean\ x\ of\ loans\ in\ status\ y) - (5\%\ percentile\ x\ of\ all\ loans)}{(95\%\ percentile\ x\ of\ all\ loans) - (5\%\ percentile\ x\ of\ all\ loans)} \quad (1)$$

The reason why using 5% and 95% percentile rather than the minimum and maximum value is that percentile values could eliminate the huge effect of extreme values. So this score reflects the average value of this variable for loans in the specific status. Differences between good and bad loan could be told. Compared to good loans' applicants, applicants of bad loan are more likely to have low FICO score, low total open to buy on revolving bankcards, low credit limit, high loan amount, high interest rate and high average balance to limit for all of his or her accounts. From Figure 7, relationship between loan number, amount, issued date and good ratio can be seen.

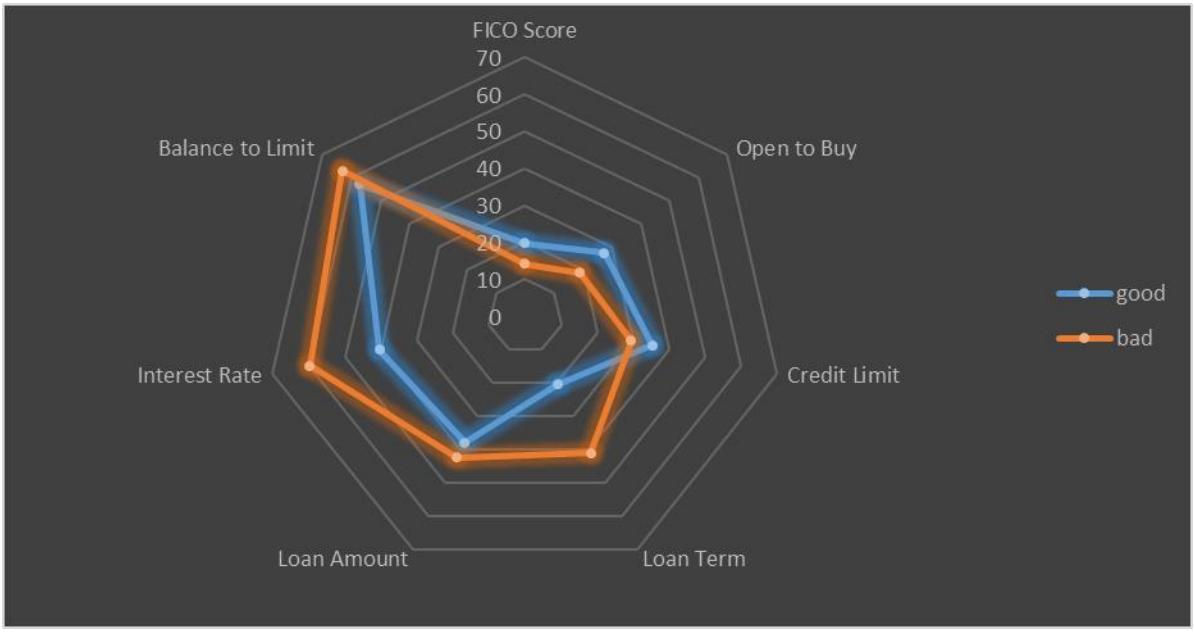

Figure 6. Radar plot of some variables, grouped by Status

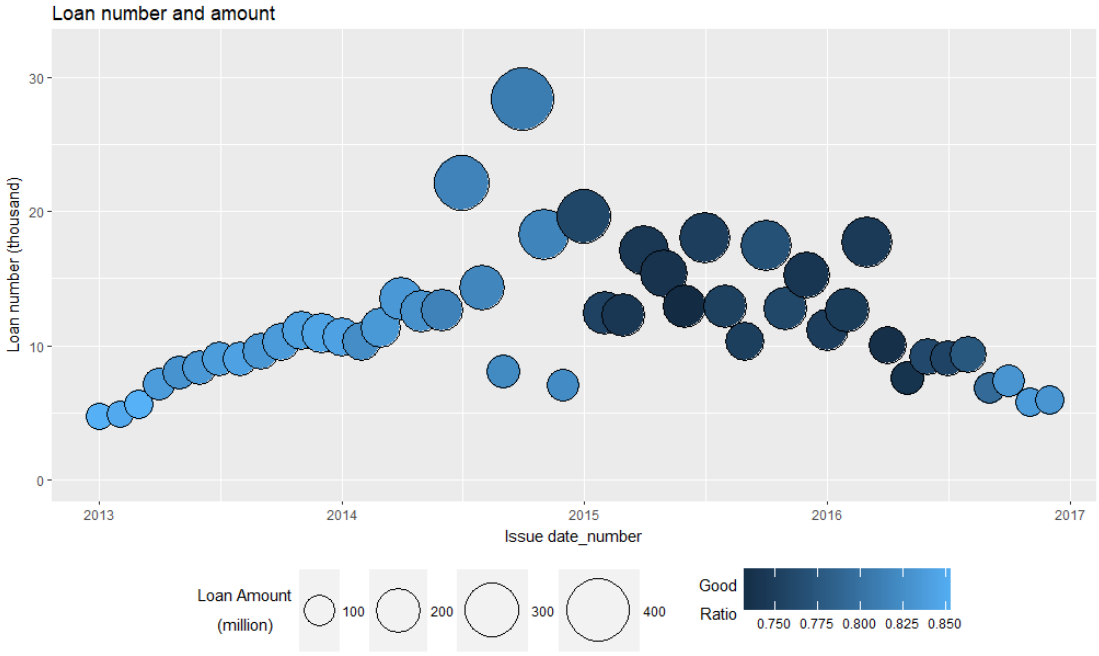

Figure 7. Bubble Plot Loan Number, Amount, Issued Date and Good Ratio

4 Models and Quantitative Results

In this dataset, loan's issued year ranges from 2012 to 2017. Figure 8 shows the ratio of each year's issued loan number. This paper uses loans issued from 2013 to August 2016 as training and validate data, which take up about 85% of the whole dataset, and uses the rest loans as testing data. The two models in the following sessions are built based on training and validate data. The ratio between training data number and validating data number is 4:1. Therefore, the approximate ratio between number of these three kinds of data is training: validating: testing = 0.7: 0.15: 0.15.

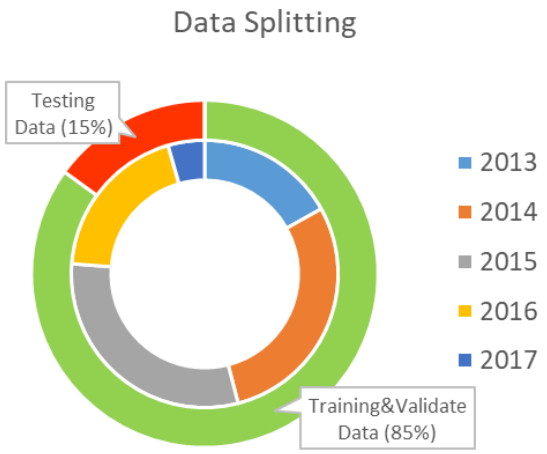

Figure 8. Data Splitting

As it is discussed in 4.1.3, the number of good loans and bad loans is unbalanced, so this paper uses 2 methods to solve this problem. First is oversampling the training data when building models. Note that when validating and testing the built model, this paper puts into the unbalanced data without oversampling [9]. Second is tuning cutoff of validation data. The following sections show which method is better in this dataset.

4.1 Model Evaluation Methodology

When taking business costs into account, the best model should encourage the algorithm to maximize the total extra return of using this model compared to not using this model.

As mentioned before, the issue date of training and validating data ranges from 2013 to August 2016, so this paper calculates the payment ratio of loans issued before August 2016. Grouped by status, good loans averagely be paid as 1.16 times as their loan amount; while bad loans averagely be paid about half of their loan amount, as Figure 9 shows. Note that although every loan's payment is a variable in the original dataset provided by Lending Club, it does not and should not be included in model's attributes. The payment ratio is calculated by Formula (2) and (3)

$$\text{(Payment Ratio)}_{good\ loan} = \frac{total\ payment\ of\ all\ good\ loans\ before\ June\ 2016}{total\ loan\ amount\ of\ all\ good\ loansbefore\ June\ 2016} = 1.16 \qquad (2)$$

$$\text{(Payment Ratio)}_{bad\ loan} = \frac{total\ payment\ of\ all\ bad\ loans\ before\ June\ 2016}{total\ loan\ amount\ of\ all\ bad\ loansbefore\ June\ 2016} = 0.52 \qquad (3)$$

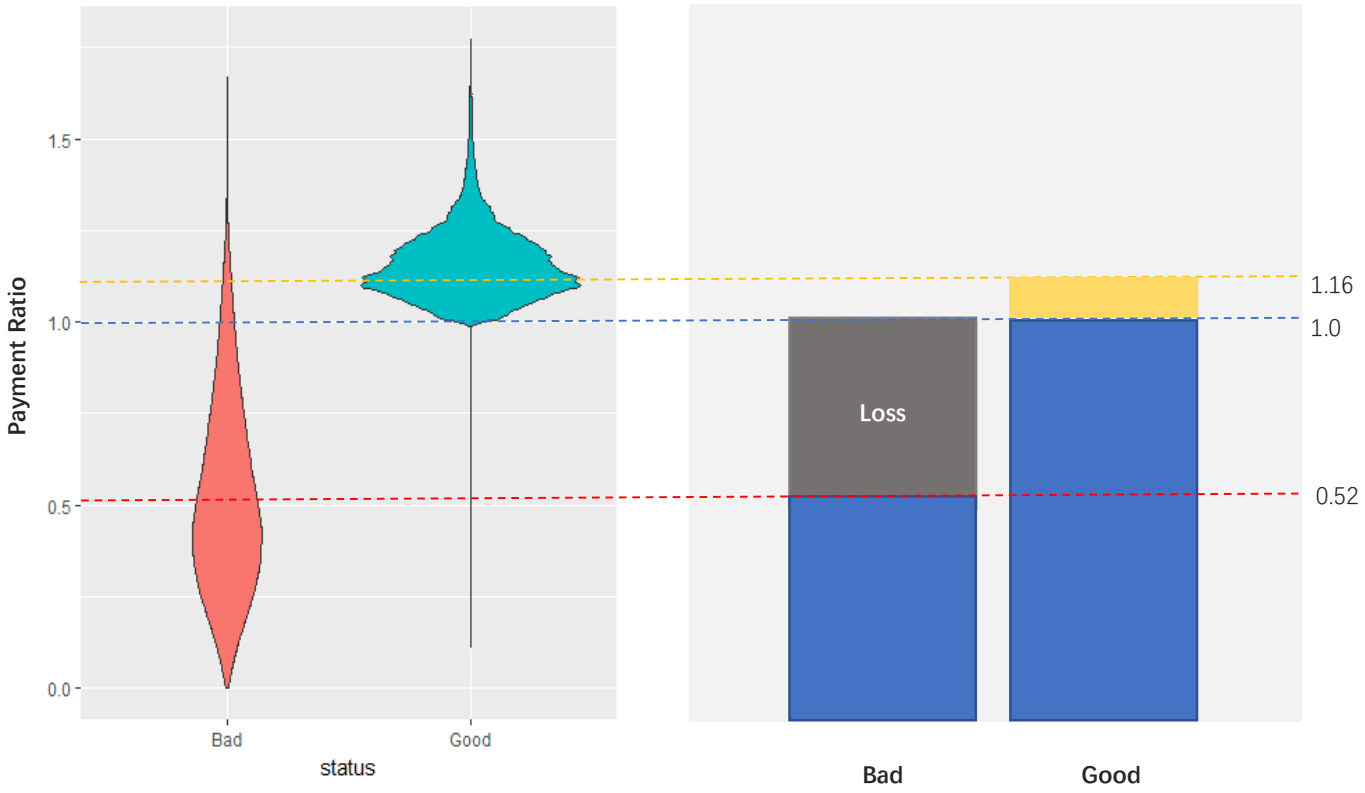

Figure 9. Violin Plot and payment ratio

Good loan means it has been fully paid; Bad loan means it is default and will no longer be paid in the future. Therefore, the return matrix can be written as Table 1: Compared with reality, in which all of those loans are accepted, rejecting a good loan will lead to losing 16% return, so the extra return is -16%; rejecting a bad loan will result in avoiding 48% loss, so the extra return is +48%; while accepting a loan is just as same as what is done in the reality, so the extra return is 0, no matter it is a good loan or bad loan. The Confusion matrix can be written as Table 2. And the Specificity and sensitivity are computed as Formula (4).

	Actual = Good	**Actual = Bad**
Predict = Good	0	0
Predict = Bad	-16%	48%

Table 1. Extra Return Matrix

	Actual = Good	**Actual = Bad**	
Predict = Good	TP (true positive)	FP (false positive)	
Predict = Bad	FN (false negative)	TN (true negative)	
	P (positive) = TP+FN	N (negative) = FP+TN	Total = TP+FN+FP+TN

Table 2. Confusion Matrix

$$\text{Sensitivity} = \frac{TP}{P}; \quad \text{Specificity} = \frac{TN}{N} \qquad (4)$$

With the cost matrix and the confusion matrix above, this paper computes the average return of each loan as Formula (5).

$$\bar{R} = \frac{1}{Total} \times \sum_{i=TP,FP,FN,TN} i \times [Cost \ \ of \ \ i] \qquad (5)$$

$$= \frac{1}{Total} \times (FN \times -0.16 \ \ + TN \times 0.48)$$

$$= \frac{P}{Total} \times \frac{FN}{P} \times -0.16 + \ \ \frac{N}{Total} \times \frac{TN}{N} \times 0.48$$

$$= \frac{P}{Total} \times (1 - \text{Sensitivity}) \times -0.16 + \ \ \frac{N}{Total} \times \text{Specificity} \times 0.48$$

Validating data as well as their status are what we already know, so P, N, and T are constant numbers. In this research, assuming number of good loan is as 4 times as the number of bad loan (which shows in Figure 3), then P/Total=0.8 and N/Total=0.2. Therefore, the final cost formula is Formula (6).

$$\begin{aligned}\bar{R} &= -0.128 \times (1 - Sensitivity) + 0.096 \times Specificity \\ &= -0.128 + 0.128 \times Sensitivity + 0.096 \times Specificity \quad (6).\end{aligned}$$

As what expected, the model needs to increase the Sensitivity and Specificity in the validating confusion matrix. With such a formula, the coefficient of sensitivity is larger than that of specificity, indicating that sensitivity is more important than specificity. So here comes an interesting phenomenon. We may propose it is more valuable to correctly predict the bad loans than to correctly predict the good loans, as the cost of each bad loan is larger than the return of each good loan. That is true for a single loan, but not for the whole model. Instead, when it comes to the total return of all loans rather than the return of a single loan, ratio of loan number between bad ones and good ones should be taken into consideration, in addition to the ratio of extra return (or cost) between good ones and bad ones.

Finally, aiming at obtaining the best model, our task is to train and tune models as well as their parameters so at to find the one that can leads maximum $\bar{R}$ in validating data.

4.2 Lasso

Lasso is the abbreviation of the term: least absolute shrinkage and selection operator. As a regression algorithm that adds an L1 norm to the calculation of the RSS minimization as a penalty constraint, it can not only do feature selection but also run regularization. The advantage of the L1 norm is that when the lambda is sufficiently large, some of the estimated coefficients can be narrowed down to exactly zero, then the variables with non-zero estimate coefficient are what the model selects out. It was initially proposed by Robert Tibshiran [50], the professor of statistics at Stanford University in 1996, based on Leo Breiman's Nonnegative Garrote (NNG) [6] proposed. Its regression formula can be written as the Formula (7) [49], of which the last part is L1 norm

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j| \quad (7)$$

Lasso makes full use of cross-validation, a method to make rotation estimations. Hastie T and Qian J give a wonderful package in R to implement this method [25]. In k-fold cross-validation, the initial dataset is divided into k sub-samples and the model would be run for k times repeated. In each time, one individual sub-sample is retained as validation dataset, and the other samples are used for training. Each sub-sample is verified once. With 10-fold cross validation, the lambda value with the least cross-validation error could be selected. Finally, according to the obtained lambda value, the whole data can be used to refit the model.

But Hastie T and Qian J do not take oversampling into consideration in the package Glmnet. If simply putting the oversampled training and validating dataset into the function cv.glmnet, validating data will contain the same bad loan instances with training data which is not consistent with the reality. Therefore, this paper revises cv.glmnet and cv.elnet. Codes are included in Appendix 2.

To find the best lasso model, two approaches below are implemented. First is oversampling the training dataset with the revised function cv.glment. In each fold, the dataset is split into a training dataset (90%) and a validating dataset (10%) and only sampling the training dataset to train the lasso model. Then selecting the lambda with minimum mean square error in validating dataset. Then using the default cutoff to calculate confusion matrix in validating dataset and to generate the average extra return. Second is changing cutoff rather than using oversampling. So the original cv.glmnet is used to select the best lambda. Then using the validating dataset to select the best cutoff that could result in minimum average extra return. Other steps are all same in these two methods, so the following content in this section only show result with the method changing cutoff.

Figure 10 shows the coefficient changes with different lambdas in this dataset. Each curve corresponds to a variable and some variables' curves are marked in the plot. The axis above indicates the number of nonzero coefficients at the current lambda. As lambda gets larger, more variables' coefficient approaches zero.

