Credit Scoring & Data Mining

Credit Scorecard

Word Count: 2066

2020-5-13

CONTENTS

ABSTRACT

This report uses EM Model to deal with missing values, remove outliers, and use imblearn package to deal with sample imbalance. Then, this report build a credit scorecard with a 0.84 accuracy using Binning, Weight of Evidence(WoE), Information Value(IV) and Logistic Regression. This report primarily uses SPSS in the exploratory analysis process, and Python for model building. All data and codes in this report are available at https://github.com/mktming/creditscorecard (Open Access After 5/6/2020)

EXPLORATORY ANALYSIS

First, perform descriptive statistics and normality test on the original data set (cs-train.csv) (as shown in ***Table app-1*** and ***Table app-2,*** in Appendix), and then remove the duplicate cases in SPSS (a total of 6096 duplicates cases, 149391 cases remain after removing).

Missing Value

If there are a small number of missing values (less than 5% of the total number of cases) and these values can be considered as missing randomly, then deleting missing values is relatively safe. However, the missing ratios of the two variables *MonthlyIncome* and *NumberOfDependents* are 19.56% and 2.56% respectively. Because of the huge amount of missing data, this report uses EM method to estimate missing value through SPSS.

EM method (Expectation-Maximization algorithm) assumes a distribution of partially missing data and makes inferences based on the probability under this distribution. Each iteration includes an E step and an M step. Given the observations and the current parameter estimates, E step finds the conditional expected values of missing data and replaces these missing values with the expected values. In M step, even if the missing data is filled in, the maximum likelihood estimate of the parameter will be calculated. For *NumberOfDependents,* its true result should be an integer. However, decimals appear after the EM method, so the Excel Round function is used for rounding. EM Model Summary are shown in ***Table app-3*** in Appendix**.**

Outliers

By drawing box plots (***Figure app-1***), this report finds that there are many outliers in the variables. But these outliers can be considered reasonable, such as people with extremely high monthly income. Therefore, this report only deletes the 23 most significant outliers among them (as shown in Table app-4). These deleted outliers may not be wrong, but they become influential points in model fitting process, which in turn affects the model training process. In addition, it is found that there is a data with an age of 0, which should be due to an entry error (149367 cases left).

In addition, there are three indicators that look strange: *NumberOfTime3059DaysPastDueNotWorse*, *NumberOfTime6089DaysPastDueNotWorse* and *NumberOfTimes90DaysLate*. These three indicators are still 2 at a 99% distribution, however the maximum value is 98 (or 96). That means that a person is overdue 98 times within 35 to 59 days in the past two years. But there are only six 60 days a year. There are 225 cases of this situation. This report believes that these cases are to some kinds of abnormality, and they should be deleted (149,142 cases left). Therefore, data set train_clean2.csv

Samples Imbalance

In the descriptive statistical results (Table app-1), there is a partial distribution. This will cause the model to be more biased towards zero during the model training process. For example, a fool model forever output 0,and it can also achieve 93.38% accuracy, but obviously this model is not helpful for banking business. Therefore, this report uses Python's imblearn package to balance the samples to get the new dataset train_clean3.csv

- Number of original samples: 149142; 1 accounts for 6.62%; 0 accounts for 93.38%
- Number of new samples: 278548; 1 accounts for 50.00%; 0 accounts for 50.00%

The descriptive statistics and normality test of the data after sample equalization are shown in ***Table app-5*** and ***Table app-6***

MODEL BUILDING

Collinearity & Data Segmentation

Next, this report uses the seaborn package in Python to check the collinearity of the data. As can be seen from ***Figure app-2***, correlation between variables is not significant. Only the correlation coefficient of *NumberOfOpenCreditLinesAndLoans* and *NumberRealEstateLoansOrLines* is 0.48. In order to verify the fitting effect of the model, this report divides train_clean3.csv according to the ratio of 7: 3 into training set (TrainData.csv) and test set (TestData.csv).

Binning

Binning is a term for continuous variable discretization. Credit scorecard commonly used Equal length intervals and Optimal Binning. Equal length intervals means that the intervals of the segments are consistent, for example, age is ten years as a segment; Optimal Binning is also called supervised discretization. based on conditional inference to find a better grouping, using Recursive Partitioning algorithm to divide continuous variables into segments. This report first does Optimal Binning for continuous variables. When the distribution of continuous variables does not meet the requirements of Optimal Binning, it is then considered to divide the continuous variables using Equal length intervals

WoE

WoE (Weight of Evidence) is to calculate the WoE value of each binning segment and observe their trend changing with indicators. Calculated as follows. According to the logarithmic transformation curve, WoE takes the value of all real numbers. In the part greater than 0, the larger the WoE, the greater the possibility of default samples in the group (positive effect). In the less than 0 part, the smaller the WoE, the more a kind of reverse effect. Therefore, WoE reflects the influence of the value of the independent variable on the target variable.

$$WOE_i = ln\frac{P(y_i)}{P(n_i)}$$

where

$P\ (yi)$ represents the proportion of default samples of group i in the all default samples;

$P\ (ni)$ represents the proportion of non-default samples of group i in the all non-default samples;

IV

IV (Information Value) considers the proportion of samples in the group to the total sample, which is equivalent to the weighted sum of WoE. The calculation formula is as follows:

$$IV_i = \sum_{i=1}^{n} (P(y_i) - P(n_i))\ ln\frac{P(y_i)}{P(n_i)}$$

The above formula can calculate IV value of each variable as shown in ***Figure 1***.

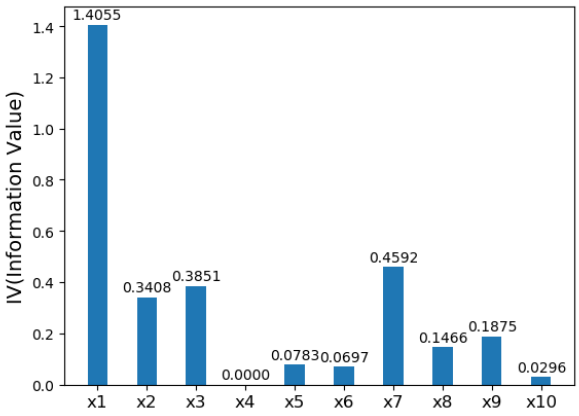

Figure 1 *Information Value*

The IV values of four variables (*DebtRatio*, *MonthlyIncome*, *NumberOfOpenCreditLinesAndLoans* and *NumberOfDependents*) are significantly lower, therefore delete them.

