参赛队员：陶重年

学校：南京外国语学校

省份：江苏省

指导教师：严青

论文题目：A Study on Constructing Credit Card Fraud Detection Models

Table of Contents

A Study on Constructing Credit Card Fraud Detection Models

Abstract

This paper intends to conduct an in-depth research on risk warnings of credit card frauds. Logistic regression classification forecasting model is first constructed with desensitized data set of credit card transactions. Kernel density estimation is utilized to determine model thresholds, with full consideration of fraud characteristics (time, amount) distribution. Good performance of logistic regression classification forecasting model on data extracted from principal component analysis proves that desensitized data can be effectively applied to data analysis projects, overcoming security concerns of data analysis. For the purpose of thorough analysis and comparison, AdaBoost ensemble learning model and AutoEncoder model are further devised based on two methods from traditional machine learning and deep learning, in order to achieve accurate identification of fraudulent transactions. Comparisons on predicative ability are made between undersampling and SMOTE oversampling approaches to select the optimal way of dealing with imbalanced data set.

Considering the serious imbalanced feature of data set which has only 0.17% of frauds, this paper enhances the forecasting capability of classification model using SMOTE algorithm to conduct oversampling on fraudulent transaction data set. The classification forecasting model is then divided into training set and validation set, with a ratio of 7:3, using k-fold cross validation. Grid search tuning parameters are utilized to optimize the model. Model evaluation results show that the recall rate of optimized logistic regression classification forecasting model reaches as high as 98.5%, indicating a strong predictive ability.

The paper then proposes a framework algorithm based on the AdaBoost ensemble learning model of single-layer decision tree classifier. After randomly selecting 70% as training set and 30% as test set from the processed data, 8 weak classifiers are set up for training data using single-layer decision tree algorithm, to perform adaptive parameter learning through iterative algorithm. Based on various defined indicators, with regard to the test data of undersampled data set, the recall rate is 93.6%, precision rate is 97.8%, F-score is 0.957, and error rate is 4.13%; with regard to the test data of oversampled data set, the recall rate is 95.9%, precision rate is 98.8%, F-score is 0.973, and error rate is 2.65%. Therefore, the results demonstrate that the proposed model has strong fraud detection capability. For ease of comparison, AutoEncoder model is built up based on Tensorflow and Keras open source top-level frame work. The model establishes 29 neurons as input layer, uses 4 hidden layers as encoder and decoder, iteratively updates its parameters using BP algorithm, and at the same time uses L1 paradigm regularization to help balancing its fitting ability and generalization ability. Results of the undersampled data set test data are: AUC (Area Under Curve) score is 0.9439, the accuracy of classification is 94.5%; results of the oversampled data set test data are: AUC score is 0.965, and the accuracy of the classification is 98.7%.

Key Words: Kernel Density, Logistic Regression, AdaBoost Ensemble Learning Model, AutoEncoder Model

I. Background and Problem Restatement

As a means of payment for non-cash transactions, credit cards have become the most convenient and most popular credit service with attractive features such as longer interest-free period, fast and simple procedure. However, the continuous development of the Internet and information technology has also made the risk of credit card business increasingly apparent. Compared with other credit methods, credit cards have higher risks with the characteristics of no mortgage, higher overdue interest rate, technology dependence, being vulnerable to the economic cycle and easy to cause vicious circle. At present, due to the leakage of customer information, credit card fraud has become a serious and growing problem. With the economic downturn, risks of credit card business continue to rise. *The Overall Performance of the Payment Systems in the Second Quarter of 2018* released by the People's Bank of China shows that the amount of credit cards with more than half-a-year overdue was 75.667 billion yuan, a year-on-year increase of 6.35%. Compared with the 7.302 billion yuan in the same period of 2010, there was a growth of nearly ten times over the past eight years [1]. Increasing number of credit card frauds and malicious overdrafts has caused the industry to attach great importance to credit card risk management and early warning mechanisms.

Data set of this paper comes from *https://www.kaggle.com/mlg-ulb/creditcardfraud*. For the total number of 284,807 credit card transactions, data are pre-processed after PCA transformation, with no clear meaning and are highly imbalanced. By sampling and analyzing these data, the paper attempts to find a high generalization ability model suitable for classification, and establish a credit card fraud risk identification and prediction model, to effectively prevent risks before credit card overdrafts and frauds happen, making it possible for early risk warning.

The highly imbalanced data set, if not processed, shall impact the model's learning ability. Some scholars use clustering methods to preprocess the training data, extract representative training examples, and reduce the noise and size, in order to improve the training accuracy of the model. Vijay Hanagandi et al. [2] established a model based on the combination of density clustering and radial basis function network to calculate risk scores of credit card frauds. The empirical research finds that the method of density clustering can effectively alleviate the quantity imbalance between frauds and non-frauds, and therefore improve the classification effect of the model. Philip K.Chan [3] used a random cutting method to cut the large number of non-fraud transaction data in the original data set into several subsets in appropriate proportion, and put the fraudulent data into the cut subsets, based on which multiple classifiers are built up.

However, this clustering method is to further classify based on categories, which increases the complexity of the model structure, as well as the time and effort of model training. Therefore, the current mainstream is to solve the problem of imbalanced data by using random undersampling or oversampling [4]. The principle of random oversampling is to increase the sample of "minority class", while the principle of random undersampling is to remove part of the "majority class" in the original data set to balance samples. But in fact, undersampling is likely to result in the loss of important information about the "majority class" due to the operation of removing data, which has a great negative impact on model training. Especially considering that fraud data are already pre-processed and incomplete, drawbacks of this processing method are more obvious. Though the random oversampling method is also not perfect, it is generally better than undersampling.

II. Sample Selection and Data Screening

2.1 Sample Analysis

According to the sample data set, data are divided into 31 fields, where Time is the time calculated from the first record, with the unit of second, and the total time is 172,792 seconds, about 2 days. Amount is the amount of credit card consumption. Class is whether the transaction is fraudulent or not. In the case of fraud, its value is 1 otherwise 0. Fields V1 to V28 are the 28 principal components extracted by the principal component analysis of the original data set. And there is no vacancy data. Through data analysis, the sample data set has the following characteristics:

1. Imbalanced. Among 284,807 transactions, only 492 are frauds, accounting for 0.172%.

2. Due to privacy protection, these 28 fields are not given detailed explanation. Modeling cannot be done with common business experience.

Based on the above characteristics, it is necessary to perform preprocessing and sample screening on the original data set, to lay a foundation for training a more scientific model. Suppose sample data of the fraudulent transaction are positive, and those of the legal transaction are negative. Then 284,807 original sample data are divided into 492 positive sample data and 284,315 negative sample data.

2.2 Sample Screen

2.2.1 Synthetic Minority Oversampling Technique, SMOTE

The basic idea of the SMOTE algorithm is to generate new samples based on minority class samples and add the new ones to the data set. First, for each sample x in the minority class, calculate its Euclidean distances to all samples in the minority class sample set to obtain its k -nearest neighbor; randomly select several samples in its k -nearest neighbor, and assume that the selected nearest neighbor is x_n; finally, for each randomly selected neighbor x_n, randomly select one point on its line with x as a new sample to add into the data set.

2.2.2 Particle Size Balance Analysis after Oversampling

Oversampling is performed using the SMOTE algorithm described above. 100% negative sample data was picked up as the basis for the expansion of the positive sample size. The result presents an increase of the data set from 284,807 to 568,630, with 284,315 positive sample data, accounting for 50% of the total after expansion, and 284,315 negative sample data, accounting for 50% of the total after expansion to achieve balance.