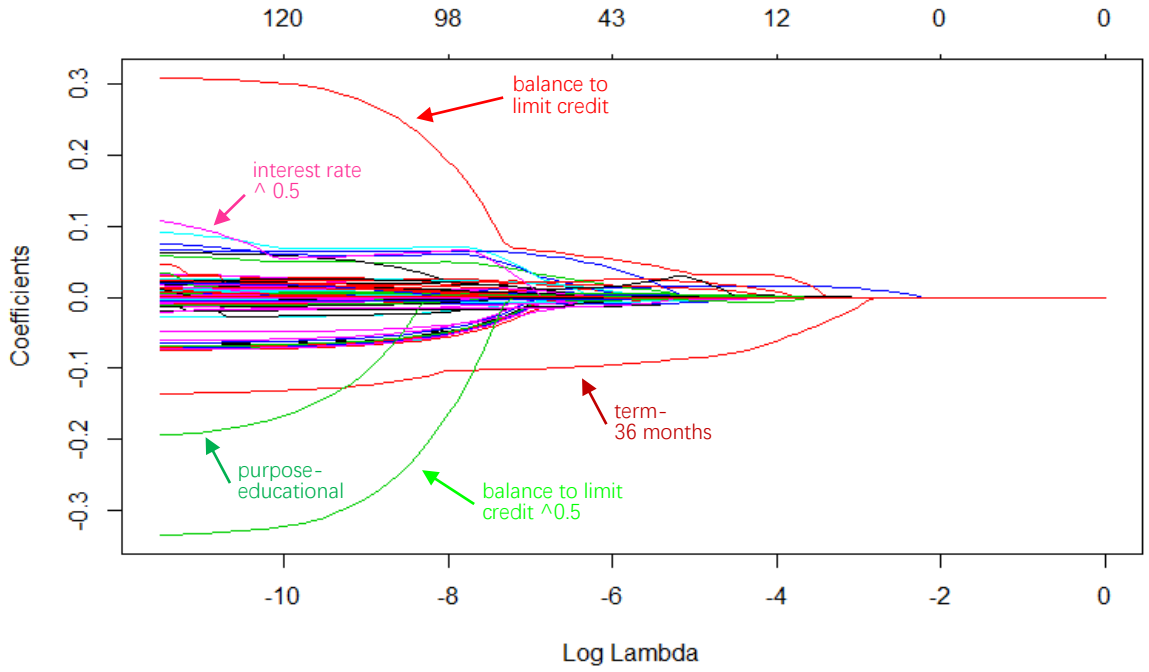

Figure 10. Coefficients with different lambda

Figure 11 shows the mean square errors corresponding to different lambdas. In this dataset, the lambda with the minimum MSE is 4.33*10^(-5).

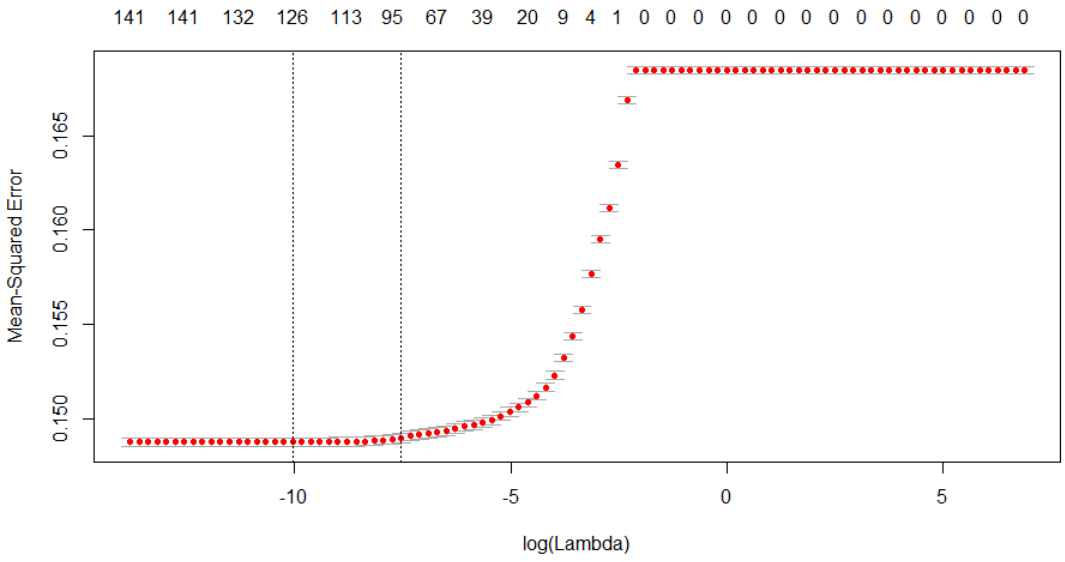

Figure 11. MSE with different Lambda

Although it will result in MSE of only 0.149, it is too small that only 26 out 164 variables are excluded (with 0 coefficient). When applying lambda with 0.01, the MSE increases to 0.151, but 126 variables could be excluded and the regression function is as Formula (8), where status = {1 if Good; 2 if Bad}. The positive coefficient means that when the value of this variable gets larger, the loan is more likely to be a bad loan, and vice versa.

$$
\begin{aligned}
100 * status = \ & 26.02 \\
+ \ & 3.20 \times [balance\ \ to\ \ limit\ \ of\ \ credit] \\
+ \ & 1.61 \times [interest\ \ rate] \\
+ \ & 0.78 \times [Ratio\ \ of\ \ satisfactory\ \ accounts] \\
+ \ & 0.30 \times [Number\ \ of\ \ trades\ \ opened\ \ in\ \ past\ \ 24\ \ months] \\
+ \ & 0.25 \times [ratio\ \ of\ \ monthly\ \ debt\ \ payments\ \ to\ \ income] \\
+ \ & 0.08 \times [Number\ \ of\ \ revolving\ \ trades\ \ with\ \ balance > 0] \\
+ \ & 0.01 \times [USA\ \ individual\ \ consumption\ \ of\ \ the\ \ year\ \ before\ \ the\ \ issued\ \ year] \\
+ \ & 0.001 \times [USA\ \ total\ \ saving\ \ of\ \ the\ \ year\ \ before\ \ the\ \ issued\ \ year] \\
- \ & \{8.01\ \ if\ \ term == 36\ \ months; 0\ \ if\ \ term == 60\ \ months\} \\
- \ & 0.59 \times [average\ \ current\ \ balance]^{0.2} \\
- \ & 0.27 \times [Number\ \ of\ \ mortgage\ \ accounts] \\
- \ & \{0.25\ \ if\ \ verification\ \ status = Not\ \ Verified;\ \ 0\ \ if\ \ verification\ \ status = Verified\} \\
- \ & \{0.13\ \ if\ \ grade = B; 0\ \ if\ \ else\ \ grades\} \\
- \ & 0.03 \times [FICO\ \ score] \\
- \ & 0.03 \times [USA\ \ CPI\ \ ratio\ \ of\ \ the\ \ year\ \ before\ \ the\ \ issued\ \ year] \\
+ \ & \begin{Bmatrix} 0.90\ \ if\ \ home_{ownership} = RENT; \\ -0.11\ \ if\ \ home_{ownership} = MORTGAGE; \\ 0\ \ \ if\ \ home_{ownership}\ \ is\ \ else \end{Bmatrix} \\
+ \ & 0\ \ \times value\ \ of\ \ else\ \ variables \qquad (8)
\end{aligned}
$$

With such a best lambda, the relationship between sensitivity, specificity and average extra return ratio is shown as Figure 12 with AUC of 0.725. The marked point is the one with highest extra return. This point corresponding to cutoff of 1.26.

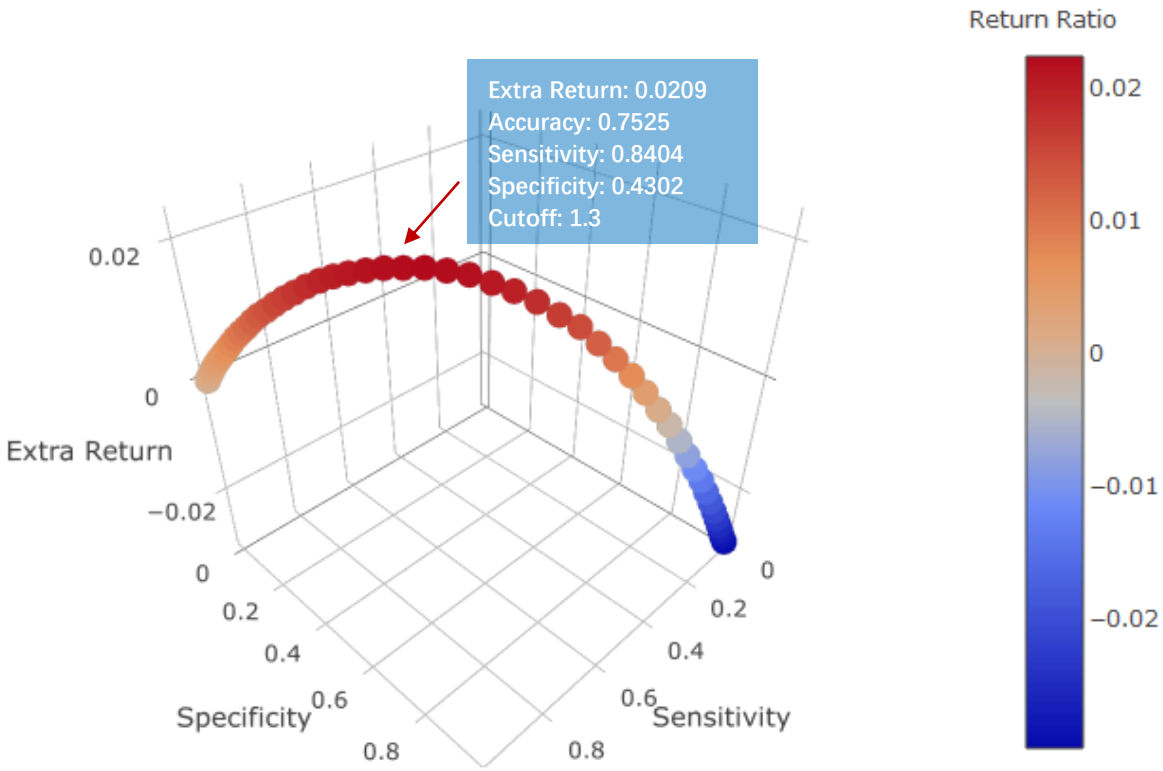

Figure 12. 3D plot of cutoff and average extra return ratio

4.3 Random Forest

Random forest is a data mining algorithm for both classification and regression proposed by Breiman [5]. Consisted of hundreds of decision tree, its output is determined by the average (regression) or mode (classification) of the outputs of individual trees. Not only can it give out accurate prediction, but also could rank the importance of features. Also, as an ensemble algorithm, it tends to be much more robust to changes and noise in the data. Small changes in the training dataset will have little, if any, impact on the final classifications made by the resulting model [53]. Randomness is one of the biggest characteristics of random forest, which contains two kinds of action: first is to randomly select a sample of the observations as training data when building its decision trees; second is to randomly select a subset of features be used at every split of decision tree.

Similar to Lasso, oversampling and changing cutoff are two approaches to address the unbalance problem. For the former one, only training dataset is oversampled and the cutoff is default; For the latter one, no dataset is oversampled and cutoff is selected to maximize the average extra return in validating dataset. Other steps are all same in these two methods, so the following content in this section only show result with the method oversampling.

Showing the error rates with various trees, Figure 13 (a) demonstrates that more trees can decrease the error rates thus improve performance, but the decreasing rate also slow down as more trees included. Figure 13 (b) is the list chart of validating dataset with this model.

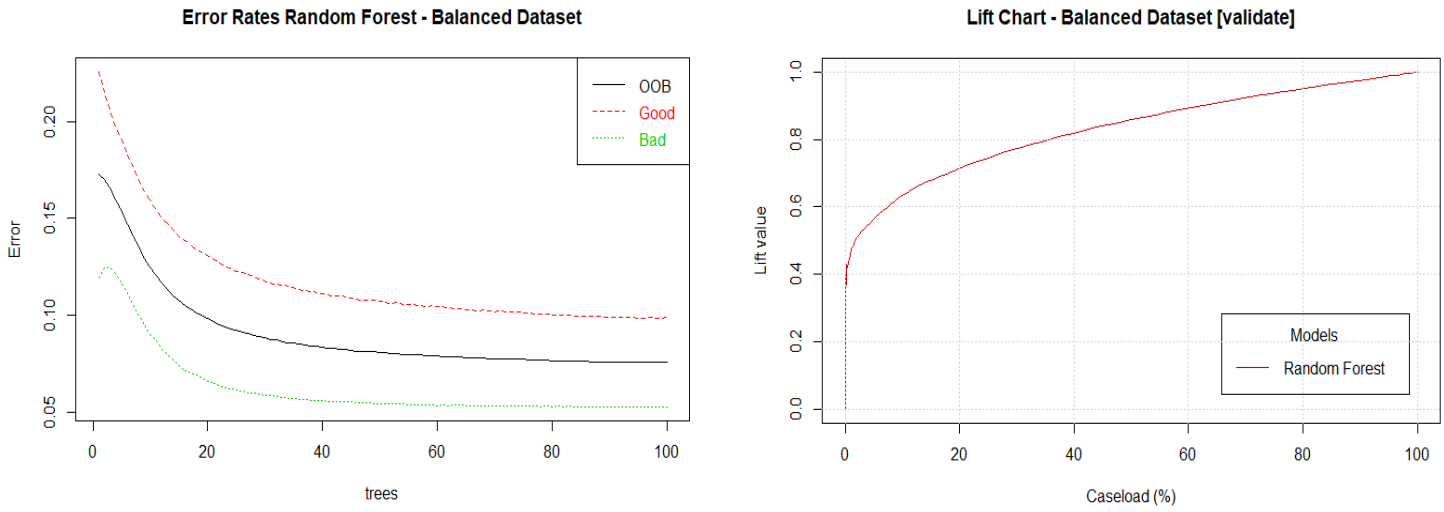

Figure 13. Plots of random forest: a) Error Rates; b) Lift Chart

4.4 Best model

Eight indexes are calculated to evaluate models' performance. They are MSE (Mean Square Error), MAPE (Mean Absolute Percentage Error), RMSE (Root Mean Square error), AUC (Area Under Curve), Accuracy, Sensitivity, Specificity and the Average Extra Return Ratio (See Table 3). Among these indexes, the better model is more likely to have lower value for the former three indexes and have higher value for the latter five indexes. This paper uses Average Extra Return Ratio as the measure to select the best model. In Figure 14, the right y axe is for Average Extra Return Ratio and the left y axe is for other seven indexes. Therefore, the random forest with unbalanced dataset (i.e. using changing cutoff rather than oversampling to solve the unbalance problem) is the best model in this research. The cutoff is 1.3, as Figure 15 shows.

Model	AUC	MSE	MAPE	RMSE	Acc-uracy	Sensi-tivity	Speci-ficity	Average Extra Return Ratio
Lasso oversampling	0.73	0.21	39.25	0.46	0.67	0.67	0.66	2.0880%
Lasso cutoff	0.73	0.15	22.98	0.39	0.71	0.75	0.6	2.0869%
RF oversampling	0.73	0.18	32.09	0.42	0.75	0.82	0.48	2.2493%
RF cutoff	0.73	0.15	23.47	0.39	0.73	0.78	0.53	**2.2973%**

Table 3. Model performance comparison

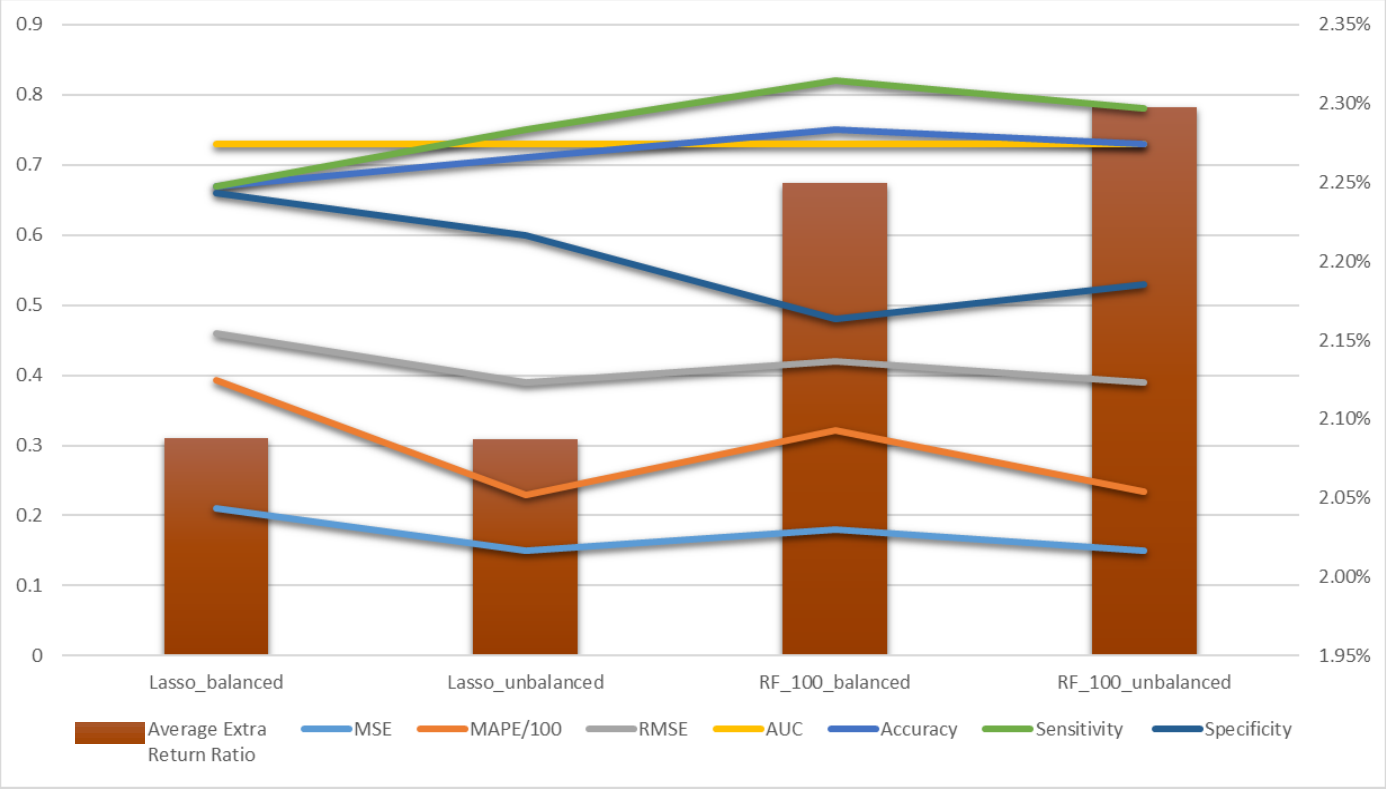

Figure 14. Model performance comparison

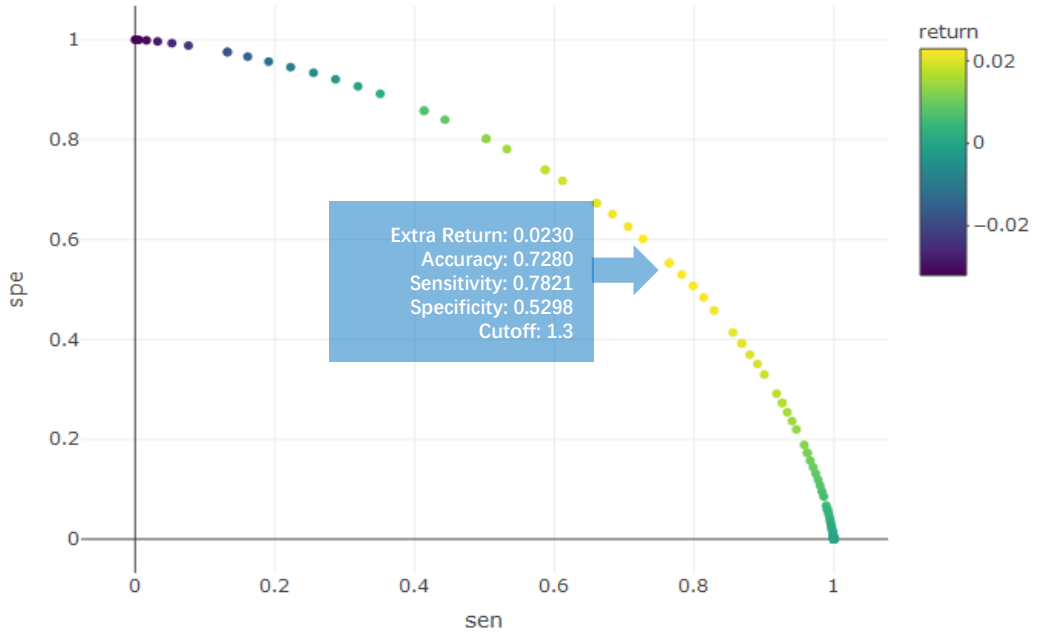

Figure 15. Plot of cutoff and average extra return ratio in best model

Figure 16 shows the top 20 important variables. It can be seen that **interest rate, average current balance, total credit limit and debt to income ratio** are the most important features to determine loan status.