Logistic Regression

WOE can transform logistic regression into a standard scorecard format. Before establishing a logistic regression, this report converts the remaining six variables into their WoE values for credit score and saves them in the WoeData.csv. Then this report calls Python's statsmodels package to implement logistic regression. According to ***Table app-5***, all variables of logistic regression have passed the significance test.

Model Evaluation

Next, this report uses the TestData.csv (not cs-test.csv) reserved at the beginning of modeling for verification. In Python, sklearn.metrics is used to automatically calculate ROC and AUC to evaluate the fitting ability of the model. It can be seen from ***Figure 2*** that the AUC value is 0.84. The prediction effect of this model is good, and the accuracy rate is relatively high.

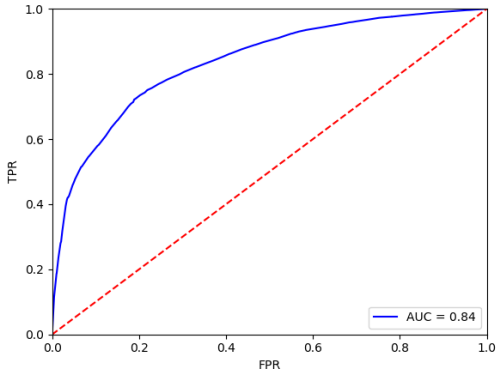

Figure 2 *ROC Curve*

Credit Scorecard

This report has basically completed the work related to modeling, and used the ROC curve to verify the predictive ability of the model. The next step is to convert the Logistic model into a standard scorecard format. This report takes 300 as the basic score and PDO is 20 (every 20 points is better than double). Based on this, get the scorecard as

shown in ***Table 1***

Table 1 *Credit Scorecard*

RevolvingUtilizationOfUnsecuredLines	score	Age	score	NumberOfTime3059DaysPastDueNotWorse	score	NumberOfTimes90DaysLate	score	NumberRealEstateLoansOrLines	score	NumberOfTime6089DaysPastDueNotWorse	score
(-0.001, 0.0984]	51	[21.0, 32.0]	-8	(-inf, 1.0]	16	(-inf, 1.0]	16	(-inf, 0.0]	-7	(-inf, 0.0]	6
(0.0984, 0.465]	8	(32.0, 38.0]	-6	(1.0, 2.0]	-39	(1.0, 2.0]	-116	(0.0, 1.0]	3	(0.0, 1.0]	-62
(0.465, 0.875]	-22	(38.0, 42.0]	-5	(2.0, 3.0]	-62	(2.0, 3.0]	-152	(1.0, 2.0]	11	(1.0, 3.0]	-79
(0.875, 22198.0]	-27	(42.0, 46.0]	-4	(3.0, 5.0]	-69	(3.0, 5.0]	-172	(2.0, 3.0]	7	(3.0, inf]	-73
		(46.0, 50.0]	-4	(5.0, inf]	-54	(5.0, inf]	-148	(3.0, inf]	-1		
		(50.0, 55.0]	-1								
		(55.0, 60.0]	3								
		(60.0, 67.0]	11								
		(67.0, 109.0]	21								

Apply the scorecard to TrainData.csv can get ScoreData.csv. The average value of ScoreData.csv is 220.37. Since the number of customers with *SeriousDlqin2yrs* = 1/0 in the data set is the same, the average value is used as the basis for segmentation. That is

$$\begin{cases} If \quad Score >= \ 220, SeriousDlqin2yrs = 1 \\ Else\ if\ Score < \ 220, SeriousDlqin2yrs = 0 \end{cases}$$

Applying this model to cs-test.csv (processed for outliers and missing values) can get the prediction result of its *SeriousDlqin2yrs*. As shown in cs-test-score.csv. Some results are shown in ***Table 2***.

Table 2 *Prediction Results for cs-test.csv*

SeriousDlq	RevolvingU	age	NumberOf	DebtRatio	MonthlyInc	NumberOf	NumberOf	NumberRe	NumberOf	NumberOf	BaseScore	x1	x2	x3	x7	x8	x9	Score
1	0.041905	45	0	0.202724	6387	6	0	1	0	43	441	51	-4	-40	-120	11	-62	277
1	0.227651	41	0	4151	0	10	0	1	0	20	441	8	-5	-40	-120	11	-62	233
0	0.957847	47	0	0.477725	9000	16	0	2	0	10	441	-28	-4	-40	-120	9	-62	196
1	0.054512	52	0	0.010944	18000	11	0	0	0	10	441	51	-3	-40	-120	4	-62	271
1	0.021415	55	0	0.780105	2100	4	0	1	0	10	441	51	2	-40	-120	11	-62	283
0	0.53263	48	1	0.309114	16666	18	0	2	0	9	441	-22	-3	-62	-120	9	-62	181
0	0.581969	53	0	0.205485	12871	16	0	0	0	9	441	-22	2	-40	-120	4	-62	203
0	0.486619	46	0	0.179879	8766	4	0	1	0	9	441	-22	-4	-40	-120	11	-62	204
0	1	37	0	0.124837	13000	2	1	0	0	8	441	-28	-6	-40	-153	4	-62	156
0	0.845031	43	2	0.145665	6665	6	0	0	1	8	441	-22	-5	-62	-120	4	-78	158
0	1	40	0	0.172746	5400	2	0	0	0	8	441	-28	-5	-40	-120	4	-62	190
0	1	50	0	0.982509	2000	2	0	1	0	8	441	-28	-3	-40	-120	11	-62	199
1	0.20654	36	0	0.31402	4400	3	0	1	0	8	441	8	-6	-40	-120	11	-62	232
1	0.160535	44	0	0.210802	8775	12	0	2	0	8	441	8	-4	-40	-120	9	-62	232

APPLICATION

Credit scorecard is a supervised learning model, which divides the customer group into non-default customers and default customers. Analyze the characteristics of the two types of customer groups from historical data, and establish a mathematical model according to the classification rules, and then use the model to calculate the lender's default risk as the basis for consumer credit decision. The model can be used for pre-loan approval to predict the borrower's probability of default. Compared to most machine learning models, credit scorecard is easy to explain and understand. Specifically, in the case of most machine learning models are black box models, if a customer is refused a loan because of credit problems, the bank teller can only tell this customer that it was rejected, but cannot tell why. The credit scorecard can tell customers why the loan was refused (for example, too young, too many credit card overdue, low monthly salary, etc.). This is conducive for banks to improve customer satisfaction to obtain a better brand reputation and encourage customers to more targeted increase their credit score and thereby increase loan turnover. However, compared with the scorecard model, the risk control model designed based on the machine learning does not need to be established under strict statistical assumptions, and the prediction effect is more accurate.