2.2.3 Divide Samples

From the oversampled and expanded sample data, 70% are randomly selected as the training set and the remaining 30% as the test set.

2.3 Select and Determine Indicators

The variables V1 to V28 in the data set are the 28 principal components after the principal component analysis of the original data set. Principal component analysis eliminates the correlation among features based on the principle of variance maximization. Using a new set of linearly independent and mutually orthogonal features to characterize the original data would retain the information of the original data to a large extent. However, it is not clear whether the selected indicators will contribute to or the extend of the contribution before the principal component rotation. It is impossible to judge the merits of the selected indicators. Whether the 28 principal components V1 to V28 extracted after the rotation can well distinguish the positive and negative samples is yet to be seen. Therefore, the indicators need to be optimized.

The optimization idea is to select indicators with stronger predictive ability from the original indicator system (28 principal components extracted by principal component analysis). The optimization of the indicator is the feature selection process. The commonly used method is Filter, i.e., each feature is scored according to the correlation between the feature and the dependent variable and set the threshold or the number of features to be chosen to select the feature. . Filter includes chi-square test, Pearson correlation coefficient, information value, statistics principle, etc. [5] It should be noted that after PCA, correlation among V1 to V28 indicators is eliminated, but the correlation between each indicator and the Class still remains. Therefore, the Pearson correlation coefficient method is chosen to select indicators.

The correlation coefficient is calculated as

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y} \quad (2\text{-}1)$$

According to this formula, the correlation between each indicator and Class is calculated, and the result is

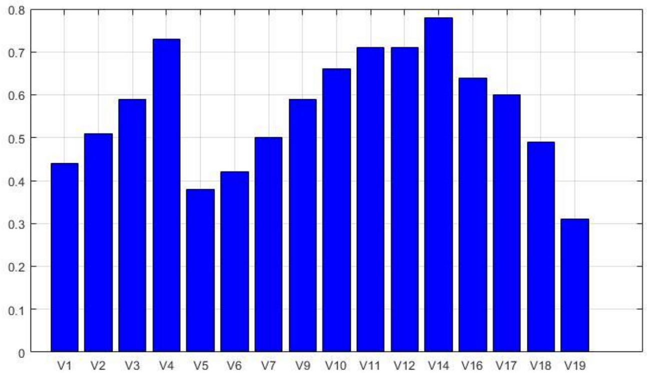
[Table 2-1] Correlation Coefficient between Indicators and Class

Indicator	V1	V2	V3	V4	V5	V6	V7
Correlation Coefficient	-0.44	0.51	-0.59	0.73	-0.38	-0.42	-0.5
Indicator	V8	V9	V10	V11	V12	V13	V14
Correlation Coefficient	0.083	-0.59	-0.66	0.71	-0.71	-0.056	-0.78
Indicator	V15	V16	V17	V18	V19	V20	V21
Correlation Coefficient	-0.017	-0.64	-0.6	-0.49	0.31	0.19	0.14
Indicator	V22	V23	V24	V25	V26	V27	V28
Correlation Coefficient	0.038	-0.024	-0.11	-0.0044	0.082	0.14	0.087

[Table 2-2] Self-defined Pearson Correlation Coefficient Threshold

Absolute value of correlation coefficient	**Correlation**
0-0.09	No
0.1-0.3	Weak
0.3-0.5	Middle
0.5-1.0	Strong

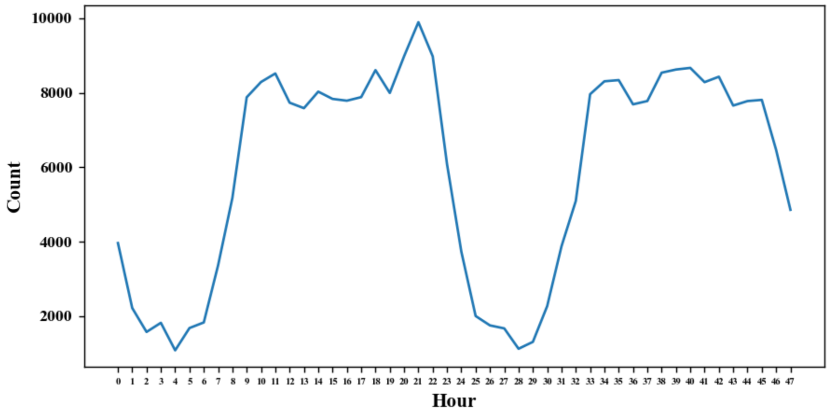
The stronger the correlation between each indicator and Class, the more capable the indicator is to distinguish between positive and negative samples. 0.3 is defined as the threshold. When the absolute value of the correlation coefficient between the indicator and the Class is less than 0.3, the indicator is deleted. With this screening criteria, the indicator system obtained consists of indicators V1, V2, V3, V4, V5, V6 , V7, V9, V10, V11, V12, V13, V14, V16, V17, V18, and V19. Result is shown in Figure 2-1.



[Figure 2-1] Chosen Indicators

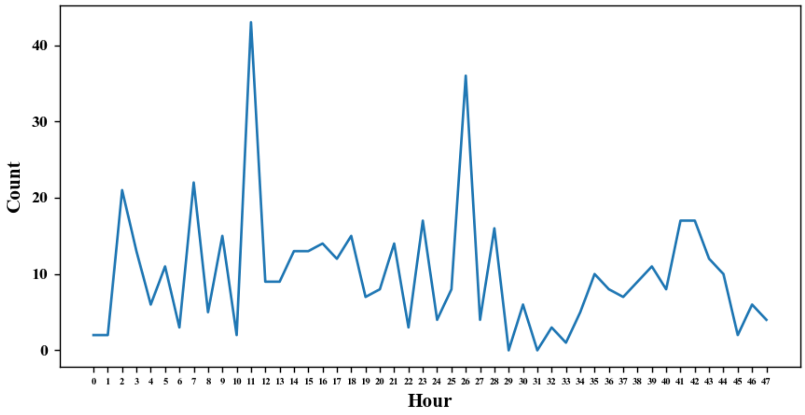
2.4 Analysis of Transaction Time

Since the unit of the Time is seconds, the data are too scattered to have statistical significance. Thus, the unit is converted to hour and statistics for each hour after the first transaction are gathered. The statistical result is shown in Figure 2-2. As can be seen from the figure, in 48 hours after the first transaction, the credit card transaction volume presents obvious periodicity, with about 24 hours as one cycle; and each cycle shows obvious consumption peak and valley, which may be due to the fact that the credit card users rest at night.

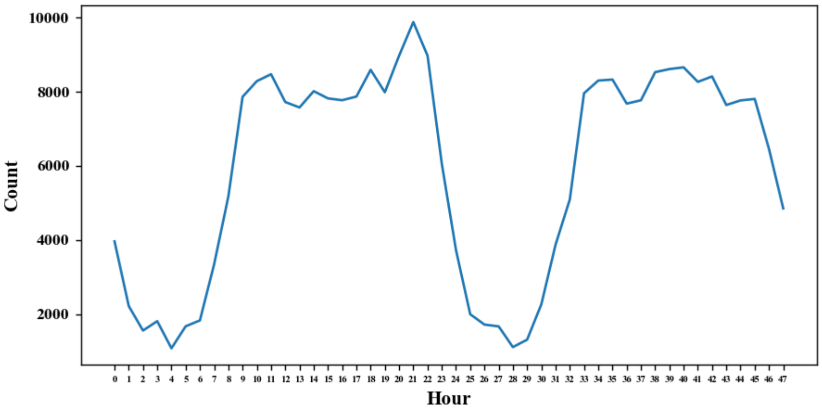


[Figure 2-2] Statistics on Hourly Transactions

After the first transaction record, statistical analysis of the hourly transaction volume of the fraudulent transaction samples and that of the legal transaction samples were conducted respectively. Results are shown in Figure 2-3 and Figure 2-4. As can be seen from the two figures, compared with the legal transactions, fraudulent transactions have a higher volume during the low-traffic period of the legal transactions, indicating that the credit card thieves are more inclined to commit crimes when card owners are at rest and when frequency of consumption is low, in order not to attract the attention of card owners.