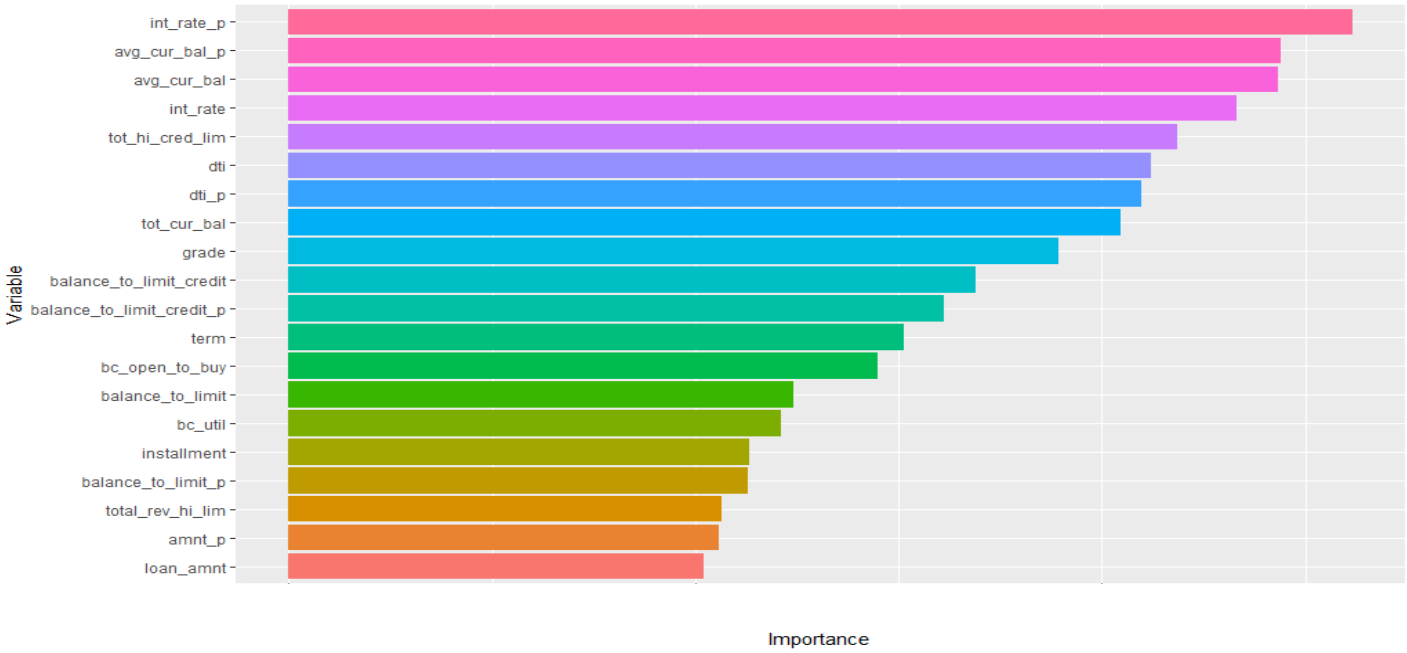

Figure 16. Top 20 important variables in best model

5 Business Conclusions

Last but not the least, the paper puts the testing data which never used before into the best model, i.e. random forest with cutoff 1.3. Loans in the testing dataset are all issued after August 2016 and all loans after August 2016 are in testing dataset. So this is consistent with reality that, when predicting future status of current issued loans, only information of loans issued before can be used to train the model.

There is a variable call "total payments" in the original raw dataset. This research excludes this variable when building the model because this is not what can be known before the loan funded, but when validating model's performance, this can be used to compute business return. Formulas are listed as (9).

$$Actual\ \ Return = \sum_{all\ \ loans} (total\ \ payment - loan\ \ amount) \quad (9)$$

$$Model\ \ Return = \sum_{loans\ \ precited\ \ good} (total\ \ payment - loan\ \ amount)$$

$$Savings = Model\ \ Return - Actual\ \ Return$$

$$Increasing\ \ Return\ \ Ratio = \frac{Model\ \ Return}{\sum_{loans\ \ precited\ \ good}(tloan\ \ amount)} - \frac{Actual\ \ Return}{\sum_{all\ \ loans}(loan\ \ amount)}$$

Table 4 is the confusion matrix. It can be seen that the accuracy is not good at all so this is the point that should be solved in the future work. But it turns out that **Savings is 64,568,053 and Increasing Return Ratio is 5.5%! That is to say, with this model, nearly 65million dollars of investment could be saved and investors return ratio can increase 5.5%.**

	Actual = Good	Actual = Bad
Predict = Good	31557 (46.8%)	2852 (4.2%)
Predict = Bad	25493 (37.8%)	7557 (11.2%)
Accuracy	58%	

Table 4. Confusion Matrix of testing data

More specifically, Figure 17 tells us that this model picked no loans with G grade and very few (less than 10%) loans with F or E grade. In reality, these loans do have so low return ratio (lower than -15%) that should not be accepted.

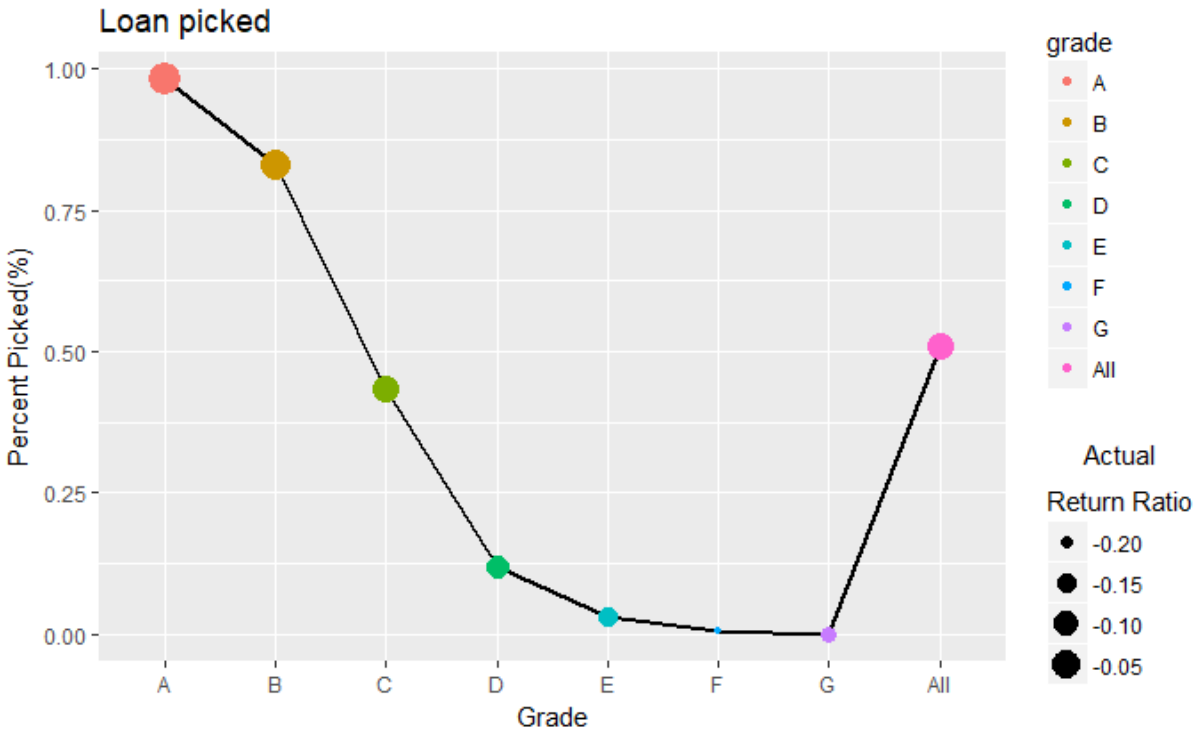

Figure 17. Loan picked ratio grouped by grade

This can also be illustrated in Figure 18 in which bars show the return in million dollars and lines show the return ratio, grouped by baseline and model. In baseline, all loans in the testing dataset are accepted; while in model, only loans predicted Good are accepted. The model improves the return ratio of all grades, especially poor grade loans where 13% improvement happens. Overall, the model increases about 65 million dollars return and 5.5% return ratio.

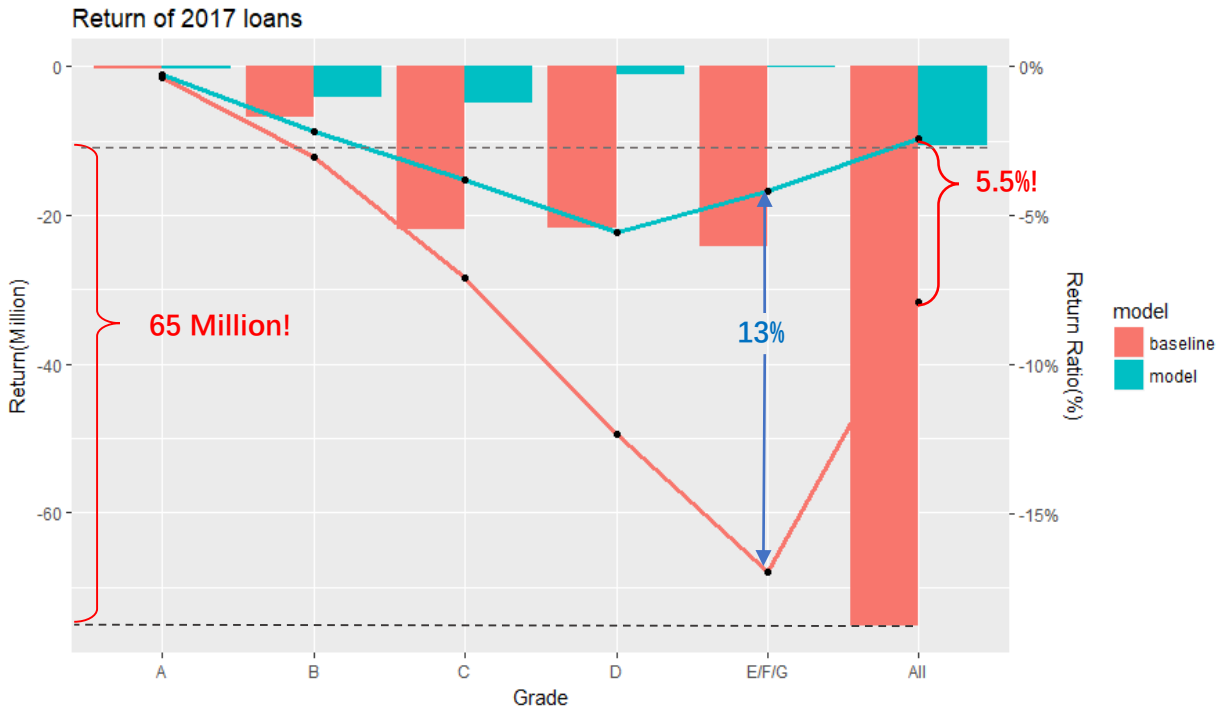

Figure 18. Return and Return Ratio of loans issued between 2016.8-2017.9, group by models

Note that one may be confused about why return could be negative 75 million. A reasonable explanation is due to the censoring dataset. This research only uses loans in status "Well Paid" or "Charged off", assumes these loans are in their final status (i.e. have been completed and will not be paid any more) and drops all loans in other status (Current, Late, Default and so on). Major loans are issued with the term of 36 months, so it is obvious that

many loans issued after August 2016 have not been completed and are in the status “Current”. The now completed loans are likely to be in two situations: one is charged off and with no future payment expected; the other one is well-prepaid already (but prepaid loans do not have too much interest return). In 2019, after major loans have been completed, more good loans will be included in the dataset thus the return will be more likely to be positive.

In conclusion, this research achieves the original two goals although some limitation exists (See chapter 6). Firstly, loan investors in Lending Club may use the best model in this paper to invest only loans predicted good. In the testing period, that could averagely increase their investment return for about 5.5% compared to invest loans randomly. Secondly, workers in Lending Club may use the best model in this paper to filter loans that are like to be charged off in the future. In the testing period, that would save investors for about 65 million dollars and raise their return for about 5.5%. Because the testing data is untouched when training and validating models, and all loans in training and validating are issued before the earliest issued loan in testing data, it is reasonable to conclude that this model could give similar performance in classifying future loans. Therefore, with obvious improvement in investment return, this research can benefit not only investors, but also the Lending Club, as more investors may be attracted to fund their money in the platform.

6 Limitation and Future Studies

Censoring data

Two kinds of data are unknown in this research. First is the information of rejected loans; Second is the information of current loans. This research only uses loans in status “Well Paid” or “Charged off”, and assumes these loans are in their final status (i.e. have been completed and will not be paid any more). So it is obvious that many loans issued after August 2016 have not been completed and are in the status “Current”. So the testing dataset only contain values partially known. If in 2019 when major loans issued in 2016 and 2017 have been completed, the function and the significance of this model could be judged more accurately.

Low accuracy of testing data

When validating data, the best model gives out 73% accuracy; while in testing dataset, although the return achieves about 65 million dollars, the accuracy is only 58%. With 0.55 sensitivity and 0.73 specificity, the model can recognize bad loans accurately, but is so strict to accept loans that many good loans are also be rejected. Because of time and technique limitation, not so many models are tried in this paper. In random forest, for example, only 100 trees with a set of mtrys be trained because 200+ trees result in out-of-memory in the author’s computer. Therefore, more improvement might be made when trying more models.

Lag Term of Economic Variables

This paper uses lag 1 economic variables in the dataset, like the United States GDP of the last year of the loan's issued date. Some of them are useful because they are selected by Lasso or Random Forest. So it is reasonable to put lag 2 or even lag 3 economic data to build models.

Evaluation Method

Rather than using RMSE, AUC or accuracy, one of the innovation of this paper is to use Average Extra Return Ratio as the measure to select best model when validating models. But in the process of training models, it is still minimizing MSE that be used as the objective function. Therefore, if revising the current models to use maximizing Average Extra Return Ratio as the objective function, the result may become more significant (more dollars may be saved).

Text Mining

In the original raw dataset, there is a variable called "desc" which is the loan's description written by the borrower. This paper simply drops this variable. But it might be helpful when applying text mining techniques.

7 Acknowledgement

As the author of this paper, I greatly thank Roger Bohn, Professor of Tech Management, UC San Diego, for instructions with big data analysis techniques, and for comments that significantly improved this research.

References

[1] Lending Club Statistics. 2018. Retrieved from https://www.lendingclub.com/info/demand-and-credit-profile.action

[2] Altman E I. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy[J]. The journal of finance, 1968, 23(4): 589-609.

[3] Odom M D, Sharda R. A neural network model for bankruptcy prediction[C]//Neural Networks, 1990., 1990 IJCNN International Joint Conference on. IEEE, 1990: 163-168.

[4] Hulme, Michael K., and Collette Wright. "Internet based social lending: Past, present and future." Social Futures Observatory 115 (2006).

[5] Klafft M. Online peer-to-peer lending: a lenders' perspective[J]. 2008.

[6] Herrero-Lopez S. Social interactions in P2P lending[C]//Proceedings of the 3rd Workshop on Social Network Mining and Analysis. ACM, 2009: 3.

[7] Berkovich E. Search and herding effects in peer-to-peer lending: evidence from prosper. com[J]. Annals of Finance, 2011, 7(3): 389-405.

[8] Gonzalez, Laura, and Yuliya Komarova Loureiro. "When can a photo increase credit? The impact of lender and borrower profiles on online peer-to-peer loans." Journal of Behavioral and Experimental Finance 2 (2014): 44-58.

[9] Riza Emekter, Yanbin Tu, Benjamas Jirasakuldech & Min Lu (2014) Evaluating credit risk and loan performance in online Peer-to-Peer (P2P) lending, Applied Economics, 47:1, 54-70, DOI: 10.1080/00036846.2014.962222

[10] Fatemeh Nemati Koutanaei, Hedieh Sajedi, Mohammad Khanbabaei, A hybrid data mining model of feature selection algorithms and ensemble learning classifiers for credit scoring, Journal of Retailing and Consumer Services, Volume 27, 2015, Pages 11-23, ISSN 0969-6989, https://doi.org/10.1016/j.jretconser.2015.07.003.

[11] Jin Y, Zhu Y. A data-driven approach to predict default risk of loan for online Peer-to-Peer (P2P) lending[C]//Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on. IEEE, 2015: 609-613.