LITERATURE REVIEW

Maldonado, S., Peters, G., & Weber, R. (2020). Credit scoring using three-way decisions with probabilistic rough sets. *Information Sciences*, *507*, 700-714.

As financial data becomes more comprehensive, credit scoring has become one effective way to predict financial risks. Personal credit scores revolve around data, which includes financial data and big data derived from daily life. In the past year (2019-2020), many scholars are interested in predicting whether a customer will default or not. Are people who have overdue experience more likely to default (Xiao et al., 2020)? Can social network information help predict default (Luo,2019; Niu et al., 2019)? Which machine learning and traditional models have higher accuracy (Pei et al., 2020; Tripathi, 2019)? Whether focusing on the selection of independent variables, the source of the data set, or model selection, their final goal is to make prediction as accurate as possible. However, methods proposed in the above literature significantly increases the banks' data acquisition cost and labor cost.

For solving this problem, Maldonado et al. (2020) proposed to apply the three-way decision framework to one data set provided by the Chilean bank (including 7309 cases). This three-way decision framework uses logistic regression as basic classifier for the purpose of reducing bank costs. Borrowers are divided into three categories by this framework: For repeat customers (RET) and most new customers (NEW), the basic credit assessment is sufficient to determine their credit application. However, some new customers need to be interviewed and further detailed analysis. In this way, only a small number of new customers need to fill in all the information, while most customers only need to fill out a basic form. This framework not only does not reduce the prediction accuracy rate compared to mainstream machine learning methods (86.94%), also significantly reduces data search costs, labor costs and computer operating costs. Thereby make the bank gain more 1% profits in personal loan business.

This change has had a huge impact on repeat customers and new customers. Using three-way decision framework rather than traditional machine learning models of reporting revenue gains considerable profits in highly competitive markets, which are characterized by low personal profit margins for huge number of customers. In addition to financial advantage, a more efficient credit granting process also leads to faster decision-making speed, thereby gaining a significant strategic advantage among competitors. For consumers, faster credit decisions will lead to higher satisfaction.

However, this report believes that the framework is only applicable to personal petty loans. When it comes to large loans such as entrepreneurship loans or juridical person loans, banks must make decisions after grasping all the available information of the enterprise or legal person. Because such loans often involve huge amounts and extremely high interest rates and companies often fail to repay after bankruptcy. Then, the cost of investigation is insignificant. Besides, compared with credit scorecard, this framework is still lack of explanation. Credit scorecard can tell customers why the loan was refused (for example, too young, too many credit card overdue, low monthly salary, etc.). Customers who was refused loans for unknown reasons may have negative emotions for the bank and affect more people through social networks.

REFERENCES

Luo, C. (2019). A comprehensive decision support approach for credit scoring. *Industrial Management & Data Systems*.

Maldonado, S., Peters, G., & Weber, R. (2020). Credit scoring using three-way decisions with probabilistic rough sets. *Information Sciences*, *507*, 700-714.

Niu, B., Ren, J., & Li, X. (2019). Credit Scoring Using Machine Learning by Combing Social Network Information: Evidence from Peer-to-Peer Lending. *Information*, *10*(12), 397.

Pei, S., Shen, T., Wang, X., Gu, C., Ning, Z., Ye, X., & Xiong, N. (2020). 3DACN: 3D Augmented convolutional network for time series data. *Information Sciences*, *513*, 17-29.

Tripathi, D., Edla, D. R., Cheruku, R., & Kuppili, V. (2019). A novel hybrid credit scoring model based on ensemble feature selection and multilayer ensemble classification. *Computational Intelligence*, *35*(2), 371-394.

Xiao, J., Zhou, X., Zhong, Y., Xie, L., Gu, X., & Liu, D. (2020). Cost-sensitive semi-supervised selective ensemble model for customer credit scoring. *Knowledge-Based Systems*, *189*, 105118.

APPENDIX

Figures and Tables

Table app-1 *Descriptive Statistic for cs-train.csv*

					Mean				Skewness		Peakedness	
	N	Range	Min	Max	Statictic	Std Err	Std Dev	Variance	Statistic	Std Err	Statistic	Std Err
SeriousDlqin2yrs	150000	1	0	1	.07	.001	.250	.062	3.469	.006	10.033	.013
RevolvingUtilizationOfUnsecuredLines	150000	50708.0000	.0000	50708.0000	6.048438	.6448656	249.7553706	62377.745	97.632	.006	14544.713	.013
age	150000	109	0	109	52.30	.038	14.772	218.208	.189	.006	-.495	.013
NumberOfTime3059DaysPastDueNotWorse	150000	98	0	98	.42	.011	4.193	17.579	22.597	.006	522.377	.013
DebtRatio	150000	329664.0000	.000000	329664.0000	353.0050758	5.261624802	2037.818523	4152704.333	95.158	.006	13734.289	.013
MonthlyIncome	150000	3325666.075	-316916.075	3008750.000	6231.918166	33.64945146	13032.37651	169842837.5	122.713	.006	23236.215	.013
NumberOfOpenCreditLinesAndLoans	150000	58	0	58	8.45	.013	5.146	26.481	1.215	.006	3.091	.013
NumberOfTimes90DaysLate	150000	98	0	98	.27	.011	4.169	17.383	23.087	.006	537.739	.013
NumberRealEstateLoansOrLines	150000	54	0	54	1.02	.003	1.130	1.276	3.482	.006	60.477	.013
NumberOfTime6089DaysPastDueNotWorse	150000	98	0	98	.24	.011	4.155	17.266	23.332	.006	545.683	.013
NumberOfDependents	150000	25	-5	20	.75	.003	1.104	1.219	1.605	.006	3.124	.013
MissingValueCount	150000	2	0	2	.22	.001	.476	.226	2.021	.006	3.331	.013
valid cases	150000											