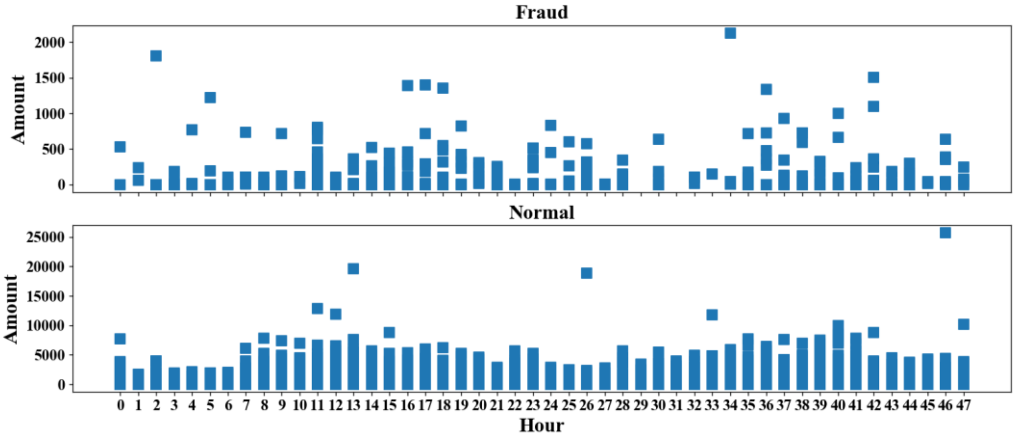


[Figure 2-3] Statistics on Fraudulent Transactions by Hour



[Figure 2-4] Statistics on Legal Transactions by Hour

After the first transaction record, statistical analysis of the consumption amount of the fraudulent transaction samples and that of the legal transaction samples were also conducted respectively. Result is shown in Figure 2-5. As can be seen from the figure, compared with the consumption amount of the legal transactions, the credit card fraudulent transactions have a higher frequency of outliers during the consumption valley period. This again indicates that fraudulent transactions are easier to occur during periods of low consumption.



[Figure 2-5] Distributions of Consumption Amount by Hour

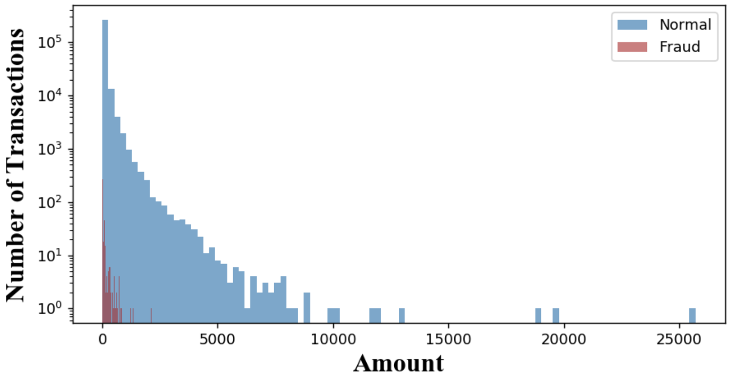
From the above analysis, it can be inferred that fraudulent transactions are more likely to occur during the consumption valley periods. The variable Hour plays a significant role on judging whether the record is fraudulent or not. It should be used as an input feature of the model. The original variable Second can be abandoned.

2.5 Analysis of Transaction Amount

Statistical analysis of characteristics of the Amount column of the fraudulent transaction samples and that of the legal transaction samples were conducted respectively. Result is shown in Table 2-3, and its consumption amount distribution is shown in Figure 2-6. From Table 2-3 and Figure 2-6, it can be seen that compared with the transaction amount of legal samples, the amount of frauds tend to be scattered and small with 2,125.9 as the maximum number, which indicates that the credit card thieves are more inclined to choose small transaction amounts in order not to attract the attention of card owners.

[Table 2-3] Statistics on Transaction Amount

	Fraudulent	Normal
Occurence	492	284,807
Average	124.1	88.3
Standard deviation	258.7	250.1
Minimum	0.0	0.0
25%	1.0	5.7
50%	9.8	22.0
75%	106.3	77.1
Maximum	2,125.9	25,691.2

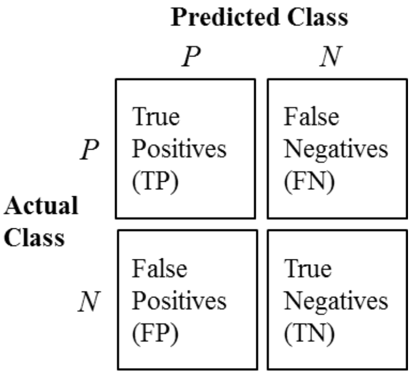


[Figure 2-6 Distribution of Consumption Amount

III. Evaluation Indicators

3.1 Confusion Matrix

There are many indicators for evaluating the effectiveness of the credit card fraud detection model, but most are derived from the confusion matrix[6], as shown in Figure 3-1.



		Predicted Class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

[Figure 3-1] Confusion Matrix

Meanings of different positions in the confusion matrix are explained below.
True Positives (TP): Number of fraudulent consumption behaviors with accurate classification
False Positives (FP): Number of fraudulent consumption behaviors with wrong classification
True Negatives (TN): Number of legal consumption behaviors with accurate classification
False Negatives (TN): Number of legal consumption behaviors with wrong classification

Basic performance measures derived from the confusion matrix are as follows [7].

(1) Precision Rate: $Precision = \frac{TP}{TP+FP}$

(2) Recall Rate: $Recall = TPR = \frac{TP}{TP+FN}$

(3) False Positive Rate: $FPR = \frac{FP}{FP+TN}$

(4) F-score: $F_{score} = \frac{(1+\beta^2)\times Recall\times Precision}{\beta^2\times Reall+Precision}$ (β shows the importance of Precision relative to Recall. Usually $\beta = 1$.)

(5) Error Rate: $ERR = \frac{FP+FN}{TP+TN+FN+FP} = \frac{FP+FN}{P+N}$

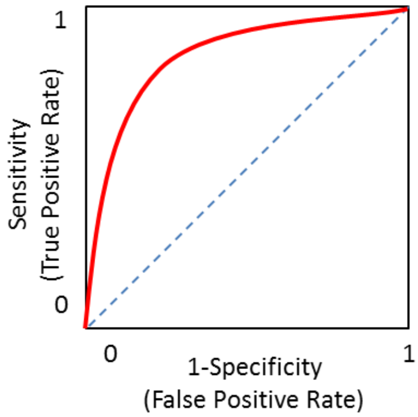
If a legal transaction is misjudged as a fraud, the bank will lose the profit of this transaction; while if a fraudulent transaction is misjudged as legal, the bank will suffer loss. Therefore, Error Rate is also a major indicator.