[12] Wind Database. Retrieved from http://www.wind.com.cn/.

[13] Dealing with imbalanced data: undersampling, oversampling and proper cross-validation. (2015, August 17). Retrieved from https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation.

[14] Tibshirani, Robert. 1996. “Regression Shrinkage and Selection via the lasso”. Journal of the Royal Statistical Society. Series B (methodological) 58 (1). Wiley: 267–88.

[15] Breiman, Leo. Better Subset Regression Using the Nonnegative Garrote. Technometrics. 1995-11-01, 37 (4): 373–384.

[16] Tibshirani R, James G, Witten D, et al. An introduction to statistical learning-with applications in R[J]. 2013.

[17] Hastie T, Qian J. Glmnet vignette[J]. 2014.

[18] Breiman L. Random forests[J]. Machine learning, 2001, 45(1): 5-32.

[19] Williams G. Data mining with Rattle and R: The art of excavating data for knowledge discovery[M]. Springer Science & Business Media, 2011.

[20] Wright M N, Ziegler A. ranger: A fast implementation of random forests for high dimensional data in C++ and R[J]. arXiv preprint arXiv:1508.04409, 2015.

[21] glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models. (2018, April 2). Retrieved from https://CRAN.R-project.org/package=glmnet.

[22] Predict LendingClub’s Loan Data. Retrieved from https://rstudio-pubs-static.s3.amazonaws.com/203258_d20c1a34bc094151a0a1e4f4180c5f6f.html#model-comparison.

[23] Raffaella Calabrese (2014) Optimal cut-off for rare events and unbalanced misclassification costs, Journal of Applied Statistics, 41:8, 1678-1693, DOI: 10.1080/02664763.2014.888542

[24] How to handle Imbalanced Classification Problems in machine learning? (MARCH 17, 2017). Retrieved from https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/.

[25] Hyerim Kim. Cost Changes of Diabetes Treatment by Age Group and Region in South Korea. 2016.3

[26] Fei Ren, Sarah Raimi. Flu Trend Prediction in the World of Big Data. 2016.

[27] Zeyuan yu Long, Di Dai. Twitter Sentiments Analysis on Fitbit.

[28] Zhang Y, Jia H, Diao Y, et al. Research on credit scoring by fusing social media information in online peer-to-peer lending[J]. Procedia Computer Science, 2016, 91: 168-174.

[29] Yap B W, Ong S H, Husain N H M. Using data mining to improve assessment of credit worthiness via credit scoring models[J]. Expert Systems with Applications, 2011, 38(10): 13274-13283.

[30] Geng R, Bose I, Chen X. Prediction of financial distress: An empirical study of listed Chinese companies using data mining[J]. European Journal of Operational Research, 2015, 241(1): 236-247.

[31] Hooman A, Marthandan G, Yusoff W F W, et al. Statistical and data mining methods in credit scoring[J]. The Journal of Developing Areas, 2016, 50(5): 371-381.

[32] Gahlaut A, Singh P K. Prediction analysis of risky credit using Data mining classification models[C]//Computing, Communication and Networking Technologies (ICCCNT), 2017 8th International Conference on. IEEE, 2017: 1-7.

[33] Shmueli G, Patel N R, Bruce P C. Data mining for business intelligence: Concepts, techniques, and applications in Microsoft Office Excel with XLMiner[M]. John Wiley and Sons, 2011.

[34] Roger Bohn. BigData@UCSD. 2018. Retrieved from https://bda2020.wordpress.com/

[35] Wright M N, Ziegler A. ranger: A fast implementation of random forests for high dimensional data in C++ and R[J]. arXiv preprint arXiv:1508.04409, 2015.

[36] (2017). readr: Read Rectangular Text Data. R package version 1.1.1. https://CRAN.R-project.org/package=readr.

[37] Hadley Wickham, Romain Fran<U+00E7>ois, Lionel Henry and Kirill Müller (2018). dplyr: A Grammar of Data Manipulation. R package version 0.7.5. https://CRAN.R-project.org/package=dplyr.

[38] Hadley Wickham and Lionel Henry (2018). tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions. R package version 0.8.1. https://CRAN.R-project.org/package=tidyr.

[39] Nicola Lunardon, Giovanna Menardi, and Nicola Torelli (2014). ROSE: a Package for Binary Imbalanced Learning. R Journal, 6(1), 82-92.

[40] Hyndman R, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E and Yasmeen F (2018). _forecast: Forecasting functions for time series and linear models_. R package version 8.3, <URL: http://pkg.robjhyndman.com/forecast>.

[41] Hyndman RJ and Khandakar Y (2008). "Automatic time series forecasting: the forecast package for R." _Journal of Statistical Software_, *26*(3), pp. 1-22. <URL: http://www.jstatsoft.org/article/view/v027i03>.

[42] Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

[43] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez and Markus Müller (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics, 12, p. 77. DOI: 10.1186/1471-2105-12-77 http://www.biomedcentral.com/1471-2105/12/77/

[44] Sing T, Sander O, Beerenwinkel N and Lengauer T (2005). "ROCR: visualizing classifier performance in R." _Bioinformatics_, *21*(20), pp. 7881. <URL: http://rocr.bioinf.mpi-sb.mpg.de>.

[45] H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.

[46] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2018). caret: Classification and Regression Training. R package version 6.0-79. https://CRAN.R-project.org/package=caret

[47] Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.

[48] Friedrich Leisch & Evgenia Dimitriadou (2010). mlbench: Machine Learning Benchmark Problems. R package version 2.1-1.

[49] Terry Therneau and Beth Atkinson (2018). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-13. https://CRAN.R-project.org/package=rpart

[50] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

[51] Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec and Pedro Despouy (2017). plotly: Create Interactive Web Graphics via 'plotly.js'. R package version 4.7.1. https://CRAN.R-project.org/package=plotly

[52] D. Kahle and H. Wickham. ggmap: Spatial Visualization with ggplot2. The R Journal, 5(1), 144-161. URL http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf

[53] Original S code by Richard A. Becker, Allan R. Wilks. R version by Ray Brownrigg. Enhancements by Thomas P Minka and Alex Deckmyn. (2018). maps: Draw Geographical Maps. R package version 3.3.0. https://CRAN.R-project.org/package=maps

[54] Original S code by Richard A. Becker and Allan R. Wilks. R version by Ray Brownrigg. (2018). mapdata: Extra Map Databases. R package version 2.3.0. https://CRAN.R-project.org/package=mapdata

[55] William Murphy (2016). fiftystater: Map Data to Visualize the Fifty U.S. States with Alaska and Hawaii Insets. R package version 1.0.1. https://CRAN.R-project.org/package=fiftystater

[56] Hadley Wickham (2018). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.3.1. https://CRAN.R-project.org/package=stringr

[57] Wind Information Co. and Ltd (2014). WindR: Financial Data API of Wind Information Co., Ltd. R package version 1.1.

[58] H. Wickham. Reshaping data with the reshape package. Journal of Statistical Software, 21(12), 2007.

[59] Hadley Wickham (2017). tidyverse: Easily Install and Load the 'Tidyverse'. R package version 1.2.1. https://CRAN.R-project.org/package=tidyverse

[60] Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL http://www.jstatsoft.org/v40/i03/.

[61] Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra

[62] Hadley Wickham (2017). scales: Scale Functions for Visualization. R package version

0.5.0. https://CRAN.R-project.org/package=scales

[63] Picture in the cover is derived from http://www.infinitaccounting.com/blog/5-ways-to-prepare-your-company-for-a-business-loan/

Appendix

Appendix 1. Data Dictionary

	Variable Name	Description	Sample Instance
1	acc_now_delinq	The number of accounts on which the borrower is now delinquent.	0
2	acc_open_past_24mths	Number of trades opened in past 24 months.	8
3	amnt_p	loan_amount^0.5	97.97958971
4	annual_inc	Annual Income	56017
5	avg_cur_bal	Average current balance of all accounts	3214
6	avg_cur_bal_p	avg_cur_bal^0.2	5.028160992
7	balance_to_limit	Ratio of total current balance to high credit/credit limit	69.2
8	balance_to_limit_p	balance_to_limit^0.8	0.523471137
9	balance_to_limit_credit	Average balance to limit for all accounts	0.723513053
10	balance_to_limit_credit_p	balance_to_limit_credit^0.5	29.65410873
11	bc_open_to_buy	Total open to buy on revolving bankcards.	6494
12	bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts.	69.2
13	chargeoff_within_12_mths	Number of charge-offs within 12 months	0
14	collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections	0
15	delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years	2
16	delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.	0
17	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.	25.81
18	dti_p	dti^0.3	2.651773348

19	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.	10+ years
20	fico_range_low	The lower boundary range the borrower's FICO at loan origination belongs to.	680
21	grade	LC assigned loan grade	RENT
22	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	f
23	initial_list_status	The initial listing status of the loan. Possible values are – W, F	0
24	inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)	326.53
25	installment	The monthly payment owed by the borrower if the loan originates.	13.66
26	int_rate	Interest Rate on the loan	3.695943723
27	int_rate_p	int_rate^0.5	9600
28	loan_amnt	loan amount	10.9334105
29	log_fico	log(fico score)	1.098612289
30	log_income	log(income)	1.098612289
31	log_mths_account	log(mths_account)	1.945910149
32	log_num_account_12m	log(num_account_12m)	183
33	log_num_rev_trades	log(num_rev_trades)	265
34	mo_sin_old_il_acct	Months since oldest bank installment account opened	23
35	mo_sin_old_rev_tl_op	Months since oldest revolving account opened	3
36	mo_sin_rcnt_rev_tl_op	Months since most recent revolving account opened	265
37	mo_sin_rcnt_tl	Months since most recent account opened	0
38	month_cr_line	months since the borrower's earliest reported credit line was opened	3
39	mort_acc	Number of mortgage accounts.	24
40	mths_account	Months since most recent account opened	47

41	mths_since_recent_bc	The monthly payment owed by the borrower if the loan originates.	3
42	num_account	Number of account s(mortgage accounts, currently active bankcard accounts, installment accounts, revolving accounts)	4.66486568
43	num_account_p	num_account^0.4	0
44	num_account_12m	Number of accounts opened in past 12 months (sum of Number of installment accounts opened in past 12 months, Number of accounts opened in past 12 months)	4
45	num_accts_ever_120_pd	Number of accounts ever 120 or more days past due	7
46	num_actv_bc_tl	Number of currently active bankcard accounts	5
47	num_actv_rev_tl	Number of currently active revolving trades	16
48	num_bc_sats	Number of satisfactory bankcard accounts	17
49	num_bc_tl	Number of bankcard accounts	8
50	num_il_tl	Number of installment accounts	26
51	num_op_rev_tl	Number of open revolving accounts	7
52	num_rev_accts	Number of revolving accounts	7
53	num_rev_tl_bal_gt_0	Number of revolving trades with balance >0	12
54	num_rev_trades	Number of revolving trades (sum of Number of currently active revolving trades, Number of revolving trades with balance >0)	0
55	num_sats	Number of satisfactory accounts	0
56	num_tl_120dpd_2m	Number of accounts currently 120 days past due (updated in past 2 months)	0
57	num_tl_30dpd	Number of accounts currently 30 days past due (updated in past 2 months)	3
58	num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months	12
59	num_tl_op_past_12m	Number of accounts opened in past 12 months	1.64375183

60	open_acc	The number of open credit lines in the borrower's credit file.	100
61	open_acc_p	open_acc^0.2	60
62	pct_tl_nvr_dlq	Percent of trades never delinquent	0
63	percent_bc_gt_75	Percentage of all bankcard accounts > 75% of limit.	0
64	pub_rec	Number of derogatory public records	debt_consolidation
65	pub_rec_bankruptcies	Number of public record bankruptcies	0.235289855
66	ratio_bc_open_to_by	Total open to buy on revolving bankcards/Total revolving high credit/credit limit	0.255319149
67	ratio_sats	Ratio of satisfactory accounts (Number of satisfactory accounts/Number of accounts)	16388
68	revol_bal	Total credit revolving balance	59.4
69	revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.	11.59502974
70	revol_util_p	revol_util^0.6	59467
71	states_income	average annual income in the issue state of the last month before the issue date	6.6
72	states_unemp	unemployment rate in the issue state of the last year before the issue date	Good
73	tax_liens	Number of tax liens	0
74	term	loan term	36 months
75	tot_coll_amt	Total collection amounts ever owed	0
76	tot_cur_bal	Total current balance of all accounts	38566
77	tot_hi_cred_lim	Total high credit/credit limit	52490
78	total_acc	The total number of credit lines currently in the borrower's credit file	87
79	total_bal_ex_mort	Total credit balance excluding mortgage	38566
80	total_bc_limit	Total bankcard high credit/credit limit	21100
81	total_il_high_credit_limit	Total installment high credit/credit limit	24890

82	total_rev_hi_lim	Total revolving high credit/credit limit	27600
83	USA_CPI	CPI rate of USA of the last year before the issue date	1.3
84	USA_GDP	Total GDP of USA of the last year before the issue date	17622.3
85	USA_ind_consume	Individual consumption in USA of the last year before the issue date	11055.4
86	USA_ind_GDP	Individual GDP of USA of the last year before the issue date	55198
87	USA_int_rate	month average of 10-year Treasury Yield of the last month before the issue date	0.445555556
88	USA_net_saving	Net savings of USA of the last year before the issue date	663.6
89	USA_stock	month average of S&P 500 of the last month before the issue date	2044.572105
90	USA_total_saving	Total savings of USA of the last year before the issue date	3426.3
91	USA_unemp	Unemployment rate of USA of the last year before the issue date	5.8
92	verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified	Source Verified
93	status	Loan status	Good