Table app-2 *Normality Test for cs-train.csv*

	SeriousDlqin2yrs	Statistic	Degree of Freedom	Sig.
RevolvingUtilizationOfUnsecuredLines	0	.505	139956	.000
	1	.500	10021	.000
age	0	.032	139956	.000
	1	.041	10021	.000
NumberOfTime3059DaysPastDueNotWorse	0	.462	139956	.000
	1	.419	10021	.000
DebtRatio	0	.426	139956	.000
	1	.433	10021	.000
MonthlyIncome	0	.226	139956	.000
	1	.200	10021	.000
NumberOfOpenCreditLinesAndLoans	0	.110	139956	.000
	1	.103	10021	.000
NumberOfTimes90DaysLate	0	.484	139956	.000
	1	.429	10021	.000
NumberRealEstateLoansOrLines	0	.230	139956	.000
	1	.236	10021	.000
NumberOfTime6089DaysPastDueNotWorse	0	.483	139956	.000
	1	.451	10021	.000
NumberOfDependents	0	.347	139956	.000
	1	.294	10021	.000
MissingValueCount	0	.482	139956	.000
	1	.499	10021	.000

Table app-3 *EM Model Summary*

RevolvingUtilizationOfUnsecuredLines	age	DebtRatio	MonthlyIncome	NumberOfOpenCreditLinesAndLoans	NumberRealEstateLoansOrLines	NumberOfDependents
5.310000E+000	52.41	3.44475020E+002	6709.32	8.45	1.01	.76

	RevolvingUtilizationOfUnsecuredLines	age	DebtRatio	MonthlyIncome	NumberOfOpenCreditLinesAndLoans	NumberRealEstateLoansOrLines	NumberOfDependents
RevolvingUtilizationOfUnsecuredLines	3.8477191E+004						
age	-2.0963058E+001	218.441					
DebtRatio	1.2299969E+003	832.312	2.665367189E+006				
MonthlyIncome	6.3547126E+003	11305.781	-2.306033695E+005	1332641751.162			
NumberOfOpenCreditLinesAndLoans	-1.1210074E+001	10.932	5.468221107E+002	6410.504	26.462		
NumberRealEstateLoansOrLines	8.6594510E-001	.438	2.342479973E+002	1894.831	2.462	1.233	
NumberOfDependents	4.4721740E-001	-3.718	-8.597943369E+001	866.869	.390	.155	1.294

	RevolvingUtilizationOfUnsecuredLines	age	DebtRatio	MonthlyIncome	NumberOfOpenCreditLinesAndLoans	NumberRealEstateLoansOrLines	NumberOfDependents
RevolvingUtilizationOfUnsecuredLines	1						
age	-.007	1					
DebtRatio	.004	.034	1				
MonthlyIncome	.001	.021	-.004	1			
NumberOfOpenCreditLinesAndLoans	-.011	.144	.065	.034	1		
NumberRealEstateLoansOrLines	.004	.027	.129	.047	.431	1	
NumberOfDependents	.002	-.221	-.046	.021	.067	.122	1