（6） Accuracy: $ACC = \frac{TP+TN}{TP+TN+FN+FP} = \frac{TP+TN}{P+N}$

3.2 Receiver Operating Characteristic Curve

The ROC curve offers an effective tool for comprehensive and accurate evaluation of classification models by means of graphic illustration. It is generated by dividing the results of the training model into several critical points, plotting the sensitivity (TPR) corresponding to each critical point on the Y-axis, and 1-specificity (FPR) on the X-axis, and then connecting all these critical points together. The ROC curve can also determine the optimal threshold for detection. The cut-off point closest to the (0,1) on the curve is the best critical point. This method can make the model's precision rate and recall rate higher, and the false positive rate lower [8].

In other words, the threshold will affect the effectiveness of the classification rules themselves. For example, in order to improve the hit rate of prediction, if the credit card fraud behavior has a time characteristic, the bank can appropriately shrink the classification rule by reasonably adjusting the threshold value.



[Figure 3-2] ROC Curve

3.3 Area Under the ROC Curve

AUC (Area Under the ROC Curve) score is another indicator to measure the performance of classification model.

IV. Model I：Logistic Regression Model

4.1 Build Logistic Regression Model

Logistic regression is a very widely used statistical technique for identifying fraud behavior models in binary classification problems. It is a multi-class variable analytical method that studies and observes the relationship between the result y and some influencing variables $(x_1, x_2, \ldots x_n)$. Despite being a simple structure, it has a series of advantages such as small amount of calculation, fast speed and low storage resources when dealing with complex data systems composed of multiple indicators [9].

4.1.1 Build Logistic Model

The dependent variable class is an integer variable of 0-1, so the binomial logistic regression model is chosen.

$$\mathrm{Y} = \begin{cases} 1\ Fraudulent \\ 0\ Legal \end{cases} \tag{4-1}$$

Assume that the independent variable x and the model parameter θ are defined as vectors of $\mathrm{n}+1$:

$$X^T = [1, x_1 x_2 \ldots x_n] \tag{4-2}$$

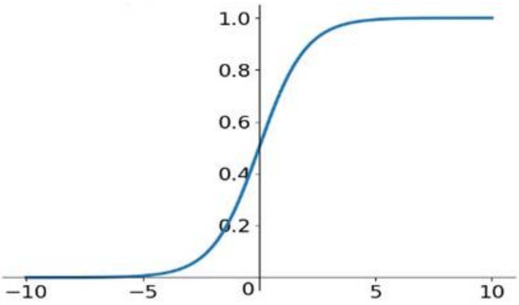
$$\theta^T = [1, \theta_1 \theta_2 \ldots \theta_n] \tag{4-3}$$

The probability of fraudulent behavior p in the logistic regression model can be obtained.

$$z = \theta + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T X \tag{4-4}$$

$$p = h_\theta(x) = \frac{1}{1+e^{-z}} \tag{4-5}$$

Where z is actually a linear regression formula, it also reflects that the core concept of logistic regression is derived from generalized linear regression model. This function is also called the sigmoid function.



[Figure 4-1] Sigmoid Function Curve

As can be seen from Figure 4-1, the output of sigmoid function is between (0, 1), indicating that the data belongs to the probability of a certain category, and therefore the probability of fraudulent behavior and legal behavior is

$$P(Y=1|x;\theta)=h_\theta(x) \tag{4-6}$$

$$P(Y=0|x;\theta)=1-h_\theta(x) \tag{4-7}$$

On this basis, the ratio of the probability of credit card fraudulent consumption to the probability of legal consumption, also known as the odd of experiencing an event, is

$$R_{odds}=\frac{P(Y=1|x;\theta)=h_\theta(x)}{P(Y=0|x;\theta)=1-h_\theta(x)}=e^{\theta+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n} \tag{4-8}$$

$$\ln(R_{odds})=\theta+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n=z \tag{4-9}$$

Equation 4-9 is the logit function. It can be seen that the logistic regression model is a linear regression model with the logit function as its dependent variable.

4.1.2 Maximum Likelihood Estimation

Parameters need to be estimated based on the existing objective function. The maximum likelihood estimation method is adopted here. Its calculation process is as follows:

Suppose there are m observation samples with values of $y_1, y_2, y_3, \ldots y_m$ respectively. According to equation 4-6, the probability of one of the observation values is

$$P(y_i)=p_i^y\times(1-p_i)^{1-y} \tag{4-10}$$

Due to the independence among the observation samples, the product of each edge distribution is used to construct the joint distribution. The likelihood function is obtained as:

$$L(\theta)=\prod_{i=1}^{m}p(x_i)^{y_i}(1-p(x_i))^{1-y_i} \tag{4-11}$$

The goal is to find the parameter estimate that maximizes the likelihood function, that is, to find the parameters $\theta_0, \theta_1, \theta_2 \ldots \theta_n$, such that L(w) takes the maximum value. Therefore, Log-likelihood function

$$\ln L(\theta)=\sum_{i=1}^{m}(y_i\ln\big(p(x_i)\big)+(1-y_i)\ln(1-p(x_i))) \tag{4-12}$$

To solve the maximum value, the partial derivative of the objective function is set to be 0.

$$\frac{\partial \text{In}L(\theta_k)}{\partial\theta_k}=\sum_{i=1}^{m}x_{ik}\,[y_i-p(x_i)]=0 \tag{4-13}$$

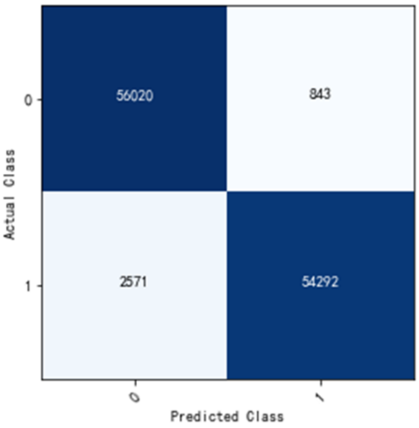
There are $n+1$ partial derivative equations. The $n+1$ nonlinear functions of the $n+1$ model parameters are solved using the Newton-Raphson(N-R) method. Then

$$\theta^{i+1} = \theta^{i} - \hat{E}^{-1} g \tag{4-14}$$

θ^{i} and θ^{i+1} are iterative i and $i+1$ estimated parameter vectors, $\hat{E}^{-1}$is the second derivative information matrix, and g is the first derivative vector.

4.2 Model Solution

4.2.1 Confusion Matrix



[Figure 4-2] Confusion Matrix

Use k-fold cross-validation, and at the same time conduct training and validation on randomly-generated sub-samples repeatedly. Based on the indicators defined in Chapter 3.1, the following evaluation values can be obtained for logistic regression model.

[Table 4-1] Result Analysis

Precision	Recall	F_score	Accuracy	Error
95.6%	98.5%	0.970	97.0%	3.00%

It can be concluded from Accuracy that the overall accuracy of the model is high, although this is related to the fact that most of the samples are predicted to be legal. However, as seen from the Recall of 0.985, the model achieves fairly good accuracy for fraud detection on this slanted data set.