Appendix 2. R Code

Data Cleaning

```
rm(list = ls())
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
library(readr)
library(dplyr)
library(tidyr)

df_original = read.csv('z-accepted_2007_to_2017.csv')[-1,]
df = df_original
df = filter(df,loan_status %in% c('Charged Off','Fully Paid'))
df = mutate(df,status=ifelse(loan_status=='Fully Paid',"Good","Bad"))

### Feature Creation
df$loan_status=NULL
df$date = df$issue_d
library(lubridate)
Date = dmy(paste("01-", df$date , sep =""))
df$date = substr(Date,1,7)
df['year'] = substr(Date,1,4)
df['month'] = substr(Date,6,7)
l_combine = c()
l_var = c('mo_sin_rcnt_rev_tl_op','mo_sin_rcnt_tl','mths_since_rcnt_il')
l_combine = c(l_combine,l_var)
df1 = df[l_var]
df1[is.na(df1)]=999999999
df['mths_account']=apply(df1, 1, min)
df = df[which(df$mths_account!=999999999),]
l_var = c('mths_since_last_major_derog','mths_since_recent_bc_dlq',
          'mths_since_recent_revol_delinq','mths_since_last_delinq')
l_combine = c(l_combine,l_var)
df1 = df[l_var]
df1[is.na(df1)]=999999999
df['mths_bad']=apply(df1, 1, min)
df$mths_bad[which(df$mths_bad==999999999)]=NA
df = df[-which(is.na(df$il_util) & is.na(df$bc_util)),]
l_var = c('il_util','bc_util')
l_combine = c(l_combine,l_var)
df1 = df[l_var]
x = apply(df1, 2, as.numeric, na.rm = TRUE)
df['balance_to_limit']=apply(x, 1, mean, na.rm = TRUE)
l_var = c('num_actv_bc_tl','num_il_tl','num_rev_accts','mort_acc')
l_combine = c(l_combine,l_var)
df1 = df[l_var]
x = apply(df1, 2, as.numeric, na.rm = TRUE)
df['num_account']=apply(x, 1, sum, na.rm = TRUE)
l_var = c('num_actv_rev_tl','num_rev_tl_bal_gt_0')
```

```
l_combine = c(l_combine,l_var)
df1 = df[l_var]
df['num_rev_trades']=apply(df1, 1, min)
l_var = c('open_il_12m','num_tl_op_past_12m')
l_combine = c(l_combine,l_var)
df1 = df[l_var]
x = apply(df1, 2, as.numeric, na.rm = TRUE)
df['num_account_12m']=apply(x, 1, sum, na.rm = TRUE)
df = df[-which(is.na(df$bc_open_to_buy)),]
l_combine=c(l_combine,c('bc_open_to_buy','num_sats'))
df['ratio_bc_open_to_by']=df$bc_open_to_buy/df$total_rev_hi_lim
df['ratio_sats']=df$num_sats/df$num_account
l_var = c('revol_bal','total_bal_ex_mort','total_bal_il')
l_combine = c(l_combine,l_var)
df1 = df[l_var]
x = apply(df1, 2, as.numeric, na.rm = TRUE)
total_balance = apply(x, 1, sum, na.rm = TRUE)
l_var = c('tot_hi_cred_lim','total_il_high_credit_limit','total_rev_hi_lim')
l_combine = c(l_combine,l_var)
df1 = df[l_var]
x = apply(df1, 2, as.numeric, na.rm = TRUE)
total_limit = apply(x, 1, sum, na.rm = TRUE)
df = mutate(df,
            log_income = log(annual_inc),
            dti_p = dti^0.3,
            fico_range_high = fico_range_high%>%as.numeric(),
            balance_to_limit_credit = total_balance/total_limit,
            annual_inc = annual_inc%>%as.numeric()
            log_fico = log(fico_range_low),
            amnt_p = funded_amnt^0.5,
            int_rate_p = (df$int_rate%>%as.numeric())^0.5,
            date = as.factor(date),
            open_acc_p = open_acc^0.2,
            revol_util_p = (revol_util%>%as.numeric())^0.6,
            avg_cur_bal_p = (avg_cur_bal%>%as.numeric())^0.2,
            log_mths_account = log(mths_account),
            log_mths_bad = log(mths_bad),
            balance_to_limit_p = balance_to_limit^0.8,
            num_account_p = num_account^0.4,
            log_num_rev_trades = log(num_rev_trades),
            log_num_account_12m = log(num_account_12m),
            balance_to_limit_credit_p = balance_to_limit_credit^0.5)
df[df==-Inf] = 0
df$payment = df$total_pymnt
l_joint = c('revol_bal_joint','sec_app_fico_range_low','sec_app_fico_range_high',
            'sec_app_earliest_cr_line','sec_app_inq_last_6mths','sec_app_mort_acc',
            'sec_app_open_acc','sec_app_revol_util','sec_app_open_act_il',
            'sec_app_num_rev_accts','sec_app_chargeoff_within_12_mths',
            'sec_app_collections_12_mths_ex_med','sec_app_mths_since_last_major_derog',
            'application_type','annual_inc_joint','dti_joint','verification_status_joint')
l_after = c('hardship_flag','hardship_type','hardship_reason','hardship_status',
            'deferral_term','hardship_amount','hardship_start_date','hardship_end_date',
            'payment_plan_start_date','hardship_length','hardship_dpd','hardship_loan_status',
```

```
            'orig_projected_additional_accrued_interest','hardship_payoff_balance_amount',
            'hardship_last_payment_amount','disbursement_method','debt_settlement_flag',
            'debt_settlement_flag_date','settlement_status','settlement_date',
            'settlement_amount','settlement_percentage','settlement_term',
            'out_prncp','out_prncp_inv','total_pymnt','pymnt_plan',
            'total_pymnt_inv','total_rec_prncp','total_rec_int','total_rec_late_fee',
            'recoveries','collection_recovery_fee','last_pymnt_d','last_pymnt_amnt',
            'next_pymnt_d','last_credit_pull_d','last_fico_range_high','last_fico_range_low')
l_useless = c('id', 'member_id', 'funded_amnt', 'funded_amnt_inv',
              'zip_code', 'policy_code', 'mths_since_recent_inq')
l_other = c('emp_title','desc','title')
l_drop = c(l_joint,l_after,l_useless,l_other)
df[l_drop]=NULL

df$addr_state = factor(df$addr_state)
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
df_usa = read.csv('econ_usa.csv')[,-1]
df_States_income = read.csv('econ_States_income.csv')[,-1]
df_States_unemp = read.csv('econ_States_unemp.csv')[,-1]
df_1 = df[,c('date','status','year','addr_state')]
df_1 = unite(df_1,year_state,year,addr_state,sep='_',remove=FALSE)
df_1 = unite(df_1,date_state,date,addr_state,sep='_',remove=FALSE)
df_1 = mutate(df_1,
              USA_int_rate = df_usa[,'df_USA_int_rate'][match(date, df_usa$date)],
              USA_stock = df_usa[,'df_USA_stock'][match(date, df_usa$date)],
              USA_unemp = df_usa[,'USA_unemp'][match(date, df_usa$date)],
              USA_ind_consume = df_usa[,'USA_ind_consume'][match(date, df_usa$date)],
              USA_CPI = df_usa[,'USA_CPI'][match(date, df_usa$date)],
              USA_total_saving = df_usa[,'USA_total_saving'][match(date, df_usa$date)],
              USA_net_saving = df_usa[,'USA_net_saving'][match(date, df_usa$date)],
              USA_GDP = df_usa[,'USA_GDP'][match(date, df_usa$date)],
              USA_ind_GDP = df_usa[,'USA_ind_GDP'][match(date, df_usa$date)],
              states_income = df_States_income[,'average_income']
[match(df_1$year_state,df_States_income$year_state)],
              states_unemp = df_States_unemp[,'unemp']
[match(date_state,df_States_unemp$date_state)])
df_2 = df_1[,7:length(df_1)]
df_2 = as.data.frame(sapply(df_2, as.numeric))
df_3 = cbind.data.frame(df,df_2, stringsAsFactors = TRUE)
df = df_3

l_missing = c()
l_missing_n = c()
i=1
while (i<=length(colnames(df))) {
  name = colnames(df)[i]
  x = df[,name]
  if (sum(is.na(x))>100) {
    l_missing = c(l_missing,name)
    l_missing_n = c(l_missing_n,sum(is.na(x)))
  }
  i=i+1
}
```

```
df_missing = as.data.frame(tibble(variable=l_missing,missing_n = l_missing_n))
l_missing_2 = l_missing[c(1:17,19:20,24:25)]
df[l_missing_2]=NULL

l_categs = c()
l_categs_n = c()
i=1
while (i<=length(colnames(df))) {
  name = colnames(df)[i]
  x = df[,name]
  if (length(levels(x))>=30 | type_sum(x)=='chr') {
    l_categs = c(l_categs,name)
    l_categs_n = c(l_categs_n,length(levels(x)))
  }
  i=i+1
}
df_categs = data.frame(l_categs,l_categs_n)
df$month = as.factor(df$month)
df$year = as.factor(df$year)
df$status = as.factor(df$status)
df$total_acc = as.numeric(df$total_acc)
df$delinq_2yrs = as.numeric(df$delinq_2yrs)
earliest_cr_line = df$earliest_cr_line
earliest_cr_line = dmy(paste("01-", df$earliest_cr_line , sep =""))
monnb <- function(d) {
  lt <- as.POSIXlt(as.Date(d, origin="1900-01-01"))
  lt$year*12 + lt$mon }
mondf <- function(d1, d2) { monnb(d2) - monnb(d1) }
Date = as.Date(paste(df$date,'-01',sep=''))
df$month_cr_line = mondf(earliest_cr_line,Date)
df['earliest_cr_line']=NULL
df$sub_grade=NULL
df = na.omit(df)
df$issue_d = NULL


write.csv(df,'0608_CleanData_Full.csv')
df_clean_0608 = df
saveRDS(df_clean_0608, file = "cleanData0608.rds")
```

Economic Data Extracting

```
library(WindR)
w.start()
library(readr)
library(dplyr)
library(tidyr)

multiMerge = function(l,by){
  df_merge = l[[1]]
```

```
  l[[1]]=NULL
  for (x in l) {
    df_merge = merge(df_merge,x,by=by)
  }
  return(df_merge)
}

l_states_names = c('AL','AK','AZ','AR','CA','CO','CT','DE','DC','FL',
                   'GA','HI','ID','IL','IN','IA','KS','KY','LA','ME','MD',
                   'MA','MI','MN','MS','MO','MT','NE','NV','NH','NJ',
                   'NM','NY','NC','ND','OH','OK','OR','PA','RI','SC',
                   'SD','TN','TX','UT','VT','VA','WA','WV','WI','WY')
# States unemployment rate
l_number = 'G1125905'
for (i in 1:50) {
  number = 1125905+i
  symbol = paste('G',number,sep='')
  l_number = paste(l_number,symbol,sep=',')
}
States_unemp<-w.edb(l_number,'2011-12-01','2018-05-31','Fill=Previous')
df_States_unemp = States_unemp$Data
colnames(df_States_unemp)[1]='date'
colnames(df_States_unemp)[2:52]=paste('States_unemp',l_states_names,sep='_')
# States annuall income
l_number = 'G1127078'
for (i in 1:50) {
  number = 1127078+i
  symbol = paste('G',number,sep='')
  l_number = paste(l_number,symbol,sep=',')
}
States_income<-w.edb(l_number,'2011-01-01','2018-05-31','Fill=Previous')
df_States_income = States_income$Data
colnames(df_States_income)[1]='date'
colnames(df_States_income)[2:52]=paste('States_income',l_states_names,sep='_')
df_States_income = separate(df_States_income,date,c('year','month','day'))
df_States_income[,c('month','day')]=NULL
colnames(df_States_income)[2:52]=l_states_names
colnames(df_States_unemp)[2:52]=l_states_names
Date = df_States_unemp$date[-1]
Date = substr(Date,1,7)
df_States_unemp=df_States_unemp[-nrow(df_States_unemp),]
df_States_unemp$date = Date
df_States_income$year=c('2012','2013','2014','2015','2016','2017')

df_States_income_new = gather(df_States_income,key='state',value='average_income',-year)
df_States_income_new = unite(df_States_income_new,year_state,year,state,sep='_',remove=FALSE)
df_States_unemp_new = gather(df_States_unemp,key='state',value='unemp',-date)
df_States_unemp_new = unite(df_States_unemp_new,date_state,date,state,sep='_',remove=FALSE)
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
write.csv(df_States_income_new,'econ_States_income.csv')
write.csv(df_States_unemp_new,'econ_States_unemp.csv')

df_USA_int_rate = w.edb('G1147401','2011-12-01','2018-05-31','Fill=Previous')$Data
```

```
df_USA_int_rate = separate(df_USA_int_rate,DATETIME,c('year','month','day'))
df_USA_int_rate = unite(df_USA_int_rate,year-month,year,month,sep='-',remove=FALSE)
df_USA_int_rate = tapply(df_USA_int_rate$CLOSE, df_USA_int_rate$`year - month`, mean)
df_USA_int_rate = as.data.frame(df_USA_int_rate)
df_USA_int_rate['date']=row.names(df_USA_int_rate)
df_USA_stock = w.edb('G0001672','2011-12-01','2018-05-31','Fill=Previous')$Data
df_USA_stock = separate(df_USA_stock,DATETIME,c('year','month','day'))
df_USA_stock = unite(df_USA_stock,year-month,year,month,sep='-',remove=FALSE)
df_USA_stock = tapply(df_USA_stock$CLOSE, df_USA_stock$`year - month`, mean)
df_USA_stock = as.data.frame(df_USA_stock)
df_USA_stock['date']=row.names(df_USA_stock)
USA_unemp = w.edb('G0000067','2011-12-01','2018-05-31','Fill=Previous')$Data
df_USA_unemp = data.frame('USA_unemp'=USA_unemp$CLOSE,
                          'date'=substr(USA_unemp$DATETIME,1,7))
USA_ind_consume = w.edb('G0003891','2011-12-01','2018-05-31','Fill=Previous')$Data
df_USA_ind_consume = data.frame(USA_ind_consume=USA_ind_consume$CLOSE,
                                'date'=substr(USA_ind_consume$DATETIME,1,7))
USA_CPI = w.edb('G0000027','2011-12-01','2018-05-31','Fill=Previous')$Data
df_USA_CPI = data.frame('USA_CPI'=USA_CPI$CLOSE,
                        'date'=substr(USA_CPI$DATETIME,1,7))
l = list(df_USA_unemp,df_USA_ind_consume,df_USA_CPI,df_USA_stock,df_USA_int_rate)
df_States_monthly = multiMerge(l,'date')
Date = df_States_monthly$date[-1]
df_States_monthly=df_States_monthly[-nrow(df_States_monthly),]
df_States_monthly$date = Date
USA_total_saving = w.edb('G1109599','2011-12-01','2018-05-31','Fill=Previous')$Data
df_USA_total_saving = data.frame(USA_total_saving=USA_total_saving$CLOSE,
                                 'date'=substr(USA_total_saving$DATETIME,1,7))
USA_net_saving = w.edb('G1109600','2011-12-01','2018-05-31','Fill=Previous')$Data
df_USA_net_saving = data.frame(USA_net_saving=USA_net_saving$CLOSE,
                                 'date'=substr(USA_net_saving$DATETIME,1,7))
USA_GDP = w.edb('G0000003','2011-12-01','2018-05-31','Fill=Previous')$Data
df_USA_GDP = data.frame(USA_GDP=USA_GDP$CLOSE,
                                 'date'=substr(USA_GDP$DATETIME,1,7))
USA_ind_GDP = w.edb('G1138044','2011-12-01','2018-05-31','Fill=Previous')$Data
df_USA_ind_GDP = data.frame(USA_ind_GDP=USA_ind_GDP$CLOSE,
                                 'date'=substr(USA_ind_GDP$DATETIME,1,7))
l = list(df_USA_total_saving,df_USA_net_saving,df_USA_GDP,df_USA_ind_GDP)
df_States_quarterly = multiMerge(l,'date')
Date = df_States_quarterly$date[-1]
df_States_quarterly=df_States_quarterly[-nrow(df_States_quarterly),]
df_States_quarterly$date = Date
USA_total_saving = rep(df_States_quarterly$USA_total_saving,each=3)
USA_net_saving = rep(df_States_quarterly$USA_net_saving,each=3)
USA_GDP = rep(df_States_quarterly$USA_GDP,each=3)
USA_ind_GDP = rep(df_States_quarterly$USA_ind_GDP,each=3)
Date = df_States_monthly$date[1:75]
df_States_quarterly = data.frame(Date,USA_total_saving,USA_net_saving,USA_GDP,USA_ind_GDP)
colnames(df_States_quarterly)[1]='date'
df_usa = merge(df_States_monthly,df_States_quarterly,by='date')
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
write.csv(df_usa,'econ_usa.csv')
```

Mapping

```
rm(list = ls())
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
library(ggplot2)
library(ggmap)
library(maps)
library(mapdata)
library(ggplot2)
library(fiftystater)
library(readr)
library(dplyr)
library(tidyr)
library(stringr)

### Map
usa <- map_data("usa")
states <- map_data("state")
ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE)  # do this to leave off the color legend

Df = read.csv('0531_CleanData_train&validate.csv')[,-1]
df=Df
set.seed=100
df = df[c('status','addr_state','year','month','date')]
Date = as.Date(paste(df$date,"-01", sep =""))
df$date = Date
df_test = df[df$date>=as.Date('2016-08-01'),]
df = df[df$date<as.Date('2016-08-01'),]
DrawMap = function(dff,year_number=''){
  if (nchar(year_number)>4) {
    df_map = dff
  } else{
    df_map = dff[dff$year==year_number,]
  }
  map_data = as.data.frame(table(df_map$status,df_map$addr_state))
  colnames(map_data)=c('status','region','freq')
  x = split(map_data,map_data$status)
  df1 = x['1Good']$`1Good`
  rownames(df1) = df1$region
  df1 = df1['freq']
  colnames(df1) = '1Good'
  df2 = x['3Bad']$`3Bad`
  rownames(df2) = df2$region
  df2 = df2['freq']
  colnames(df2) = '3Bad'
  map_data = cbind(df1,df2)
  map_data['state_abb'] = rownames(map_data)
  map_data$`1Good`[map_data$`1Good`<=10]=0
  map_data$`1Good`[map_data$`3Bad`<=10]=0
```

```
  map_data$`3Bad`[map_data$`1Good`<=10]=0
  map_data$`3Bad`[map_data$`3Bad`<=10]=0
  map_data['ratio']=round(map_data$`1Good`/(map_data$`1Good`+map_data$`3Bad`),digits = 2)
  if (nrow(map_data)>50) {
    map_data = map_data[-8,]
  }
  map_data$States=state.name[match(map_data$state_abb,state.abb)]
  map_data$states = tolower(map_data$States)
  data("fifty_states")
  p=ggplot(map_data, aes(map_id = map_data$states)) +
    geom_map(aes(fill = ratio), map = fifty_states) +
    scale_fill_gradient(limits=c(0.65,0.95),
                        low = 'dodgerblue4',
                        high = 'lightskyblue1',
                        guide = guide_legend(title = "Good ratio")) +
    borders("state",colour = "white") +
    borders("usa",colour = "dodgerblue4") +
    expand_limits(x = fifty_states$long, y = fifty_states$lat) +
    scale_x_continuous(breaks = NULL) +
    scale_y_continuous(breaks = NULL) +
    labs(x = "", y = "", title=year_number,
         legend.background = element_rect(colour = 'blue')) +
    theme(legend.position = "bottom",
          panel.background = element_blank()) +
    fifty_states_inset_boxes()+
    coord_fixed(1.3)
  p
  ggsave(paste(year_number,".jpg",sep=''))
  write.csv(map_data,paste(year_number,'.csv',sep=''))
  return(list(p,map_data))
}

listt = DrawMap(df_test,'test:2016.08-2017.12')
print(listt[[1]])
ggsave(paste('2016.08-2017.12',".jpg",sep=''))
map_data = listt[[2]]
map_data = map_data[c('state_abb','ratio')]
colnames(map_data)[2]= '2017'
df_ratio=map_data
ratio = map_data[,2]
ratio_2017 = c(min(ratio,na.rm=TRUE),max(ratio,na.rm=TRUE),mean(ratio,na.rm=TRUE))
df_ratio_summary = as.data.frame(ratio_2017)
l=list()
for (x in c('2013','2014','2015','2016')) {
  listt=DrawMap(df,x)
  print(listt[[1]])
  map_data = listt[[2]]
  map_data = map_data[c('state_abb','ratio')]
  colnames(map_data)[2]= x
  df_ratio = merge(x=df_ratio,y=map_data,by='state_abb')
  ratio = map_data[,2]
  ratio_summary = c(min(ratio,na.rm=TRUE),max(ratio,na.rm=TRUE),mean(ratio,na.rm=TRUE))
  ratio_summary = as.data.frame(ratio_summary)
```

```
  df_ratio_summary = cbind.data.frame(df_ratio_summary,ratio_summary)
}
rownames(df_ratio_summary)=c('min','max','mean')
colnames(df_ratio_summary)=c('2017','2013','2014','2015','2016')

df_ratio$`2017`=NULL
df_ratio1 = df_ratio
df_ratio = df_ratio %>% gather(key = "year", value = "ratio",-state_abb)
df_ratio = unite(df_ratio,state_year,state_abb,year)


library(WindR)
w.start()
multiMerge = function(l,by){
  df_merge = l[[1]]
  l[[1]]=NULL
  for (x in l) {
    df_merge = merge(df_merge,x,by=by)
  }
  return(df_merge)
}

l_states_names = c('AL','AK','AZ','AR','CA','CO','CT','DE','DC','FL',
                   'GA','HI','ID','IL','IN','IA','KS','KY','LA','ME','MD',
                   'MA','MI','MN','MS','MO','MT','NE','NV','NH','NJ',
                   'NM','NY','NC','ND','OH','OK','OR','PA','RI','SC',
                   'SD','TN','TX','UT','VT','VA','WA','WV','WI','WY')
# States unemployment rate
l_number = 'G1125905'
for (i in 1:50) {
  number = 1125905+i
  symbol = paste('G',number,sep='')
  l_number = paste(l_number,symbol,sep=',')
}
States_unemp<-w.edb(l_number,'2011-12-01','2018-05-31','Fill=Previous')
df_States_unemp = States_unemp$Data
colnames(df_States_unemp)[1]='date'
colnames(df_States_unemp)[2:52]=paste('States_unemp',l_states_names,sep='_')
# States annuall income
l_number = 'G1127078'
for (i in 1:50) {
  number = 1127078+i
  symbol = paste('G',number,sep='')
  l_number = paste(l_number,symbol,sep=',')
}
States_income<-w.edb(l_number,'2011-01-01','2018-05-31','Fill=Previous')
df_States_income = States_income$Data
colnames(df_States_income)[1]='date'
colnames(df_States_income)[2:52]=paste('States_income',l_states_names,sep='_')
df_States_income = separate(df_States_income,date,c('year','month','day'))
df_States_income[,c('month','day')]=NULL
colnames(df_States_income)[2:52]=l_states_names
colnames(df_States_unemp)[2:52]=l_states_names
```

```
Date = df_States_unemp$date
Date = substr(Date,1,7)
df_States_unemp$date = Date

df_States_income_new = gather(df_States_income,key='state',value='average_income',-year)
df_States_income_new = unite(df_States_income_new,state_year,state,year,sep='_')
df_States_unemp_new = gather(df_States_unemp,key='state',value='unemp',-date)
library(reshape)
df1 = separate(df_States_unemp_new,date,c('year','month'))
df1 = cast(df1,year~state, value = 'unemp',mean)
df2 = gather(df1,key='state',value='unemp',-year)
df_States_unemp_new = unite(df2,state_year,state,year,sep='_')

df1=merge(df_ratio,df_States_unemp_new,by='state_year')
df=merge(df1,df_States_income_new,by='state_year')

ggplot(data = df) +
  geom_point(mapping = aes(x = ratio, y = unemp))
ggplot(data = df) +
  geom_point(mapping = aes(x = ratio, y = average_income))

ggplot(data = df, mapping = aes(x = ratio)) +
  geom_freqpoly(binwidth = 0.25)
```