MCAR Test: Chi-Square = 22672.888, Degree of Freedom = 11, Sig. = .000

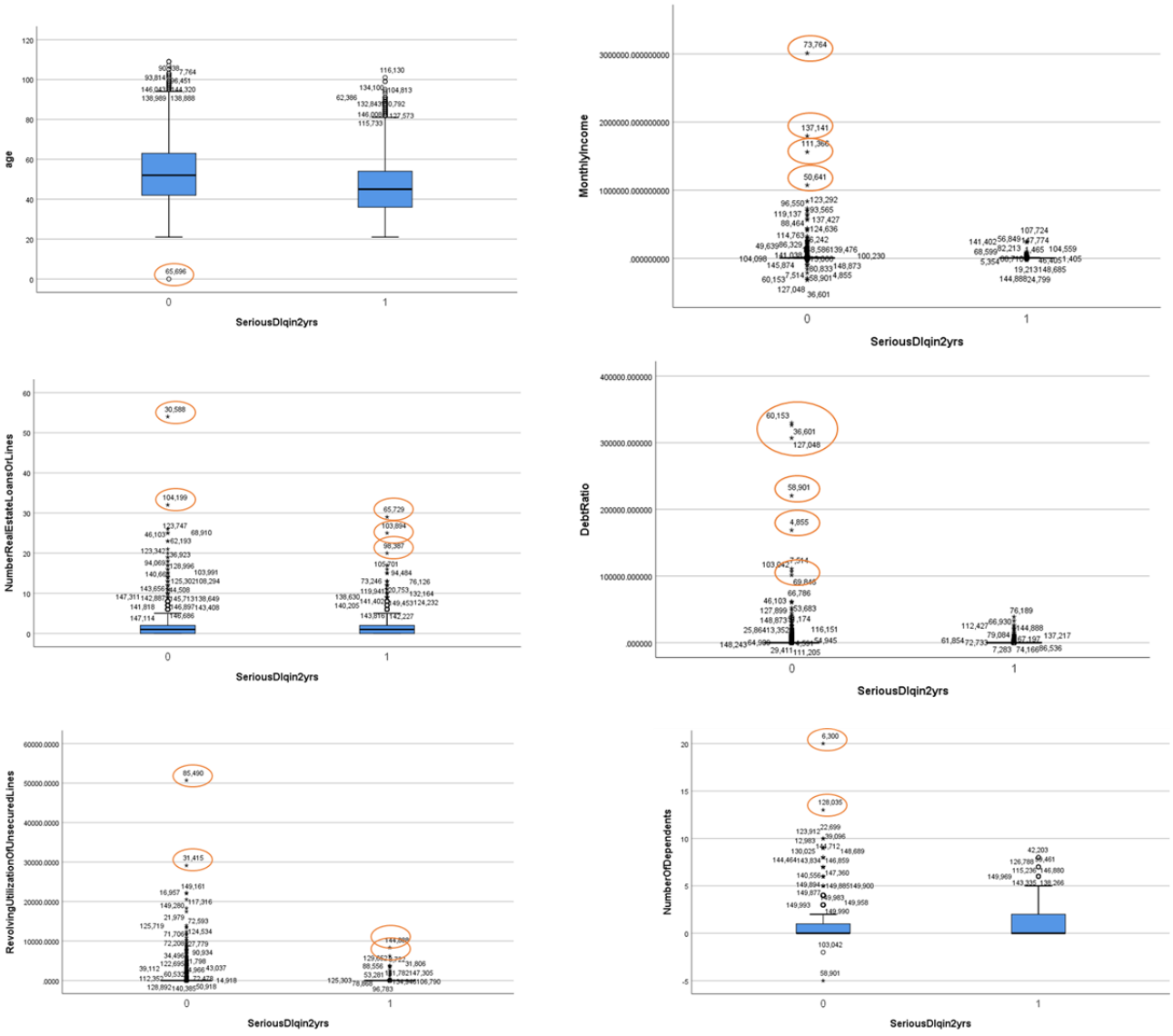

Figure app-1 *Box Plots*

Table app-4 *Deleted Cases*

deleted value ID	Variables	SeriousDlqin2yrs	Value
6300	NumberOfDependents	0	20
128036	NumberOfDependents	0	13
60153	DebtRatio	0	329664
36601	DebtRatio	0	326442
127048	DebtRatio	0	307001
58901	DebtRatio	0	220516
4855	DebtRatio	0	168835
7514	DebtRatio	0	110952
103042	DebtRatio	0	106885
69846	DebtRatio	0	101320
73764	MonthlyIncome	0	3008750
137141	MonthlyIncome	0	1794060
111366	MonthlyIncome	0	1560100
50641	MonthlyIncome	0	1072500
30588	NumberRealEstateLoansOrLines	0	54
104199	NumberRealEstateLoansOrLines	0	32
65729	NumberRealEstateLoansOrLines	1	29
103894	NumberRealEstateLoansOrLines	1	25
98387	NumberRealEstateLoansOrLines	1	20
144888	RevolvingUtilizationOfUnsecuredLines	1	8328
114779	RevolvingUtilizationOfUnsecuredLines	1	6190
85490	RevolvingUtilizationOfUnsecuredLines	0	50708
31415	RevolvingUtilizationOfUnsecuredLines	0	29110

Table app-5 *Descriptive Statistic for train_clean3.csv*

					Mean				Skewness		Peakedness	
	N	Range	Min	Max	Statictic	Std Err	Std Dev	Variance	Statictic	Std Err	Statistic	Std Err
SeriousDlqin2yrs	278548	1	0	1	.50	.001	.500	.250	.000	.005	-2.000	.009
RevolvingUtilizationOfUnsecuredLines	278548	22198.0000	.0000	22198.0000	4.196220	.2908258	153.4910209	23559.493	74.063	.005	7538.021	.009
age	278548	88	21	109	49.11	.026	13.895	193.073	.368	.005	-.256	.009
NumberOfTime3059DaysPastDueNotWorse	278548	13	0	13	.42	.002	.875	.765	2.892	.005	11.772	.009
DebtRatio	278548	61907.00000	.00000000	61907.00000	321.6236526	2.251682050	1188.384799	1412258.430	10.514	.005	249.810	.009
MonthlyIncome	278548	883128.4403	-48088.4403	835040.0000	5807.513412	12.86888925	6791.896914	46129863.68	38.364	.005	3285.129	.009
NumberOfOpenCreditLinesAndLoans	278548	57	0	57	7.99	.010	5.034	25.337	1.131	.005	2.676	.009
NumberOfTimes90DaysLate	278548	17	0	17	.23	.001	.722	.521	5.054	.005	39.046	.009
NumberRealEstateLoansOrLines	278548	26	0	26	.88	.002	1.106	1.224	2.654	.005	18.205	.009
NumberOfTime6089DaysPastDueNotWorse	278548	11	0	11	.12	.001	.424	.180	5.009	.005	36.502	.009
NumberOfDependents	278548	13	0	13	.70	.002	1.023	1.048	1.567	.005	2.574	.009
valid cases	278548											

Table app-6 *Normality Test for train_clean3.csv*

	SeriousDlqin2yrs	Statistic	Degree of Freedom	Sig.
RevolvingUtilizationOfUnsecuredLines	0	.505	139274	.000
	1	.501	139274	.000
age	0	.032	139274	.000
	1	.039	139274	.000
DebtRatio	0	.426	139274	.000
	1	.435	139274	.000
MonthlyIncome	0	.226	139274	.000
	1	.183	139274	.000
NumberOfOpenCreditLinesAndLoans	0	.111	139274	.000
	1	.101	139274	.000
NumberRealEstateLoansOrLines	0	.230	139274	.000
	1	.289	139274	.000