4.2.2 Logistic Model based on Kernel Density Estimation

(1) Credit card fraud under the Bayesian view

Suppose there are factors A and B that affect the model prediction result Y, and the factors A and B are independent of each other, then according to the Bayesian principle, the following equation is obtained.

$$P(Y|AB) = \frac{P(B|AY)P(Y|A)}{P(B|A)} = \frac{P(B|Y)P(Y|A)}{P(B)}$$

The prior probability distributions of $P(Y)$、$P(A)$ and $P(B)$ can be seen from a large amount of data.

$$P(Y|AB) \propto P(Y|A)P(B|Y) \tag{4-15}$$

In this paper, factor A consists of the characteristics of each principal component, and factor B includes time and consumption amount. These two factors are relatively independent of each other. Since the probability of credit card frauds does not change continuously with respect to time and consumption amount, it is of little significance to include it in the logistic model. Therefore, this paper mainly uses the principal component feature (corresponding to factor A) to train the logistic model, and the model after training can calculate the risk of credit card frauds $P(Y|A)$. On this basis, it is still necessary to consider the role of time and consumption amount, that is, consider the probability $P(B|Y)$. Since this paper is more concerned about the risk of credit card frauds, the only thing needs to get is the distribution $P(B|Y = 1)$.

Kernel density estimation is a technique to estimate the unknown probability distribution of a random variable, based on a sample of points taken from that distribution.[10] It can be used for the estimation of the distribution $P(B|Y = 1)$. However, since the joint distributions of time T and amount AM in the fraud samples are not continuous, the sample space is discretized, and the probability that the fraud samples appear in the discrete sample space $P(T, AM\ /Y = 1)$ is calculated.

Based on $P(Y = 1|A)$ predicted by the logistic model, joint distributions of the time and the amount of consumption under the fraud samples are utilized to correct $f(T, AM) = P(T, AM|Y = 1)$. Finally, the fraud risk $P(Y = 1|A, T, AM)$ under various factors can be estimated.

(2) Credit card fraud alert - threshold determination

A fixed fraud warning threshold α_o is set. When $P(Y = 1|A = a, T = t, AM = am) \geqslant \alpha_o$, the sample $(A = a, T = t, AM = am)$ is considered to be a fraud sample. In practice, the logistic model obtained by training is used to complete the classification, so there is another threshold α. When $P(Y = 1|A) \geqslant \alpha$, the sample can be determined to be a fraud.

$$\alpha_o \propto f(T, AM)\alpha$$

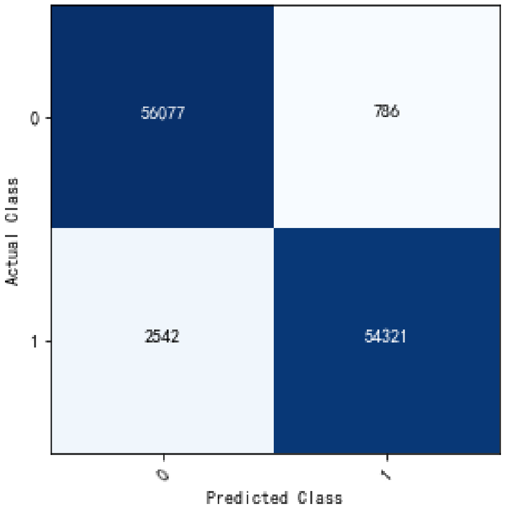
i.e.

$$\alpha \propto [f(T, AM)]^{-1} \tag{4-16}$$

α represents the bank's pursuit of recall rate, or the pressure on credit card frauds, while the probability value $f(T, AM)$ represents the likelihood of fraud samples in the corresponding time period and amount. These two have inverse relationship. That is to say, in the high incidence period of frauds, the bank should increase the intensity of the attack, i.e., lower the identification threshold; while in the low incidence period, the threshold α could be set higher. Therefore, the threshold α of the logistic model should be the proportional function of $[f(T,AM)]^{-1}$.

4.2.3 Empirical Study on Kernel Density Estimation & Logistic Model

Using kernel density estimation, the threshold is determined by Time under the condition that the consumption amount is constant. The confusion matrix of the model is shown in Figure 4-3, and the final F-score is 0.971.



[Figure 4-3] Confusion Matrix of Kernel Density Estimation + Logistic Model

4.3 Model Evaluation

4.3.1 Merits of Linear Logistic Regression Model

The linear logistic regression model is relatively successful in the application of credit card scoring. There are three main advantages.

(1) The independent variable can be a continuous variable or a discrete variable, and the value is not strictly limited.

(2) The data are not required to satisfy the assumptions that they follow normal distribution and the covariances are the same, which expands the scope of application.

(3) The value of the dependent variable of the regression model is binary. The model intuitively indicates whether an event can occur, the probability of occurrence, the influencing factors, and the weight of each influencing factor.

4.3.2 Improvement on Linear Logistic Regression Model

(1) Different strategies can be adopted for threshold selection (strict, loose, multi-standard).

(2) The model can be further tested by rejecting deduction.

V. Model II: AdaBoost Ensemble Learning Model

5.1 Model II: AdaBoost Ensemble Learning Model Based on Single-Layer Decision Tree

After the previous data processing, this paper uses the single-layer decision tree to classify fraudulent transaction data and legal transaction data. The single-layer decision tree model makes decisions based on a single feature. Steps of algorithm are as follows.

(1) The features are divided according to the horizontal and vertical coordinates. The horizontal coordinate is regarded as feature one, and the vertical coordinate as feature two, and one of them is selected to make the decision.

(2) Based on the first step, since the outer loop is the loop of the data set feature, classifications are made according to the first indicator, then there is a node in the first feature point, i.e., the left and right branches of the tree. The judgment is made using following steps.

Step 1: According to the data value, set a threshold T, in which T is equal to the minimum eigenvalue (the minimum value of the first coordinate) $+ (1,2,3,4,5,\dots) \times$ step size, and the threshold changes cyclically; the one greater than threshold T is the "right node", otherwise it is the "left node".

Step 2: Determine the error rate. Construct an all-1 column vector e. If the prediction result is the same as the label, modify the value corresponding to initialization to 0, and finally use a weight vector $\mathrm{D.T} \times \mathrm{e}$, which is the final error rate. If the error rate is less than a certain threshold, it is the most efficient decision tree.

Step 3: Compare the error rates of the "right node" and the "left node", and meanwhile judge the error rate of the first feature in the large loop, or the error rate of the second feature.

Through calculation, accuracy rate of the simple classifier is found to be too low. In order to improve it, optimization is made by AdaBoost algorithm.

AdaBoost is an iterative algorithm with a simple idea of training different classifiers (weak classifiers) for the same training set, and then weighting these classifiers together to form a stronger final classifier (strong classifier) [11].

Indicators obtained after data processing is used as a training set, where the value of each field is a training sample, and the field is used as a test set. Steps are as follows.

(1) Give a training data set: $(x_1, y_1), \dots (x_n, y_n)$, where $y_1 \in \{-1,1\}$ is used to represent the classification label of the training sample, $i = 1, \dots, \mathrm{n}$.