Data Exploration

```
rm(list = ls())
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
library(readr)
library(dplyr)
library(tidyr)
library(tidyverse)
library(lubridate)
library(ggplot2)

df_train = read.csv('0531_CleanData_train&validate.csv')[,-1]
text_data = df_train[c('ID','status','emp_title','desc','title')]
Df = df_train
Df[c('emp_title','desc','title')] = NULL
df = Df
df['date']=NULL
df['addr_state']=NULL
df$year = as.factor(df$year)
df$date = dmy(paste("01-", df$issue_d , sep =""))

cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00",
"#CC79A7")

number = count(df, year, wt = NULL, sort = FALSE) %>% as.data.frame()

df_full = read.csv('0531_CleanData_Full.csv')
```

```
##Visualizing Distributions
ggplot(data = df) +
  geom_bar(mapping = aes(x = status))
##Use a histogram to examine the distribution of a continuous variable.
ggplot(data = df) +
  geom_histogram(mapping = aes(x = USA_CPI), binwidth = 0.5)
##How to overlay multiple histograms in the same plot.
ggplot(data = df, mapping = aes(x = USA_int_rate,y = ..density.., color = status)) +
  geom_freqpoly(binwidth = 0.1)
##Box plot
ggplot(data = df, mapping = aes(x = year, y = USA_GDP)) +
  geom_violin(aes(fill=status))
# Visualize changes over time
ggplot(df, aes(date, states_unemp)) +
  geom_line(aes(group = status), color = "grey50") +
  geom_point(aes(color = status))

##To visualize the covariation between categorical variables
##we should COUNT the number of observations for each combination.
df %>% count(status, date)
##Then visualize with geom_tile() and the fill aesthetic:
df %>%
  count(status, date) %>%
  ggplot(mapping = aes(x = status, y = date)) +
  geom_tile(mapping = aes(fill = n))
##to visualize the covariation between two continuous variables
##draw a scatterplot with geom_point()
ggplot(data = df) +
  geom_point(mapping = aes(x = date, y = states_unemp, color=status),
             position="jitter")
ggplot(data = df, mapping = aes(x = date)) +
  geom_freqpoly(binwidth = 0.25)
df %>%
  count(date, status) %>%
  ggplot(aes(date, status, fill = n)) +
  geom_tile()

### Facets. Add additional categorical variables
### To split the plot into facets, subplots that each display one subset of the data.
ggplot(data = df) +
  geom_point(mapping = aes(x = int_rate, y = USA_int_rate)) +
  facet_wrap(~ status)
### To facet your plot on the combination of two variables, add facet_grid() to your plot call.
ggplot(data = df) +
  geom_point(mapping = aes(x = int_rate, y = USA_int_rate)) +
  facet_grid(year ~ status)
ggplot(data = df) +
  geom_point(mapping = aes(x = int_rate, y = USA_int_rate)) +
  facet_grid(. ~ status,nrow = 2)
```

```
df = read.csv('econ_usa.csv')[,-1]
require(gridExtra)
df$date = as.Date(paste(df$date,'01',sep='-'))
p1 = ggplot(df) + geom_point(aes(date, USA_unemp,color = date))+theme(legend.position="none")+
  labs(x='',y='')+
  # xlab('')+
  # ylab(expression(atop("unemploy", "rate")))+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
p2= ggplot(df) + geom_point(aes(x=date, y=USA_CPI, color = date))+theme(legend.position="none")+
  labs(x='',y='')+
  # xlab('')+
  # ylab(expression(atop("CPI", "Index")))+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
p3= ggplot(df) + geom_point(aes(x=date, y=df_USA_int_rate,color =
date))+theme(legend.position="none")+
  labs(x='',y='')+
  # xlab('')+
  # ylab(expression(atop("Interest", "Rate")))+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
p4= ggplot(df) + geom_point(aes(date, df_USA_stock,color = date))+theme(legend.position="none")+
  labs(x='',y='')+
  # xlab('')+
  # ylab(expression(atop("Stock", "Index")))+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
p5= ggplot(df) + geom_point(aes(date, USA_total_saving,color =
date))+theme(legend.position="none")+
  labs(x='',y='')+
  # xlab('')+
  # ylab(expression(atop("Total", "Saving")))+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
p6= ggplot(df) + geom_point(aes(date, USA_net_saving,color =
date))+theme(legend.position="none")+
  labs(x='',y='')+
  # xlab('')+
  # ylab(expression(atop("Net", "Saving")))+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
p7= ggplot(df) + geom_point(aes(date, USA_GDP,color = date))+theme(legend.position="none")+
  labs(x='',y='')+
  # xlab('')+
  # ylab(expression(atop("Total", "GDP")))+
  theme(axis.title.x=element_blank(),
```

```
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
p8= ggplot(df) + geom_point(aes(date, USA_ind_GDP,color = date))+theme(legend.position="none")+
  labs(x='',y='')+
  # ylab(expression(atop("Individual", "GDP")))+
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
p9= ggplot(df) + geom_point(aes(date, USA_ind_consume,color =
date))+theme(legend.position="none")+
  labs(x='',y='')+
  scale_y_discrete(labels = c(''))
  # xlab('')+
  # ylab(expression(atop("Individual", "Consume")))
grid.arrange(p1,p2,p3,p4,p5,p6,p7,p8,p9, ncol=1)

df1 = read.csv('econ_States_income.csv')
States1 = filter(df1,year==2012)
States1 = States1[order(States1$average_income,decreasing = T),]
#df1$state <- factor(df1$state, levels = df1$state[order(df1$average_income)])
p10=ggplot(df1) +
  geom_point(aes(average_income, state ,color = year))+
  #theme(legend.position="none")+
  scale_y_discrete(limits=States1$state)+
  labs(x='Average Annual Income',y='',colour='date')

df2 = read.csv('econ_States_unemp.csv')
Date = df2$date
date_num = c()
for (x in Date) {
  y = strsplit(x,split = '-')[[1]]
  num = as.integer(y[1]) + as.integer(y[2])/100.0
  date_num=c(date_num,num)
}
df2$date_num = date_num
States2 = filter(df2,date=='2018-04')
States2 = States2[order(States2$unemp),]
p11=ggplot(df2, aes(x = unemp, y = state,color=date_num)) +
  geom_point(stat = "identity")+
  theme(legend.position="none")+
  scale_y_discrete(limits=States2$state)+
  labs(x='Unemployment Rate',y='State Abbreviations')

grid.arrange(p11,p10, ncol=2)
library(scales)

DD1 = df
df = DD1
df$term = as.integer(substr(df$term,1,2))
df = df[c('status','int_rate','tot_coll_amt','total_rev_hi_lim',
          'bc_open_to_buy','fico_range_low','term','loan_amnt',
          'balance_to_limit','num_account')]
df_good = filter(df,status=='1Good')
```

```
df_bad = filter(df,status=='3Bad')
dff_good = apply(df_good[,-1], 2, mean)
dff_bad = apply(df_bad[,-1], 2, mean)
dff_min = apply(df[,-1], 2, min)
dff_max = apply(df[,-1], 2, max)
dff = data.frame(good=dff_good,bad=dff_bad)
dff$maxx = apply(dff,1,max)
dff$good = dff$good/dff$maxx*100
dff$bad = dff$bad/dff$maxx*100
dff$maxx=NULL
dff$variable = rownames(dff)
library(tidyr)
dff=gather(dff, key = "status", value = "value",-variable)
quantile(df$int_rate, c(.98))
coord_radar <- function (theta = "x", start = 0, direction = 1)
{
  theta <- match.arg(theta, c("x", "y"))
  r <- if (theta == "x")
    "y"
  else "x"
  ggproto("CordRadar", CoordPolar, theta = theta, r = r, start = start,
          direction = sign(direction),
          is_linear = function(coord) TRUE)
}
ggplot(dff, aes(x = variable, y = value)) +
  geom_polygon(aes(group = status, color = status), fill = NA, size = 1, show.legend = FALSE) +
  geom_line(aes(group = status, color = status), size = 1) +
  theme(strip.text.x = element_text(size = rel(0.8)),
        axis.text.x = element_text(size = rel(0.8)),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank()) +
  xlab("") + ylab("") +
  guides(color = guide_legend(ncol=2)) +
  coord_radar()


 ### Payment Ratio
load("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan
data/originalData.RData")

df = df_original
df = filter(df,loan_status %in% c('Charged Off','Fully Paid'))
df = mutate(df,status=ifelse(loan_status=='Fully Paid',"Good","Bad"))
df$date = df$issue_d
library(lubridate)
Date = dmy(paste("01-", df$date , sep =""))
Date = as.Date(Date)
Date = dmy(paste("01-", df$date , sep =""))
df$date = Date
df = df[c('status','date','total_pymnt','loan_amnt')]
df = na.omit(df)
df$pay_ratio = df$total_pymnt/df$loan_amnt
df_before = df[df$date<as.Date('2016-08-01'),]
```

```
df_after = df[df$date>=as.Date('2016-08-01'),]

ratio_before = tapply(df_before$pay_ratio,df_before$status, mean)
ratio_after = tapply(df_after$pay_ratio,df_after$status, mean)
```

Lasso with Sampling (rewrote glmnet)

```
rm(list = ls())
gc()
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
load("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan
data/0_final_Lasso_balanced.rds.RData")
library(readr)
library(dplyr)
library(tidyr)

library(ROSE)
library(caret)
library(pROC)
library (glmnet)
library(caret)
library(mlbench)
library(rpart)

### Data and seed ==============================================================================
set.seed(42)
df_clean=readRDS(file = "cleanData0608.rds")


### Functions ==================================================================================
getmin1=function(lambda,cvm,cvsd){
  cvmin=min(cvm,na.rm=TRUE)
  idmin=cvm<=cvmin
  lambda.min=max(lambda[idmin],na.rm=TRUE)
  idmin=match(lambda.min,lambda)
  semin=(cvm+cvsd)[idmin]
  idmin=cvm<=semin
  lambda.1se=max(lambda[idmin],na.rm=TRUE)
  list(lambda.min=lambda.min,lambda.1se=lambda.1se)
}

cv.glmnet1 <-
  function (outlist, lambda, df, weights, offset, foldid, type.measure,
            grouped, keep = FALSE)
  {
    ##We dont want to extrapolate lambdas on the small side
    mlami=max(sapply(outlist,function(obj)min(obj$lambda)))
    which_lam=lambda >= mlami
    predmat = matrix(NA, nrow(df), length(lambda))
    nfolds = max(foldid)
    nlams = double(nfolds)
```

```
    for (i in seq(nfolds)) {
      which = foldid == i
      df_valid = df[which,]
      x_valid=model.matrix(status~.,df_valid)[,-1]
      y_valid=as.numeric(df_valid$status)
      fitobj = outlist[[i]]
      fitobj$offset = FALSE
      preds = predict(fitobj, x_valid, s=lambda[which_lam])
      nlami = sum(which_lam)
      predmat[which, seq(nlami)] = preds
      nlams[i] = nlami
    }
    N = nrow(df) - apply(is.na(predmat), 2, sum)
    cvraw = (y_valid - predmat)^2
    cvm = apply(cvraw, 2, mean)
    cvsd = sqrt(apply(scale(cvraw, cvm, FALSE)^2, 2, mean)/(N - 1))
    out = list(cvm = cvm, cvsd = cvsd, type.measure=type.measure)
    if (keep)
      out$fit.preval = predmat
    out
  }

cv<-function (df, offset = NULL, lambda = NULL,
            nfolds = 10, grouped = TRUE, keep = FALSE, parallel = FALSE){
  N = nrow(df)
  rows = N%/%10
  drop_number = sample.int(N,N-rows*10)
  df = df[-drop_number,]
  type.measure = "default"
  N = nrow(df)
  weights = rep(1, N)
  glmnet.call = match.call(expand.dots = TRUE)
  which = match(c("type.measure", "nfolds", "foldid", "grouped",
                  "keep"), names(glmnet.call), F)
  if (any(which))
    glmnet.call = glmnet.call[-which]
  glmnet.call[[1]] = as.name("glmnet")
  x = model.matrix(status~.,df)[,-1]
  y = as.numeric(df$status)
  glmnet.object = glmnet(x, y, weights = weights, offset = offset,
                         lambda = lambda)
  glmnet.object$call = glmnet.call
  subclass=class(glmnet.object)[[1]]
  type.measure=cvtype(type.measure,subclass)
  is.offset = glmnet.object$offset
  ###Next line is commented out so each call generates its own lambda sequence
  # lambda=glmnet.object$lambda
  nz = sapply(predict(glmnet.object, type = "nonzero"),
              length)
  foldid = sample(rep(seq(nfolds), length = N))
  outlist = as.list(seq(nfolds))
  for (i in seq(nfolds)) {
    which = foldid == i
```

```
    df_train = df[-which,]
    df_train_bal = ovun.sample(status ~ ., data = df_train, method = "both",
N=nrow(df_train))$data
    x_train = model.matrix(status~.,df_train_bal)[,-1]
    y_train = as.numeric(df_train_bal$status)
    offset_sub = NULL
    outlist[[i]] = glmnet(x_train,y_train, lambda = lambda, offset = offset_sub,
                          weights = rep(1:nrow(x_train)))
  }
  lambda = glmnet.object$lambda
  cvstuff = cv.glmnet1(outlist, lambda, df, weights,
                       offset, foldid, type.measure, grouped, keep)
  cvm = cvstuff$cvm
  cvsd = cvstuff$cvsd
  cvname = names(cvstuff$type.measure)
  names(cvname)=cvstuff$type.measure# to be compatible with earlier version; silly, I know
  out = list(lambda = lambda, cvm = cvm, cvsd = cvsd, cvup = cvm +
               cvsd, cvlo = cvm - cvsd, nzero = nz, name = cvname, glmnet.fit = glmnet.object)
  lamin=getmin(lambda, cvm, cvsd)
  obj = c(out, as.list(lamin))
  class(obj) = "cv.glmnet"
  obj
}

### Data spliting ================================================================
Df = df_clean
Df$ID=NULL
Df$year = as.factor(Df$year)
Df$month = as.factor(Df$month)
Df$status = as.factor(Df$status)
Df$status = factor(Df$status,c('Good','Bad'))
Date = as.Date(paste(Df$date,"-01", sep =""))
Df$date = Date
df_test = Df[Df$date>=as.Date('2016-08-01'),]
df_train_valid = Df[Df$date<as.Date('2016-08-01'),]
df_test[c('X','ID','issue_d','addr_state','fico_range_high','date',
          'emp_title','desc','title')]=NULL
df_train_valid[c('X','ID','issue_d','addr_state','fico_range_high','date',
                 'emp_title','desc','title','payment')]=NULL
df_test = na.omit(df_test)
df_train_valid = na.omit(df_train_valid)

### Model training ===============================================================
## Split dataset into training and validating
df = df_train_valid
Status = ifelse(df$status=='Good',0,1) #Bad=1
df$status = as.factor(Status)
nobs = nrow(df)
train = sample = sample(nobs,nobs*0.8)
validate = sample(setdiff(seq_len(nobs),train),0.2*nobs)
df_train = df[train,]
df_train_bal = ovun.sample(status ~ ., data = df_train, method = "both", N=nrow(df_train))$data
x_train_bal = model.matrix(status~.,df_train_bal)[,-1]
```

```
y_train_bal = as.numeric(df_train_bal$status)
df_valid = df[validate,]
x_valid = model.matrix(status~.,df_valid)[,-1]
y_valid = as.numeric(df_valid$status)

## Run lasso regression
grid = 10^seq(3,-6,length=100) #Set lambdas used in lasso
lasso.mod_grid =glmnet(x_train_bal, y_train_bal,alpha=1, lambda=grid,thresh=1e-12)
plot(lasso.mod_grid)
# plot(lasso.mod_grid, xvar = "dev")
vnat=coef(lasso.mod_grid)
vnat=vnat[-1,ncol(vnat)] # remove the intercept, and get the coefficients at the end of the path
vnat1=vnat
vnat1[abs(vnat1)<0.05] = 0
vnat1[vnat1!=0]
vnat2=sort(vnat1[vnat1!=0],decreasing = T)
data.frame(vnat2)
vn=paste("var",1:(dim(x_train_bal)[2]))
plot(lasso.mod_grid, xvar = "lambda")
axis(4, at=vnat,line=-5.5,label=vn,las=1,tick=FALSE,
     cex.axis=2,position_jitter(width = 0,height = 0))
plot(lasso.mod_grid, xvar = "lambda")
plot(lasso.mod_grid, xvar = "dev")

## Use cross-validation in lasso (10 folds)
cv.out =cv(df_train,nfolds=10,lambda=grid)
plot(cv.out)
## Find the best lambda that generates the minimum
bestlam =cv.out$lambda.min
bestlam
bestlam=0.02
## Use model with best lambda
lasso.mod_best = glmnet(x_train_bal, y_train_bal, alpha=1, lambda=bestlam,thresh=1e-12)
## Select important variables
predict.df = predict(lasso.mod_best,type ="coefficients",s=bestlam)
lasso.coef=predict.df[1:nrow(predict.df),]
variable.select.coef = lasso.coef[lasso.coef !=0]
variable.select.coef
variable.select = attributes(variable.select.coef)$names[-1]
variable.select
coef(lasso.mod_best, s = "lambda.min")


##### Performance================================================
### Validate Data
lasso.pred=predict(lasso.mod_best,s=bestlam,newx=x_valid)
## Calculate performances of valiadation data
error = lasso.pred - y_valid
MSE = mean(error^2)
MAPE = 100*mean(abs(error/y_valid))
RMSE = MSE^0.5
perf.df = data.frame(MSE,MAPE,RMSE)
perf.df
```

```
#       MSE     MAPE      RMSE
# 0.2142599 39.25195 0.4628822
## ROC
lasso.pred=predict(lasso.mod_best,s=bestlam,newx=x_valid)
r = roc(as.numeric(y_valid),as.numeric(lasso.pred))
plot.roc(r)
r
# Area under the curve: 0.7249

lasso.pred1 = ifelse(lasso.pred<1.5,1,2)
CM = confusionMatrix(as.factor(lasso.pred1),as.factor(y_valid))
CM
Sen = c(CM$byClass['Sensitivity'])
Spe = c(CM$byClass['Specificity'])
Acc = c(CM$overall['Accuracy'])
return = (-0.128+0.128*as.numeric(Sen)+0.096*as.numeric(Spe))
return
# 0.0208796

### Test data ========================================================================
df = df_test
payment = df$payment
df_test_use = df
df_test_use$payment = NULL
x_test = model.matrix(status~.,df_test_use)[,-1]
y_test = as.numeric(df_test_use$status)
lasso.pred=predict(lasso.mod_best,s=bestlam,newx=x_test)
lasso.pred1 = ifelse(lasso.pred<1.5,1,2)
confusionMatrix(as.factor(lasso.pred1),as.factor(y_test))
r = roc(as.numeric(y_test),as.numeric(lasso.pred))
plot.roc(r)
r
#Area under the curve: 0.687
df = data.frame(pred=lasso.pred1,actual=y_test,amount=df_test$loan_amnt,pay=payment)
colnames(df)[1]='pred'
df = mutate(df,pay_ratio=pay/amount)
df_pred_good = filter(df,pred=='1')
pred_total_amount = sum(df_pred_good$amount)
pred_total_pay = sum(df_pred_good$pay)
pre_total_return = pred_total_pay-pred_total_amount
actual_total_amount = sum(df$amount)
actual_total_pay = sum(df$pay)
actual_total_return = actual_total_pay-actual_total_amount
return = pre_total_return-actual_total_return
return_ratio = (sum(df_pred_good$pay)/sum(df_pred_good$amount))-(sum(df$pay)/sum(df$amount))
return
return_ratio
# return = 68148461
# return_ratio = 0.05707064


library(ggplot2)
df$actual = as.factor(df$actual)
```

```
df$pred = as.factor(df$pred)
ggplot(df,aes(x=actual,y=pay_ratio,fill=pred))+
  geom_violin()
ggplot(df,aes(x=amount,y=pay,color=pred,alpha=0.2))+
  geom_point()
```