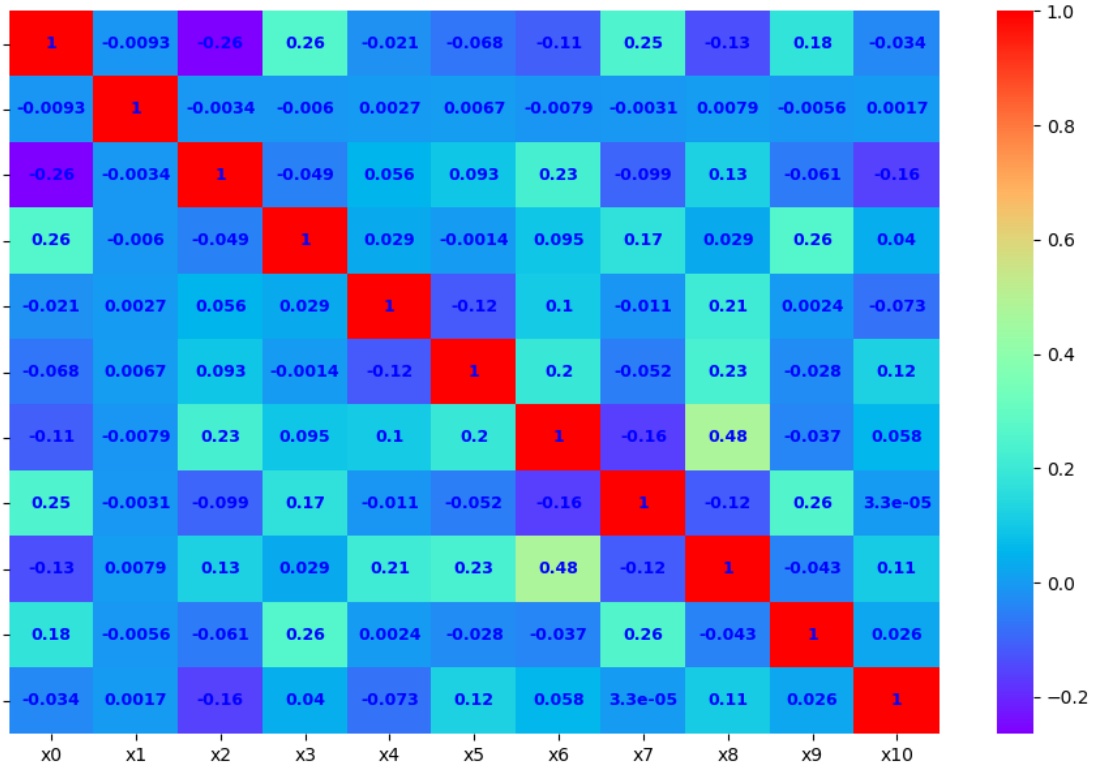

Figure app-2 *Correlation Matrix*

Table app-7 *Logistic Model Summary*

	coef	std err	z	P>|z|	[0.025	0.975]
const	7.6368	0.103	74.252	0.000	7.435	7.838
RevolvingUtilizationOfUnsecuredLines	0.8045	0.005	158.378	0.000	0.795	0.814
age	0.4836	0.010	48.982	0.000	0.464	0.503
NumberOfTime3059DaysPastDueNotWorse	1.5517	0.024	64.167	0.000	1.504	1.599
NumberOfTimes90DaysLate	2.2536	0.037	60.328	0.000	2.180	2.327
NumberRealEstateLoansOrLines	0.6126	0.026	23.210	0.000	0.561	0.664
NumberOfTime6089DaysPastDueNotWorse	1.5277	0.061	25.211	0.000	1.409	1.646