(2) First, initialize weight distribution of the training data. The same value is assigned to the initial weight: $w_i = \frac{1}{n}$. Train the initial weight distribution $D_1(i)$ of the sample set this way.

$$\mathrm{D}_1(i) = (w_1, w_2, \dots w_n) = \left(\frac{1}{n}, \frac{1}{n}, \dots \frac{1}{n}\right) \tag{5-1}$$

(3) Iterate $t = 1, \dots, T$

a) If the weak classifier h has the lowest error rate, it is selected as the t-th basic classifier H_t. And calculate the weak classifier $H_t: X \to \{-1,1\}$, the error of the weak classifier on the distribution D_t is

$$e_t = P(H_t(x_i) \neq y_i) = \sum_{i=1}^{n} w_{ti} I(H_t(x_i) \neq y_i) \tag{5-2}$$

b) The weak classifier weight is represented by α, then the weight can be calculated as follows.

$$\alpha \mathrm{t} = \frac{1}{2} \ln\left(\frac{1 - e_t}{e_t}\right)$$

c) Update the weight distribution of the training samples $D_t + 1$:

$$D_{t+1} = \frac{D_t(i)\exp(-\alpha t y_i H_i(x_i))}{z_i} \tag{5-3}$$

where z_i is normalized

$$z_i = 2\sqrt{e_t(1 - e_t)}$$

d) Finally, the weak classifiers are combined by the weak classifier weight αt, ie:

$$f(x) = \sum_{t=1}^{T} \alpha t H_t(x) \tag{5-4}$$

A strong classifier is obtained using the sign function:

$$H_{final} = sign(f(x) = sign(\sum_{t=1}^{T} \alpha t H_t(x)) \tag{5-5}$$

5.2 Model Solution

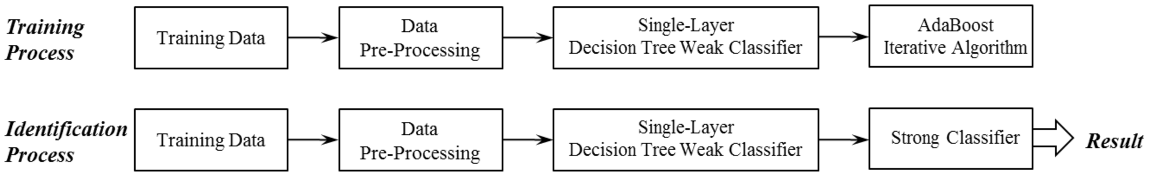
The process of identifying whether there is consumer fraud in the record is mainly divided into the following 4 steps.

(1) Preprocess data. Preprocess all the original field contents to prepare for the next step.

(2) Generate weak classifiers. Generate eight weak classifiers from the eight indicators obtained after data preprocessing through the single-layer decision tree weak classifier.

(3) Generate a strong classifier. Generate a strong classifier through the AdaBoost iterative algorithm.

(4) Identification: After pre-processing the consumption records that need to be identified, the trained strong classifier is used to determine whether there are frauds. The model uses the following process to determine whether there are frauds.



[Figure 5-1] Algorithm Flowchart

5.3 Data Preprocessing

The oversampled data set *data_PCA_Oversampling* and the undersampled data set *data _ PCA _ Undersampling* obtained in Chapter 2.2 are used. Among them, the 0,1 ratio of Class in both data sets is 1. Select 70% as the training set, and the rest as the test set.

5.4 AdaBoost Iteration on Weak Classifiers

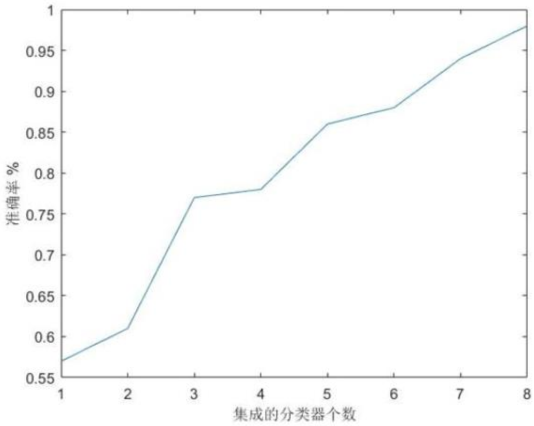
Next, using the established AdaBoost iterative algorithm model, the given transaction information is classified through the strong classifier obtained from weighted combination of weak classifiers. Final result is analyzed as follows.

(1) Test analysis of oversampling results:

[Table 5-1] Test Analysis of Oversampling Results

Confusion Matrix		**Predicted**		**Total**
		1	**0**	
Actual	1	81,804	3,500	85,304
	0	1,025	84,252	85,277
Total		82,829	87,752	170,581

During the oversampling process, there are 81,804 transactions which are predicted fraudulent and are actually frauds, 3,500 transactions which are predicted legal but are actually fraudulent, 1,025 transactions which are predicted fraudulent but are actually legal, and 84,252 transactions which are predicted legal and are actually legal. Result is shown in Table 5-1.



[Figure 5-2]　Relationship between Number of Oversampling Integrated Classifiers & Accuracy

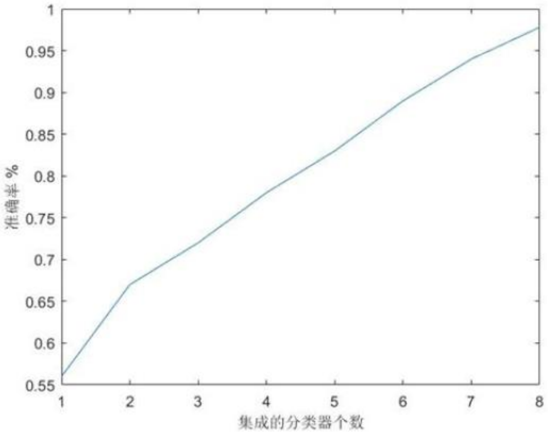
From Figure 5-2, when the number of integrated classifiers (i.e., the number of weak classifiers) is one, the accuracy of prediction is in the range of 55% to 60%; when the number of integrated classifiers is increased, although the changing rate of Accuracy varies, the overall rate is on the rise and eventually closes to 100%.

(2) Test analysis of undersampling results:

[Table 5-2]　Test Analysis of Undersampling Results

Confusion Matrix		**Predicted**		**Total**
		1	**0**	
Actual	1	132	9	141
	0	3	146	149
Total		135	155	290

During the undersampling process, there are 132 transactions which are predicted fraudulent and are actually frauds, 9 transactions which are predicted legal but are actually fraudulent, 3 transactions which are predicted fraudulent but are actually legal, and 146 transactions which are predicted legal and are actually legal. Result is shown in Table 5-2.



[Figure 5-3] Relationship between Number of Undersampling Integrated Classifiers & Accuracy

From Figure 5-3, when the number of integrated classifiers (i.e., the number of weak classifiers) is one, the accuracy of prediction is in the range of 55% to 60%; when the number of integrated classifiers is increased, Accuracy rate grows steadily and eventually closes to 100%.

(3) Comprehensive analysis of two data sets

Through the two methods of over-sampling and under-sampling, specific values of evaluation indicators are calculated according to the definitions. It can be concluded from the following table that the error rate calculated by the AdaBoost algorithm model is less than 5%, the recall rate is close to 95%, the precision rate is close to 98%, and the F-score is over 0.95, as shown in Table 5-3.

[Table 5-3] Evaluation Indicator Values of Oversampling & Undersampling

Evaluation Indicator	Oversampling	Undersampling
Recall Rate	95.9%	93.6%
Precision Rate	98.8%	97.8%
F_score	0.973	0.957
Error Rate	2.65%	4.14%
Accuracy Rate	97.3%	95.9%

Through the above analysis, the accuracy of the AdaBoost algorithm model is significantly higher than that of the original simple classifier. Therefore, the model has strong fraud detection ability, and has a high reference value.

5.5 Improvement and Promotion

Using a single-level decision tree with simple binary classifier as a weak classifier does not necessarily distinguish a small number of fraudulent customers. For this reason, it is necessary to find a weak classifier with better detection ability. For example, use SVM classifier or discriminant analysis classifier to improve the capability of fraud detection. The approach of constructing weak classifiers and iterating them into a strong classifier can be extended to fields such as NLP processing, image processing or recommendation systems. For example, achieve accurate detection by obtaining various feature indicators after data processing and selecting parameters adaptively through AdaBoost iteration.