Lasso with Cutoff Selecting

```
rm(list = ls())
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
load("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan
data/0_final_Lasso_unbalanced.rds.RData")

library(readr)
library(dplyr)
library(tidyr)
library(ROSE)
library(caret)
library(pROC)
library (glmnet)
library(caret)
library(mlbench)
library(rpart)

df_clean=readRDS(file = "cleanData0608.rds")
Df = df_clean
Df$ID=NULL
Df$year = as.factor(Df$year)
Df$month = as.factor(Df$month)
Df$status = as.factor(Df$status)
Df$status = factor(Df$status,c('Good','Bad'))
Date = as.Date(paste(Df$date,"-01", sep =""))
Df$date = Date
df_test = Df[Df$date>=as.Date('2016-08-01'),]
df_train_valid = Df[Df$date<as.Date('2016-08-01'),]
df_test[c('X','ID','issue_d','addr_state','fico_range_high','date',
          'emp_title','desc','title')]=NULL
df_train_valid[c('X','ID','issue_d','addr_state','fico_range_high','date',
                 'emp_title','desc','title','payment')]=NULL
df_test = na.omit(df_test)
df_train_valid = na.omit(df_train_valid)

df = df_train_valid
Status = ifelse(df$status=='Good',0,1) #Bad=1
df$status = as.factor(Status)
set.seed(42)
nobs = nrow(df)
train = sample = sample(nobs,nobs*0.8)
validate = sample(setdiff(seq_len(nobs),train),0.2*nobs)
df_train = df[train,]
df_valid = df[validate,]
```

```r
# Split the dataset to independent variables X and dependent variable y
x_train=model.matrix(status~.,df_train)[,-1]
y_train=as.numeric(df_train$status)
x_valid = model.matrix(status~.,df_valid)[,-1]
y_valid = as.numeric(df_valid$status)
# Run lasso regression
grid = 10^seq(0,-5,length=100) #Set lambdas used in lasso
# Use training data in lasso
lasso.mod_grid =glmnet(x_train, y_train,
                  alpha=1, lambda=grid,thresh=1e-12)
plot(lasso.mod_grid)
# plot(lasso.mod_grid, xvar = "dev")
vnat=coef(lasso.mod_grid)
vnat=vnat[-1,ncol(vnat)] # remove the intercept, and get the coefficients at the end of the path
vnat1=vnat
vnat1[abs(vnat1)<0.05] = 0
vnat1[vnat1!=0]
vnat2=sort(vnat1[vnat1!=0],decreasing = T)
data.frame(vnat2)
vn=paste("var",1:(dim(x_train)[2]))
plot(lasso.mod_grid, xvar = "lambda")
axis(4, at=vnat,line=-5.5,label=vn,las=1,tick=FALSE,
     cex.axis=2,position_jitter(width = 0,height = 0))
plot(lasso.mod_grid, xvar = "dev")

# Use cross-validation in lasso (10 folds)
cv.out =cv.glmnet(x_train,y_train,nfolds=10,lambda=grid)
plot(cv.out)
# Find the best lambda that generates the minimum
bestlam =cv.out$lambda.min
bestlam
df1 = data.frame(lambda=cv.out$lambda,cvm=cv.out$cvm)
bestlam=0.01
## Use model with best lambda
lasso.mod_best = glmnet(x_train, y_train, alpha=1, lambda=bestlam,thresh=1e-12)
## Select important variables
predict.df = predict(lasso.mod_best,type ="coefficients",s=bestlam)
lasso.coef=predict.df[1:nrow(predict.df),]
variable.select.coef = lasso.coef[lasso.coef !=0]
as.data.frame(variable.select.coef)
variable.select = attributes(variable.select.coef)$names[-1]
variable.select
coef(lasso.mod_best, s = "lambda.min")

##### Performance================================================
### Validate Data
lasso.pred=predict(lasso.mod_best,s=bestlam,newx=x_valid)
## Calculate performances of valiadation data
error = lasso.pred - y_valid
MSE = mean(error^2)
MAPE = 100*mean(abs(error/y_valid))
RMSE = MSE^0.5
```

```r
perf.df = data.frame(MSE,MAPE,RMSE)
perf.df
#       MSE     MAPE      RMSE
# 0.1505034 22.97915 0.3879476
## ROC
lasso.pred=predict(lasso.mod_best,s=bestlam,newx=x_valid)
r = roc(as.numeric(y_valid),as.numeric(lasso.pred))
plot.roc(r)
r
# Area under the curve: 0.726
lasso.pred1 = ifelse(lasso.pred<1,1,2)
CM = confusionMatrix(as.factor(lasso.pred1),as.factor(y_valid))
l_Sensitivity = c(CM$byClass['Sensitivity'])
l_Specificity = c(CM$byClass['Specificity'])
l_Accuracy = c(CM$overall['Accuracy'])
l_cutoff = seq(1.1,2,by=0.1)
for (cutoff in l_cutoff) {
  lasso.pred1 = ifelse(lasso.pred<cutoff,1,2)
  CM = confusionMatrix(as.factor(lasso.pred1),as.factor(y_valid))
  l_Sensitivity = c(l_Sensitivity,CM$byClass['Sensitivity'])
  l_Specificity = c(l_Specificity,CM$byClass['Specificity'])
  l_Accuracy = c(l_Accuracy,CM$overall['Accuracy'])
}
l_cutoff = c(1,l_cutoff)
df_cutoff = data.frame(cutoff=l_cutoff,sen=l_Sensitivity,spe=l_Specificity,acu=l_Accuracy)
df_cutoff$return = (-0.128+0.128*df_cutoff$sen+0.096*df_cutoff$spe)
return_max = max(df_cutoff$return)
cutoff_best = filter(df_cutoff,return==return_max)
cutoff_best
# cutoff       sen       spe       acu     return
#    1.3 0.8403699 0.4302304 0.7524688 0.02086947
cutoff = as.numeric(cutoff_best[1])
lasso.pred1 = ifelse(lasso.pred<cutoff,1,2)
CM = confusionMatrix(as.factor(lasso.pred1),as.factor(y_valid))
CM
df1 = df_cutoff
library(plotly)
df = df_cutoff[,c(2,3,5)]
plot_ly(
  type='scatter3d',
  x=df1$sen,
  y=df1$spe,
  z=df1$return %>%
    layout(
      xaxis=
    )
)
plot_ly(df1, x = ~sen, y = ~spe, z = ~return,
        marker = list(color = ~return, colorscale = c('#FFE1A1', '#683531'), showscale = TRUE))
%>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Sensitivity'),
                      yaxis = list(title = 'Specificity'),
```

```
                      zaxis = list(title = 'Extra Return')),
         annotations = list(
           x = 1.13,
           y = 1.05,
           text = 'Return Ratio',
           xref = 'paper',
           yref = 'paper',
           showarrow = FALSE
         ))

### Test data
Df_test = df_test
payment = Df_test$payment
df_test = Df_test
df_test$payment = NULL
x_test = model.matrix(status~.,df_test)[,-1]
y_test = as.numeric(df_test$status)
lasso.pred=predict(lasso.mod_grid,s=bestlam,newx=x_test)
lasso.pred1 = ifelse(lasso.pred<cutoff,1,2)
confusionMatrix(as.factor(lasso.pred1),as.factor(y_test))
r = roc(as.numeric(y_test),as.numeric(lasso.pred))
plot.roc(r)
r
# Area under the curve: 0.6677
df = data.frame(pred=lasso.pred1,actual=y_test,amount=df_test$loan_amnt,pay=payment)
colnames(df)[1]='pred'
df = mutate(df,pay_ratio=pay/amount)
df_pred_good = filter(df,pred=='1')
pred_total_amount = sum(df_pred_good$amount)
pred_total_pay = sum(df_pred_good$pay)
pre_total_return = pred_total_pay-pred_total_amount
actual_total_amount = sum(df$amount)
actual_total_pay = sum(df$pay)
actual_total_return = actual_total_pay-actual_total_amount
return = pre_total_return-actual_total_return
return_ratio = (sum(df_pred_good$pay)/sum(df_pred_good$amount))-(sum(df$pay)/sum(df$amount))
return
return_ratio
# return = 63396436
# return_ratio = 0.04895407

library(ggplot2)
df$actual = as.factor(df$actual)
df$pred = as.factor(df$pred)
ggplot(df,aes(x=actual,y=pay_ratio,fill=pred))+
  geom_violin()
ggplot(df,aes(x=amount,y=pay,color=pred,alpha=0.2))+
  geom_point()
```

Random Forest with Sampling

```
rm(list = ls())
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
load("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan
data/0_final_RF_balanced_100.rds.RData")
library(readr)
library(dplyr)
library(tidyr)
library(ROSE)
library(forecast)
library(MASS)
library(pROC)
library(ROCR)

df_clean=readRDS(file = "cleanData0608.rds")
Df = df_clean
Df$ID=NULL
Df$year = as.factor(Df$year)
Df$month = as.factor(Df$month)
Df$status = as.factor(Df$status)
Df$status = factor(Df$status,c('Good','Bad'))
Date = as.Date(paste(Df$date,"-01", sep =""))
Df$date = Date

df_test = Df[Df$date>=as.Date('2016-08-01'),]
df_train_valid = Df[Df$date<as.Date('2016-08-01'),]
df_test[c('X','ID','issue_d','addr_state','fico_range_high','date',
          'emp_title','desc','title')]=NULL
df_train_valid[c('X','ID','issue_d','addr_state','fico_range_high','date',
                 'emp_title','desc','title','payment')]=NULL
df_test = na.omit(df_test)
df_train_valid = na.omit(df_train_valid)

#=============================================================================
# Balanced data
df = df_train_valid
set.seed(42)
nobs = nrow(df)
train = sample = sample(nobs,nobs*0.8)
validate = sample(setdiff(seq_len(nobs),train),0.2*nobs)
df_train = df[train,]
df_train_balanced = ovun.sample(status ~ ., data = df_train, method = "both",
N=nrow(df_train))$data
require(randomForest)
rf_b = randomForest(status ~ .,
                    data = df_train_balanced,
                    ntree=100, mtry=9,
                    importance=TRUE,replace=FALSE)
saveRDS(rf_b, file = "RF_balanced.rds")
rf_b=readRDS(file = "RF_balanced.rds")
min.err = min(data.frame(rf_b$err.rate)['OOB'])
min.err.index = which(data.frame(rf_b$err.rate)['OOB']==min.err)
min.err
min.err.index
```

```
# Generate textual output of the 'Random Forest' model.
rf_b
# Calculate the Area Under the Curve (AUC).
pROC::roc(rf_b$y, as.numeric(rf_b$predicted))
# Calculate the AUC Confidence Interval.
pROC::ci.auc(rf_b$y, as.numeric(rf_b$predicted),FALSE)
# List the importance of the variables.
rn <- round(rf_b$importance, 2)
df_importance = as.data.frame(rn[order(rn[,3], decreasing=TRUE),])
df_importance
p <- rattle::ggVarImp(rf_b,title="Variable Importance Random Forest fund_df_explore.csv")
p
# Plot the error rate against the number of trees.
plot(rf_b, main="")
legend("topright", c("OOB", "Good", "Bad"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest - Balanced Dataset")

### Validate Data ======================================================
df_valid = df_train_valid[validate,]
df_valid_inputs = df_valid
df_valid_inputs$status = NULL
df_valid_output = df_valid$status
df_valid_output = factor(df_valid_output,c('Good','Bad'))
rf.pred_prob = predict(rf_b,newdata = df_valid_inputs,type = 'prob')[,2]
pred <- prediction(rf.pred_prob,df_valid_output)
# Convert rate of positive predictions to percentage.
per <- performance(pred, "lift", "rpp")
per@x.values[[1]] <- per@x.values[[1]]*100
# Plot the lift chart.
ROCR::plot(per, col="#CC0000FF", lty=1, xlab="Caseload (%)", add=FALSE)
legend("topright", c("Random Forest"), col=rainbow(1, 1, .8), lty=1:1, title="Models",
inset=c(0.05, 0.7))
title(main="Lift Chart - Balanced Dataset [validate]")
grid()
r = roc(as.numeric(df_valid_output),as.numeric(rf.pred_prob)+1)
plot.roc(r)
r
# Area under the curve: 0.6442
df_valid_output_num = as.numeric(df_valid_output)
error = (rf.pred_prob+1) - df_valid_output_num
MSE = mean(error^2)
MAPE = 100*mean(abs(error/df_valid_output_num))
RMSE = MSE^0.5
perf.df = data.frame(MSE,MAPE,RMSE)
perf.df
#       MSE     MAPE      RMSE
# 0.1751898 32.09412 0.4185568
rf.pred1 = ifelse(rf.pred_prob<0.5,'Good','Bad')
rf.pred1 = as.factor(rf.pred1)
rf.pred1 = factor(rf.pred1,levels = c('Good','Bad'))
CM = confusionMatrix(rf.pred1,df_valid_output)
Sen = c(CM$byClass['Sensitivity'])  #0.8184707
Spe = c(CM$byClass['Specificity'])  #0.4763452
```

```
Acc = c(CM$overall['Accuracy'])     #0.7451464
return = (-0.128+0.128*as.numeric(Sen)+0.096*as.numeric(Spe))
# 0.02249339


### ============================================================================
### Rerun model with cleaned dataset==========================================
l_drop = c('purpose','inq_last_6mths','open_acc','total_acc','mo_sin_rcnt_rev_tl_op',
           'mo_sin_rcnt_tl','num_actv_bc_tl','num_actv_rev_tl','num_bc_sats','num_bc_tl',
           'num_il_tl','num_op_rev_tl','num_rev_accts','num_sats','num_tl_op_past_12m',
           'pct_tl_nvr_dlq','percent_bc_gt_75','mths_account','num_account','num_account_12m',
           'open_acc_p','log_mths_account','num_account_p','log_num_rev_trades',
           'log_num_account_12m','USA_int_rate','states_income','delinq_2yrs','pub_rec',
           'initial_list_status','collections_12_mths_ex_med','acc_now_delinq',
           'tot_coll_amt','chargeoff_within_12_mths','delinq_amnt','num_accts_ever_120_pd',
           'num_tl_120dpd_2m','num_tl_30dpd','num_tl_90g_dpd_24m','pub_rec_bankruptcies',
           'tax_liens')
df_train_valid_new = df_train_valid
df_train_valid_new[l_drop]=NULL
df = df_train_valid_new
set.seed(42)
nobs = nrow(df)
train = sample = sample(nobs,nobs*0.8)
validate = sample(setdiff(seq_len(nobs),train),0.2*nobs)
df_train = df[train,]
df_train_balanced = ovun.sample(status ~ ., data = df_train, method = "both",
N=nrow(df_train))$data
require(randomForest)
rf_b_new = randomForest(status ~ .,
                        data = df_train_balanced,
                        ntree=100, mtry=9,
                        importance=TRUE,replace=FALSE)
saveRDS(rf_b_new, file = "RF_balanced_new.rds")
min.err = min(data.frame(rf_b_new$err.rate)['OOB'])
min.err.index = which(data.frame(rf_b_new$err.rate)['OOB']==min.err)
min.err
min.err.index
# Generate textual output of the 'Random Forest' model.
rf_b_new
# Calculate the Area Under the Curve (AUC).
pROC::roc(rf_b_new$y, as.numeric(rf_b_new$predicted))
# Calculate the AUC Confidence Interval.
pROC::ci.auc(rf_b_new$y, as.numeric(rf_b_new$predicted),FALSE)
# List the importance of the variables.
rn <- round(rf_b_new$importance, 2)
df_importance = as.data.frame(rn[order(rn[,3], decreasing=TRUE),])
df_importance
p <- rattle::ggVarImp(rf_b_new,title="Variable Importance Random Forest fund_df_explore.csv")
p
# Plot the error rate against the number of trees.
plot(rf_b_new, main="")
legend("topright", c("OOB", "Good", "Bad"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest - Balanced Dataset")
```

```
### Validate Data ====================================================
df_valid = df_train_valid[validate,]
df_valid_inputs = df_valid
df_valid_inputs$status = NULL
df_valid_output = df_valid$status
df_valid_output = factor(df_valid_output,c('Good','Bad'))
rf.pred_prob = predict(rf_b_new,newdata = df_valid_inputs,type = 'prob')[,2]
pred <- prediction(rf.pred_prob,df_valid_output)
# Convert rate of positive predictions to percentage.
per <- performance(pred, "lift", "rpp")
per@x.values[[1]] <- per@x.values[[1]]*100
# Plot the lift chart.
ROCR::plot(per, col="#CC0000FF", lty=1, xlab="Caseload (%)", add=FALSE)
legend("topright", c("Random Forest"), col=rainbow(1, 1, .8), lty=1:1, title="Models",
inset=c(0.05, 0.7))
title(main="Lift Chart - Balanced Dataset [validate]")
grid()
r = roc(as.numeric(df_valid_output),as.numeric(rf.pred_prob)+1)
plot.roc(r)
r
# Area under the curve: 0.7275
df_valid_output_num = as.numeric(df_valid_output)
error = (rf.pred_prob+1) - df_valid_output_num
MSE = mean(error^2)
MAPE = 100*mean(abs(error/df_valid_output_num))
RMSE = MSE^0.5
perf.df = data.frame(MSE,MAPE,RMSE)
perf.df
#       MSE     MAPE      RMSE
# 0.1759966 32.07576 0.4195194
rf.pred1 = ifelse(rf.pred_prob<0.5,'Good','Bad')
rf.pred1 = as.factor(rf.pred1)
rf.pred1 = factor(rf.pred1,levels = c('Good','Bad'))
CM = confusionMatrix(rf.pred1,df_valid_output)
Sen = c(CM$byClass['Sensitivity'])  #0.8133779
Spe = c(CM$byClass['Specificity'])  #0.4820208
Acc = c(CM$overall['Accuracy'])     #0.7423614
return = (-0.128+0.128*as.numeric(Sen)+0.096*as.numeric(Spe))
# 0.02238637



### Test data=============================================
Df_test = Df[Df$date>=as.Date('2016-08-01'),]
payment = Df_test$payment
df_test = Df_test
df_test$payment = NULL
df_test_inputs = df_test
df_test_inputs$status = NULL
df_test_output = df_test$status
rf.pred = predict(rf_b_new,newdata = df_test_inputs)
rf.pred_prob = predict(rf_b_new,newdata = df_test_inputs,type = 'prob')[,2]
```

```
rf.pred1 = ifelse(rf.pred_prob<0.5,'Good','Bad')
rf.pred1 = as.factor(rf.pred1)
rf.pred1 = factor(rf.pred1,levels = c('Good','Bad'))
CM = confusionMatrix(rf.pred1,df_test_output)
CM
r = roc(as.numeric(df_test_output),as.numeric(rf.pred_prob))
plot.roc(r)
r
#Area under the curve:
df = data.frame(pred=rf.pred1,actual=df_test_output,amount=df_test$loan_amnt,pay=payment)
df = mutate(df,pay_ratio=pay/amount)
df_pred_good = filter(df,pred=='Good')
pred_total_amount = sum(df_pred_good$amount)
pred_total_pay = sum(df_pred_good$pay)
pre_total_return = pred_total_pay-pred_total_amount
actual_total_amount = sum(df$amount)
actual_total_pay = sum(df$pay)
actual_total_return = actual_total_pay-actual_total_amount
return = pre_total_return-actual_total_return
return_ratio = (sum(df_pred_good$pay)/sum(df_pred_good$amount))-(sum(df$pay)/sum(df$amount))
return #50823725
return_ratio #0.03951466

library(ggplot2)
df$actual = as.factor(df$actual)
df$pred = as.factor(df$pred)
ggplot(df,aes(x=actual,y=pay_ratio,fill=pred))+
  geom_violin()
ggplot(df,aes(x=amount,y=pay,color=pred,alpha=0.2))+
  geom_point()
```