Python Codes

```
#utf-8  2020-04-28 20:35:44
#Samples Imbalance
import pandas as pd
import numpy as np
import imblearn
from imblearn.over_sampling import SMOTE

data=pd.read_csv('train_clean2.csv',index_col=0)
print(data.shape)

x=data.iloc[:,1:]
y=data.iloc[:,0]
print(y.value_counts())

n_sample=x.shape[0]
n_1_sample=y.value_counts()[1]
n_0_sample=y.value_counts()[0]
print('Samples:
{}; 1 {:.2%}; 0 {:.2%}'.format(n_sample,n_1_sample/n_sample,n_0_sample/n_sample))

sm = SMOTE(random_state=42)
x,y = sm.fit_sample(x,y)
n_sample_ = x.shape[0]
pd.Series(y).value_counts()
n_1_sample = pd.Series(y).value_counts()[1]
n_0_sample = pd.Series(y).value_counts()[0]
print('Samples:
{}; 1 {:.2%}; 0 {:.2%}'.format(n_sample_,n_1_sample/n_sample_,n_0_sample/n_sample_))

x=pd.DataFrame(x)
y=pd.DataFrame(y)
new_data=pd.concat([y,x],axis=1)
new_data.columns=data.columns
new_data.shape

new_data.to_csv('./train_clean3.csv',index=False,columns=new_data.columns)
```

```
#train & test split
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import seaborn as sns

#Collinear heat map
data = pd.read_csv('train_clean3.csv')
#Calculate the correlation coefficient of each variable
corr = data.corr()
xticks = ['x0','x1','x2','x3','x4','x5','x6','x7','x8','x9','x10']#xaxis label
yticks = list(corr.index)#yaxis label
fig = plt.figure()
ax1 = fig.add_subplot(1, 1, 1)
#Draw correlation coefficient heat map
sns.heatmap(corr, annot=True, cmap='rainbow', ax=ax1, annot_kws={'size': 9, 'weight': 'bold', 'color': 'blue'})
ax1.set_xticklabels(xticks, rotation=0, fontsize=10)
ax1.set_yticklabels(yticks, rotation=0, fontsize=10)
plt.show()

# training set and testing set
if __name__ == '__main__':
    data = pd.read_csv('train_clean3.csv')

    data['SeriousDlqin2yrs']=1-data['SeriousDlqin2yrs']
    Y = data['SeriousDlqin2yrs']
    X = data.iloc[:, 1:]
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
    # print(Y_train)
    train = pd.concat([Y_train, X_train], axis=1)
    test = pd.concat([Y_test, X_test], axis=1)
```

```
    clasTest = test.groupby('SeriousDlqin2yrs')['SeriousDlqin2yrs'].count()
    train.to_csv('TrainData.csv',index=False)
    test.to_csv('TestData.csv',index=False)
    print(train.shape)
    print(test.shape)
```

```
#utf-8 2020-05-03 19:34:32
#bin & weo & scoring calculating

import pandas as pd
import numpy as np
from pandas import Series,DataFrame
import scipy.stats.stats as stats
import matplotlib.pyplot as plt
import statsmodels.api as sm
import math

# define binning function
def mono_bin(Y, X, n = 20):
    r = 0
    good=Y.sum()
    bad=Y.count()-good
    while np.abs(r) < 1:
        d1 = pd.DataFrame({"X": X, "Y": Y, "Bucket": pd.qcut(X, n)})
        d2 = d1.groupby('Bucket', as_index = True)
        r, p = stats.spearmanr(d2.mean().X, d2.mean().Y)
        n = n - 1
    d3 = pd.DataFrame(d2.X.min(), columns = ['min'])
    d3['min']=d2.min().X
    d3['max'] = d2.max().X
    d3['sum'] = d2.sum().Y
    d3['total'] = d2.count().Y
    d3['rate'] = d2.mean().Y
    d3['woe']=np.log((d3['rate']/(1-d3['rate']))/(good/bad))
    d3['goodattribute']=d3['sum']/good
    d3['badattribute']=(d3['total']-d3['sum'])/bad
    iv=((d3['goodattribute']-d3['badattribute'])*d3['woe']).sum()
    d4 = (d3.sort_values(by = 'min'))
    print("=" * 60)
    print(d4)
    cut=[]
    cut.append(float('-inf'))
    for i in range(1,n+1):
        qua=X.quantile(i/(n+1))
        cut.append(round(qua,4))
    cut.append(float('inf'))
    woe=list(d4['woe'].round(3))
    return d4,iv,cut,woe
#binning function
def self_bin(Y,X,cat):
    good=Y.sum()
    bad=Y.count()-good
    d1=pd.DataFrame({'X':X,'Y':Y,'Bucket':pd.cut(X,cat)})
    d2=d1.groupby('Bucket', as_index = True)
    d3 = pd.DataFrame(d2.X.min(), columns=['min'])
    d3['min'] = d2.min().X
    d3['max'] = d2.max().X
    d3['sum'] = d2.sum().Y
    d3['total'] = d2.count().Y
    d3['rate'] = d2.mean().Y
    d3['woe'] = np.log((d3['rate'] / (1 - d3['rate'])) / (good / bad))
    d3['goodattribute'] = d3['sum'] / good
    d3['badattribute'] = (d3['total'] - d3['sum']) / bad
    iv = ((d3['goodattribute'] - d3['badattribute']) * d3['woe']).sum()
    d4 = (d3.sort_values(by='min'))
    print("=" * 60)
    print(d4)
    woe = list(d4['woe'].round(3))
    return d4, iv,woe
#woe
```

```
def replace_woe(series,cut,woe):
    list=[]
    i=0
    while i<len(series):
        value=series[i]
        j=len(cut)-2
        m=len(cut)-2
        while j>=0:
            if value>=cut[j]:
                j=-1
            else:
                j -=1
                m -= 1
        list.append(woe[m])
        i += 1
    return list
# define score function
def get_score(coe,woe,factor):
    scores=[]
    for w in woe:
        score=round(coe*w*factor,0)
        scores.append(score)
    return scores

#score function
def compute_score(series,cut,score):
    list = []
    i = 0
    while i < len(series):
        value = series[i]
        j = len(cut) - 2
        m = len(cut) - 2
        while j >= 0:
            if value >= cut[j]:
                j = -1
            else:
                j -= 1
                m -= 1
        list.append(score[m])
        i += 1
    return list

if __name__ == '__main__':
    data = pd.read_csv('TestData.csv')
    pinf = float('inf')
    ninf = float('-inf')
    dfx1, ivx1,cutx1,woex1=mono_bin(data.SeriousDlqin2yrs,data.RevolvingUtilizationOfUnsecuredLines,n=10)
    dfx2, ivx2,cutx2,woex2=mono_bin(data.SeriousDlqin2yrs, data.age, n=10)
    dfx4, ivx4,cutx4,woex4 =mono_bin(data.SeriousDlqin2yrs, data.DebtRatio, n=20)
    dfx5, ivx5,cutx5,woex5 =mono_bin(data.SeriousDlqin2yrs, data.MonthlyIncome, n=10)
    # Discretization of continuous variables
    cutx3 = [ninf, 0, 1, 3, 5, pinf]
    cutx6 = [ninf, 1, 2, 3, 5, pinf]
    cutx7 = [ninf, 0, 1, 3, 5, pinf]
    cutx8 = [ninf, 0,1,2, 3, pinf]
    cutx9 = [ninf, 0, 1, 3, pinf]
    cutx10 = [ninf, 0, 1, 2, 3, 5, pinf]