VI. Model III: AutoEncoder Deep Learning Model based on Tensorflow

AutoEncoder is a multi-layer forward neural network. It is an unsupervised learning and has important applications in data dimensionality reduction and feature extraction. AutoEncoder can be used to initialize the weight matrix before the start of deep learning training.

6.1 Model Solution

In the unsupervised learning algorithm, usually use an input vector x for encoding, select "tanh" as the calculation method for activating function as shown in equation (6-1), and then decode to obtain a coding result y, which produces reconstruction vector z by the decoder, as shown in equation (6-2). The process can be regarded as compression encoding of the input data, representing the high-dimensional original data with a low-dimensional vector. This way the compressed low-dimensional vector can retain the typical characteristics of the input data, so that the original data can be restored more conveniently [12].

$$\tanh x = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{6-1}$$

$$\mathrm{z} = g_{\theta}(y) = s(W_{y} + b) \tag{6-2}$$

The advantage of AutoEncoder is that its learning process is completely unsupervised. The loss function adopted in this paper is shown in equation (6-3)

$$\mathrm{L(X,Z)} = \sum_{i=1}^{n} KL(x_i \parallel z_i) \tag{6-3}$$

x represents a matrix of n sample vectors, and $KL(x_i \parallel z_i)$ represents the divergence between x_i and z_i. Since the dimension of y is much smaller than that of x, y can learn not only low-dimensional information but also high-dimensional information. Using the stochastic gradient descent algorithm as the weight training algorithm, the error of the output result can be minimized. The weight matrix is updated by the following formula (6-4), where η is the updated step size.

$$W \leftarrow W - \eta \frac{\partial L(X,Z)}{\partial W} \tag{6-4}$$

The nonlinear dimension reduction approach is to keep certain local structure or information of the original high-dimensional data unchanged and directly find a low-dimensional matrix y to replace the high-dimensional matrix x. In order to prevent the over-fitting phenomenon, the learning of the model is constrained by adopting the variable selection method. The data is encoded sparsely to ensure the sparsity of each code in the algorithm. Formula (6-4) is specifically adjusted to equations (6-5) and (6-6) for calculation [13].

$$\mathrm{L(x,z)} = \mathrm{KL(x \parallel z)} + \mathrm{Lasso}(\theta) \tag{6-5}$$

$$\mathrm{Lasso}(\theta) = \lambda \sum_{j=0}^{|\theta|} |\theta_j| \tag{6-6}$$

6.2 Data Preprocessing

Since AutoEncoder itself comes with dimensionality reduction, there is no need to use PCA processed data sets. The oversampled data set *data_Oversampling* and the undersampled data set *data _ Undersampling* for PCA in Chapter 2.2 are used. Among them, the 0,1 ratio of Class in both data sets is 1. Ratio of training set and test set is still 7:3.

6.3 Build AutoEncoder Model

This paper builds the AutoEncoder model based on the open source top-level framework of Tensorflow and Keras. Set 29 neurons as input layers, and establish two coding layers and two decoding layers, a total of four fully connected Dense layers, which are 28, 10, 10, 29 neurons respectively. L1 normalization will be used during training. Each layer uses the "tanh" function as the activation function. During training, set 150 epochs and the batch bitch size as 32 samples.

6.4 Load Training Model to Identify Test Data

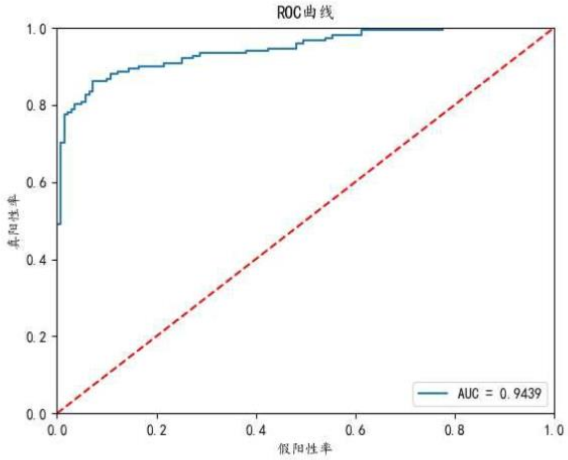
According to the trained model, count the prediction data and test data in undersampling and oversampling data sets, and calculate specific values of various indicators with correct classification. The results are shown in Table 6-1.

[Table 6-1] Values of Undersampling and Oversampling Indicators

Data Set	**Undersampling**		**Oversampling**	
	data_Undersampling		***data_Oversampling***	
Category	**Prediction**	**Test**	**Prediction**	**Test**
count	290	290	170581	170581
mean	14.23425	0.52069	9.73891	0.49824
std	31.66916	0.50044	19.08866	0.49999
min	0.16633	0	0.07328	0
25%	0.49985	0	0.39873	0
50%	1.04733	1	1.10841	0
75%	9.27170	1	9.11526	1
max	266.32854	1	301.32378	1

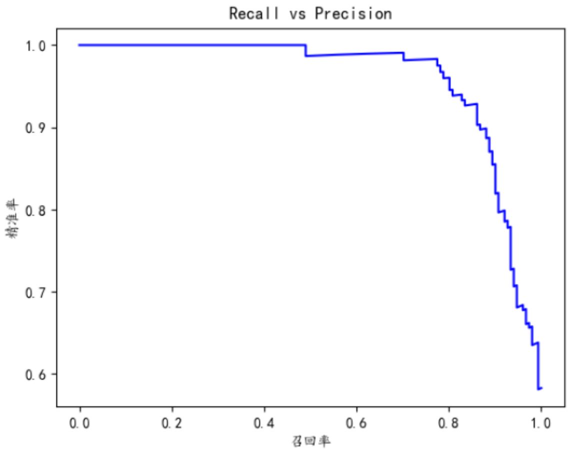
At the same time, the identification results are analyzed according to indicators defined previously. The ROC curve evaluation method is used to assess the classification ability and precision of the model.

(1) Undersampling test data



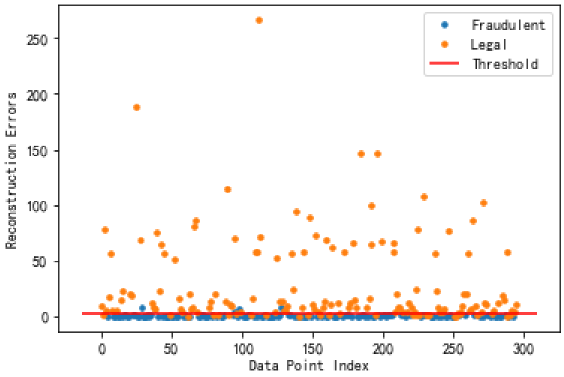
[Figure 6-1] Undersampling ROC Curve on Cumulative Risk Assessment

The closer the ROC curve is to the upper left corner, the stronger the model’s classification ability. The overall diagnostic accuracy can be assessed by the area under the curve. According to the above Figure 6-1, by undersampling test, the ROC curve of the training model achieves the expected effect, indicating that the model is effective to detect fraud transactions with the AUC score of 0.9439.