Random Forest with Cutoff Selecting

```
rm(list = ls())
setwd("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan data")
load("C:/Users/JessicaGAO/Desktop/BDA/Final_Project/DataSet/All Lending Club loan
data/0_final_RF_unbalanced_100.rds.RData")

library(readr)
library(dplyr)
library(tidyr)
library(ROSE)
library(forecast)
library(MASS)
library(pROC)
library(ROCR)
library(ggplot2, quietly=TRUE)

df_clean=readRDS(file = "cleanData0608.rds")
Df = df_clean
Df$ID=NULL
```

```
Df$year = as.factor(Df$year)
Df$month = as.factor(Df$month)
Df$status = as.factor(Df$status)
Df$status = factor(Df$status,c('Good','Bad'))
Date = as.Date(paste(Df$date,"-01", sep =""))
Df$date = Date
df_test = Df[Df$date>=as.Date('2016-08-01'),]
df_train_valid = Df[Df$date<as.Date('2016-08-01'),]
df_test[c('X','ID','issue_d','addr_state','fico_range_high','date',
          'emp_title','desc','title')]=NULL
df_train_valid[c('X','ID','issue_d','addr_state','fico_range_high','date',
                 'emp_title','desc','title','payment')]=NULL
df_test = na.omit(df_test)
df_train_valid = na.omit(df_train_valid)

#=============================================================================
# Unbalanced data
df = df_train_valid
set.seed(42)
nobs = nrow(df)
train = sample = sample(nobs,nobs*0.8)
validate = sample(setdiff(seq_len(nobs),train),0.2*nobs)
df_train = df[train,]
require(randomForest)
rf = randomForest(status ~ .,data = df_train,
                  ntree=100, mtry=9,
                  importance=TRUE,replace=FALSE)
saveRDS(rf, file = "RF_unbalanced.rds")
rf=readRDS(file = "RF_unbalanced.rds")
min.err = min(data.frame(rf$err.rate)['OOB'])
min.err.index = which(data.frame(rf$err.rate)['OOB']==min.err)
min.err
min.err.index
# Generate textual output of the 'Random Forest' model.
rf
# Calculate the Area Under the Curve (AUC).
pROC::roc(rf$y, as.numeric(rf$predicted))
# Calculate the AUC Confidence Interval.
pROC::ci.auc(rf$y, as.numeric(rf$predicted),FALSE)
# List the importance of the variables.
rn <- round(randomForest::importance(rf), 2)
df_importance = as.data.frame(rn[order(rn[,3], decreasing=TRUE),])
df_importance
# Plot the error rate against the number of trees.
plot(rf, main="")
legend("topright", c("OOB", "Good", "Bad"), text.col=1:6, lty=1:3, col=1:3,inset=c(0.05, 0.15))
title(main="Error Rates Random Forest - Unbalanced Dataset")

Df_importance = data.frame(Variable = rownames(rf$importance),
                           Importance = as.data.frame(rf$importance)$MeanDecreaseAccuracy)
Df_importance = arrange(Df_importance,desc(Importance))
df_importance = Df_importance[1:20,]
```

```
df_importance$Variable =
factor(df_importance$Variable,levels=rev(unique(df_importance$Variable)))
ggplot(df_importance,ggplot2::aes(x    = Variable,
                                  y    = Importance,
                                  fill = Variable)) +
  ggplot2::geom_bar(stat     = "identity",
                    position = "identity") +
  ggplot2::coord_flip() +
  ggplot2::theme(axis.text.x     = ggplot2::element_text(angle=45, hjust=1),
                 legend.position = "none")

### Validate Data ======================================================
df_valid = df_train_valid[validate,]
df_valid_inputs = df_valid
df_valid_inputs$status = NULL
df_valid_output = df_valid$status
df_valid_output = factor(df_valid_output,c('Good','Bad'))
rf.pred_prob = predict(rf,newdata = df_valid_inputs,type = 'prob')[,2]
pred <- prediction(rf.pred_prob,df_valid_output)
# Convert rate of positive predictions to percentage.
per <- performance(pred, "lift", "rpp")
per@x.values[[1]] <- per@x.values[[1]]*100
# Plot the lift chart.
ROCR::plot(per, col="#CC0000FF", lty=1, xlab="Caseload (%)", add=FALSE)
legend("topright", c("Random Forest"), col=rainbow(1, 1, .8), lty=1:1, title="Models",
inset=c(0.05, 0.7))
title(main="Lift Chart [validate]")
grid()

r = roc(as.numeric(df_valid_output),as.numeric(rf.pred_prob))
plot.roc(r)
r
# Area under the curve: 0.7261
df_valid_output_num = as.numeric(df_valid_output)
error = (rf.pred_prob+1) - df_valid_output_num
MSE = mean(error^2)
MAPE = 100*mean(abs(error/df_valid_output_num))
RMSE = MSE^0.5
perf.df = data.frame(MSE,MAPE,RMSE)
perf.df
#       MSE     MAPE      RMSE
# 0.1494692 23.47407 0.3866125
rf.pred1 = ifelse(rf.pred_prob<0,'Good','Bad')
rf.pred1 = as.factor(rf.pred1)
rf.pred1 = factor(rf.pred1,levels = c('Good','Bad'))
CM = confusionMatrix(rf.pred1,df_valid_output)
l_Sensitivity = c(CM$byClass['Sensitivity'])
l_Specificity = c(CM$byClass['Specificity'])
l_Accuracy = c(CM$overall['Accuracy'])
l_cutoff = seq(0.1,1,by=0.1)
for (cutoff in l_cutoff) {
  rf.pred1 = ifelse(rf.pred_prob<cutoff,'Good','Bad')
  rf.pred1 = as.factor(rf.pred1)
```

```
  rf.pred1 = factor(rf.pred1,levels = c('Good','Bad'))
  CM = confusionMatrix(rf.pred1,df_valid_output)
  l_Sensitivity = c(l_Sensitivity,CM$byClass['Sensitivity'])
  l_Specificity = c(l_Specificity,CM$byClass['Specificity'])
  l_Accuracy = c(l_Accuracy,CM$overall['Accuracy'])
}
l_cutoff = c(1,l_cutoff)
df_cutoff = data.frame(cutoff=l_cutoff,sen=l_Sensitivity,spe=l_Specificity,acu=l_Accuracy)
df_cutoff$return = (-0.128+0.128*df_cutoff$sen+0.096*df_cutoff$spe)
return_max = max(df_cutoff$return)
cutoff_best = filter(df_cutoff,return==return_max)
cutoff_best
# cutoff      sen       spe       acu     return
#    0.3 0.782118 0.5298159 0.7280446 0.02297343
cutoff = as.numeric(cutoff_best[1])
rf.pred1 = ifelse(rf.pred_prob<cutoff,'Good','Bad')
rf.pred1 = as.factor(rf.pred1)
rf.pred1 = factor(rf.pred1,levels = c('Good','Bad'))
CM = confusionMatrix(rf.pred1,as.factor(df_valid_output))
CM
df1 = df_cutoff
library(plotly)
plot_ly(df1, x = ~sen, y = ~spe, z = ~return,
        marker = list(color = ~return, colorscale = c('#FFE1A1', '#683531'), showscale = TRUE))
%>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Sensitivity'),
                      yaxis = list(title = 'Specificity'),
                      zaxis = list(title = 'Extra Return')),
         annotations = list(
           x = 1.13,
           y = 1.05,
           text = 'Return Ratio',
           xref = 'paper',
           yref = 'paper',
           showarrow = FALSE
         ))

plot_ly(df1, x = ~sen, y = ~spe, type = 'scatter', mode = 'markers', color = ~return)
```

```
### Test data=============================================
Df_test = Df[Df$date>=as.Date('2016-08-01'),]
payment = Df_test$payment
df_test = Df_test
df_test$payment = NULL
```

```
df_test_inputs = df_test
df_test_inputs$status = NULL
df_test_output = df_test$status
rf.pred = predict(rf,newdata = df_test_inputs)
rf.pred_prob = predict(rf,newdata = df_test_inputs,type = 'prob')[,2]
rf.pred1 = ifelse(rf.pred_prob<0.3,'Good','Bad')
rf.pred1 = as.factor(rf.pred1)
rf.pred1 = factor(rf.pred1,levels = c('Good','Bad'))
CM = confusionMatrix(rf.pred1,df_test_output)
CM
r = roc(as.numeric(df_test_output),as.numeric(rf.pred_prob))
plot.roc(r)
r
#Area under the curve: 0.6964
df =
data.frame(pred=rf.pred1,actual=df_test_output,amount=df_test$loan_amnt,pay=payment,grade=df_tes
t$grade)
df$grade = as.character(df$grade)
df$grade[df$grade %in% c('E','F','G')] = 'E/F/G'
df$grade = factor(df$grade)
df_baseline = df
df_model = filter(df,pred=='Good')

ratioFun = function(df){
  df_roi = data.frame(amount=tapply(df$amount, df$grade, sum),
                      pay=tapply(df$pay, df$grade, sum))
  df_roi = rbind.data.frame(df_roi,All=c(sum(df$amount),sum(df$pay)))
  df_roi = mutate(df_roi,
                  return = pay-amount,
                  roi=pay/amount-1,
                  grade=rownames(df_roi))
  df_roi$grade = factor(df_roi$grade,as.factor(df_roi$grade))
  df_roi
}
df1 = ratioFun(df_baseline) %>% mutate(model='baseline')
df2 = ratioFun(df_model) %>% mutate(model='model')
df_roi = rbind.data.frame(df1,df2)
df_roi[is.na(df_roi)]=0
# df_roi$grade = rownames(df_roi)
# df_roi = gather(df_roi,`baseline`, `model`, key = "model", value = "roi")

p_roi

ggplot(df_roi) +
  geom_bar(stat = "identity",position="dodge",
           aes(x = grade, y = return/1000000, fill = model, group = model)) +
  geom_line(aes(x=grade,y=roi*100*4,group=model,colour = model),size=1.3) +
  geom_point(aes(x=grade,y=roi*100*4,group=model)) +
  xlab("Grade") +
  labs(title = "Return of 2017 loans")+
  scale_y_continuous(name = "Return(Million)",
                     sec.axis = sec_axis(~./4, name = "Return Ratio(%)",
                                         labels = function(b) { paste0(round(b, 0), "%")}))
```

```
ggplot() +
  geom_bar(mapping = aes(x = dt$when, y = dt$numinter), stat = "identity", fill = "grey") +
  geom_line(mapping = aes(x = dt$when, y = dt$prod*5), size = 2, color = "blue") +
  scale_x_date(name = "Day", labels = NULL) +
  scale_y_continuous(name = "Interruptions/day",
                     sec.axis = sec_axis(~./5, name = "Productivity % of best",
                                         labels = function(b) { paste0(round(b * 100, 0),
"%")}))


library(ggthemes)
ggplot(df_roi, aes(x = grade, y = return/1000000, fill = model, group = model)) +
  geom_bar(stat = "identity",position="dodge") +
  theme_economist ()+
  xlab("Grade") +
  ylab("Return(Million)")

df_pick = as.data.frame(table(df$pred,df$grade))
df_pick = spread(df_pick, key = Var1, value = Freq)
df_pick$Var2 = as.character(df_pick$Var2)
df_pick = rbind.data.frame(df_pick,c('All',sum(df_pick$Good),sum(df_pick$Bad)))
df_pick$Var2 = factor(df_pick$Var2,levels=as.factor(df_pick$Var2))
df_pick$Good = as.numeric(df_pick$Good)
df_pick$Bad = as.numeric(df_pick$Bad)
df_pick = mutate(df_pick,
                 pick_ratio=Good/(Good+Bad),
                 grade = Var2,
                 model = 1)
df1 = ratioFun(df_baseline) %>% mutate(model='baseline')
df_pick$roi = df1$roi
ggplot(df_pick,aes(x=grade,y=pick_ratio,colour=grade)) +
  geom_line(colour="black",aes(group=model),size=1) +
  geom_point(aes(size=roi)) +
  xlab("Grade") +
  ylab("Percent Picked(%)") +
  labs(title = "Loan picked",
       size=expression(atop("Actual", "Return Ratio")))

df_pick1 = as.data.frame(table(df$actual,df$pred,df$grade))[-c(1:4),]
colnames(df_pick1)=c('actual','pred','grade','Freq')
df_pick1 = spread(df_pick1,key=pred,value=Freq)
df_pick1 = mutate(df_pick1,pick_ratio = Good/(Good+Bad))
ggplot(df_pick1,aes(x=grade,y=pick_ratio,group=actual)) +
  geom_line(aes(colour = actual),size=1) +
  geom_point(size=2) +
  xlab("Grade") +
  ylab("Percent Picked(%)") +
  labs(title = "Loan picked")


df_pred_good = filter(df,pred=='Good')
pred_total_amount = sum(df_pred_good$amount)
```

```
pred_total_pay = sum(df_pred_good$pay)
pre_total_return = pred_total_pay-pred_total_amount
actual_total_amount = sum(df$amount)
actual_total_pay = sum(df$pay)
actual_total_return = actual_total_pay-actual_total_amount
return = pre_total_return-actual_total_return
return_ratio = (sum(df_pred_good$pay)/sum(df_pred_good$amount))-(sum(df$pay)/sum(df$amount))
return # 64568053
return_ratio # 0.05487328



ggplot(df,aes(x=grade,y=roi*100,group=model_name)) +
  geom_line(aes(colour = model_name)) +
  geom_point() +
  xlab("Grade") +
  ylab("ROI(%)") +
  labs(title = "Return on identified 2015 loans")


library(ggplot2)
df$actual = as.factor(df$actual)
df$pred = as.factor(df$pred)
ggplot(df,aes(x=actual,y=pay_ratio,fill=pred))+
  geom_violin()
ggplot(df,aes(x=amount,y=pay,color=pred,alpha=0.2))+
  geom_point()
```