    dfx3, ivx3,woex3 = self_bin(data.SeriousDlqin2yrs, data['NumberOfTime3059DaysPastDueNotWorse'], cutx3)
    dfx6, ivx6 ,woex6= self_bin(data.SeriousDlqin2yrs, data['NumberOfOpenCreditLinesAndLoans'], cutx6)
    dfx7, ivx7,woex7 = self_bin(data.SeriousDlqin2yrs, data['NumberOfTimes90DaysLate'], cutx7)
    dfx8, ivx8,woex8 = self_bin(data.SeriousDlqin2yrs, data['NumberRealEstateLoansOrLines'], cutx8)
    dfx9, ivx9,woex9 = self_bin(data.SeriousDlqin2yrs, data['NumberOfTime6089DaysPastDueNotWorse'], cutx9)
    dfx10, ivx10,woex10 = self_bin(data.SeriousDlqin2yrs, data['NumberOfDependents'], cutx10)
    ivlist=[ivx1,ivx2,ivx3,ivx4,ivx5,ivx6,ivx7,ivx8,ivx9,ivx10]
    index=['x1','x2','x3','x4','x5','x6','x7','x8','x9','x10']
    fig1 = plt.figure(1)
    ax1 = fig1.add_subplot(1, 1, 1)
```

```
    x = np.arange(len(index))+1
    ax1.bar(x, ivlist, width=0.4)
    ax1.set_xticks(x)
    ax1.set_xticklabels(index, rotation=0,
 fontsize=12)
    ax1.set_ylabel('IV(Information Value)'
, fontsize=14)
    for a, b in zip(x, ivlist):
        plt.text(a, b + 0.01, '%.4f' % b,
ha='center', va='bottom', fontsize=10)
    # woe
    data['RevolvingUtilizationOfUnsecuredL
ines'] = Series(replace_woe(data['Revolvin
gUtilizationOfUnsecuredLines'], cutx1, woe
x1))
    data['age'] = Series(replace_woe(data[
'age'], cutx2, woex2))
    data['NumberOfTime3059DaysPastDueNotWo
rse'] = Series(replace_woe(data['NumberOfT
ime3059DaysPastDueNotWorse'], cutx3, woex3
))
    data['DebtRatio'] = Series(replace_woe
(data['DebtRatio'], cutx4, woex4))
    data['MonthlyIncome'] = Series(replace
_woe(data['MonthlyIncome'], cutx5, woex5))
    data['NumberOfOpenCreditLinesAndLoans'
] = Series(replace_woe(data['NumberOfOpenC
reditLinesAndLoans'], cutx6, woex6))
    data['NumberOfTimes90DaysLate'] = Seri
es(replace_woe(data['NumberOfTimes90DaysLa
te'], cutx7, woex7))
    data['NumberRealEstateLoansOrLines'] =
 Series(replace_woe(data['NumberRealEstate
LoansOrLines'], cutx8, woex8))
    data['NumberOfTime6089DaysPastDueNotWo
rse'] = Series(replace_woe(data['NumberOfT
ime6089DaysPastDueNotWorse'], cutx9, woex9
))
    data['NumberOfDependents'] = Series(re
place_woe(data['NumberOfDependents'], cutx
10, woex10))
    data.to_csv('WoeData.csv', index=False
)
```

```
    test= pd.read_csv('TestData.csv')
    # woe
    test['RevolvingUtilizationOfUnsecuredL
ines'] = Series(replace_woe(test['Revolvin
gUtilizationOfUnsecuredLines'], cutx1, woe
x1))
    test['age'] = Series(replace_woe(test[
'age'], cutx2, woex2))
    test['NumberOfTime3059DaysPastDueNotWo
rse'] = Series(replace_woe(test['NumberOfT
ime3059DaysPastDueNotWorse'], cutx3, woex3
))
    test['DebtRatio'] = Series(replace_woe
(test['DebtRatio'], cutx4, woex4))
    test['MonthlyIncome'] = Series(replace
_woe(test['MonthlyIncome'], cutx5, woex5))
    test['NumberOfOpenCreditLinesAndLoans'
] = Series(replace_woe(test['NumberOfOpenC
reditLinesAndLoans'], cutx6, woex6))
    test['NumberOfTimes90DaysLate'] = Seri
es(replace_woe(test['NumberOfTimes90DaysLa
te'], cutx7, woex7))
    test['NumberRealEstateLoansOrLines'] =
 Series(replace_woe(test['NumberRealEstate
LoansOrLines'], cutx8, woex8))
    test['NumberOfTime6089DaysPastDueNotWo
rse'] = Series(replace_woe(test['NumberOfT
ime6089DaysPastDueNotWorse'], cutx9, woex9
))
    test['NumberOfDependents'] = Series(re
place_woe(test['NumberOfDependents'], cutx
10, woex10))
    test.to_csv('TestWoeData.csv', index=F
alse)

    # score calculating
    coe=[7.885545,0.811083,0.494714,1.5292
14,2.286435,0.612646,1.524416]
    p = 20 / math.log(2)
    q = 300 - 20 * math.log(20) / math.log
(2)
    baseScore = round(q + p * coe[0], 0)
    x1 = get_score(coe[1], woex1, p)
    x2 = get_score(coe[2], woex2, p)
```

```
    x3 = get_score(coe[3], woex3, p)
    x7 = get_score(coe[4], woex7, p)
    x8 = get_score(coe[5], woex8, p)
    x9 = get_score(coe[6], woex9, p)
    print(x1,x2, x3, x7, x8, x9)
    test1 = pd.read_csv('cs-test.csv')
    test1['BaseScore']=Series(np.zeros(len
(test1)))+baseScore
    test1['x1'] = Series(compute_score(tes
t1['RevolvingUtilizationOfUnsecuredLines']
, cutx1, x1))
    test1['x2'] = Series(compute_score(tes
t1['age'], cutx2, x2))
    test1['x3'] = Series(compute_score(tes
t1['NumberOfTime3059DaysPastDueNotWorse'],
 cutx3, x3))
    test1['x7'] = Series(compute_score(tes
t1['NumberOfTimes90DaysLate'], cutx7, x7))
    test1['x8'] = Series(compute_score(tes
t1['NumberRealEstateLoansOrLines'], cutx8,
 x8))
    test1['x9'] = Series(compute_score(tes
t1['NumberOfTime6089DaysPastDueNotWorse'],
 cutx9, x9))
    test1['Score'] = test1['x1'] + test1['
x2'] + test1['x3'] + test1['x7'] +test1['x
8']+test1['x9']  + baseScore
    test1.to_csv('cs-tset-
score.csv', index=False)
    plt.show()
```

```
#Logit Regression
import pandas as pd
import matplotlib.pyplot as plt #Import im
age library
import matplotlib
import seaborn as sns
import statsmodels.api as sm
from sklearn.metrics import roc_curve, auc

if __name__ == '__main__':
    matplotlib.rcParams['axes.unicode_minu
s'] = False
    data = pd.read_csv('WoeData.csv')
```

```
    Y=data['SeriousDlqin2yrs']
    X=data.drop(['SeriousDlqin2yrs','DebtR
atio','MonthlyIncome', 'NumberOfOpenCredit
LinesAndLoans','NumberOfDependents'],axis=
1)
    X1=sm.add_constant(X)
    logit=sm.Logit(Y,X1)
    result=logit.fit()
    print(result.params)
    print(result.summary())

    test = pd.read_csv('TestWoeData.csv')
    Y_test = test['SeriousDlqin2yrs']
    X_test = test.drop(['SeriousDlqin2yrs'
, 'DebtRatio', 'MonthlyIncome', 'NumberOfO
penCreditLinesAndLoans', 'NumberOfDependen
ts'], axis=1)
    X3 = sm.add_constant(X_test)
    resu = result.predict(X3)
    fpr, tpr, threshold = roc_curve(Y_test
, resu)
    rocauc = auc(fpr, tpr)
    plt.plot(fpr, tpr, 'b', label='AUC = %
0.2f' % rocauc)
    plt.legend(loc='lower right')
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.ylabel('TPR')
    plt.xlabel('FPR')
    plt.show()
```