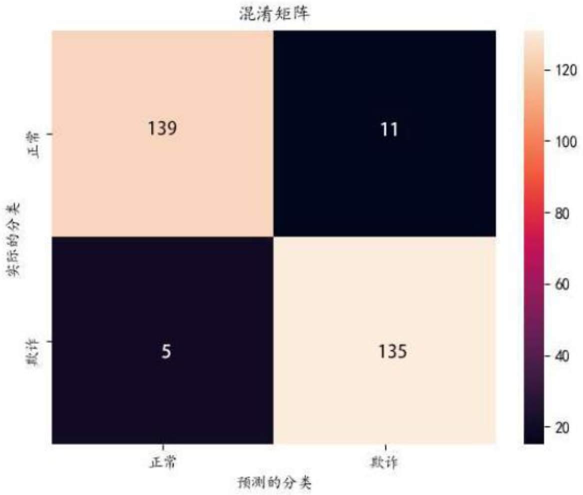
[Figure 6-2] Comparison of Recall Rate and Precision Rate

Comparing the Precision rate and the Recall rate, it can be concluded from Figure 6-2 that the Recall rate is negatively correlated with the Precision rate, that is, the higher the Recall rate, the lower the Precision rate, and vice versa. If the error is greater than the predetermined threshold, mark it as fraud, set *threshold=3.5*. The prediction is as follows.



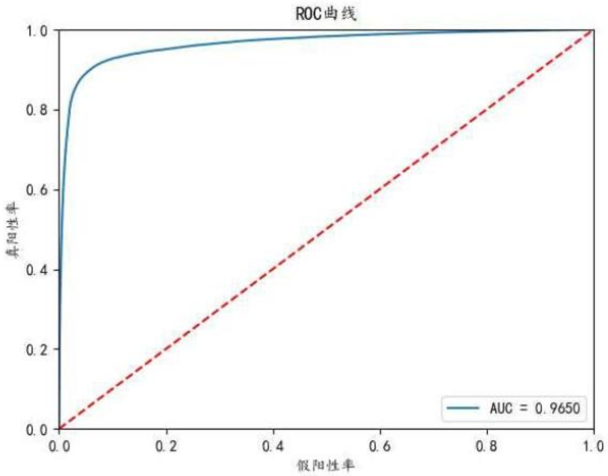
[Figure 6-3] Different Types of Reconstruction Errors for Undersampling

The following confusion matrix is constructed for different types of reconstruction errors, in which 135 are predicted to be fraudulent and are actually fraudulent, and 139 are predicted to be legal and are actually legal. It can be concluded from the error classification matrix that accuracy rate of classification is 94.482%.



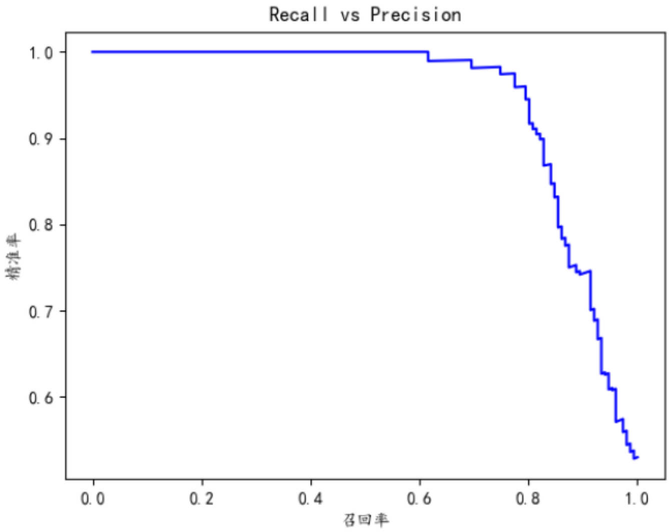
[Figure 6-4] Confusion Matrix of Different Types of Reconstruction Errors for Undersampling

(2) Oversampling test data



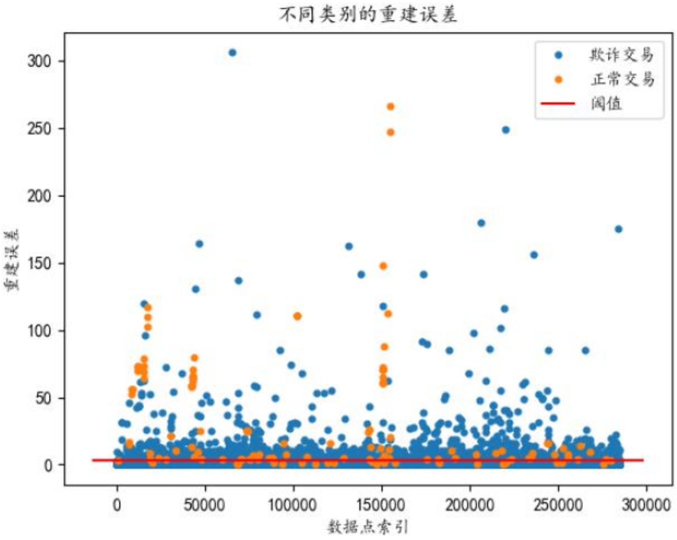
[Figure 6-5] Oversampling ROC Curve of Cumulative Risk Assessment

Through the oversampling test, according to the above figure 6-5, the ROC curve is close to the upper left corner, indicating that the model is effective to detect fraud transactions with the AUC score of 0.965.



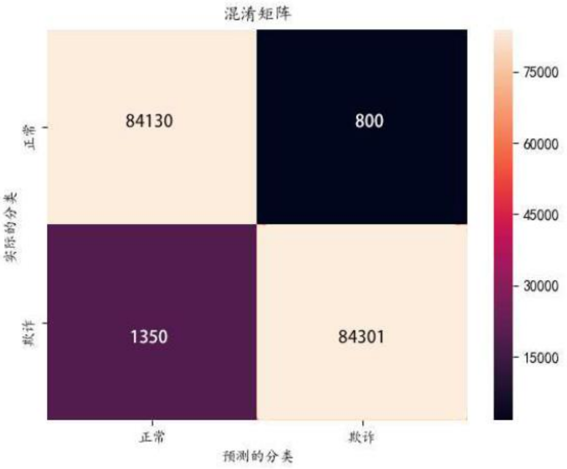
[Figure 6-6] Comparison of Recall Rate and Precision Rate

The Precision rate is compared to the Recall rate again. From Figure 6-6, it can be concluded that the Recall Rate is negatively correlated with the Precision rate, that is, the higher the Recall rate, the lower the Precision rate and vice versa. Set *threshold* = *3.6* and mark the error greater than the threshold as fraud. The prediction is as follows.



[Figure 6-7] Different Types of Reconstruction Errors for Oversampling

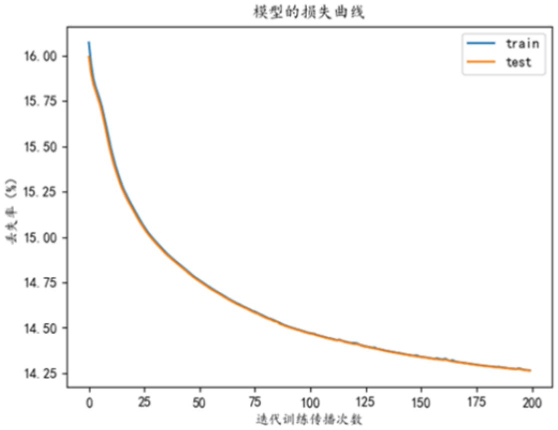
The following confusion matrix is constructed for different types of reconstruction errors, in which 84,301 are predicted to be fraudulent and are actually frauds, and 84,130 are predicted to be legal and are actually legal. It can be obtained from the error classification matrix that accuracy rate of classification is 98.74%, which is a high-quality classifier of good reference significance.



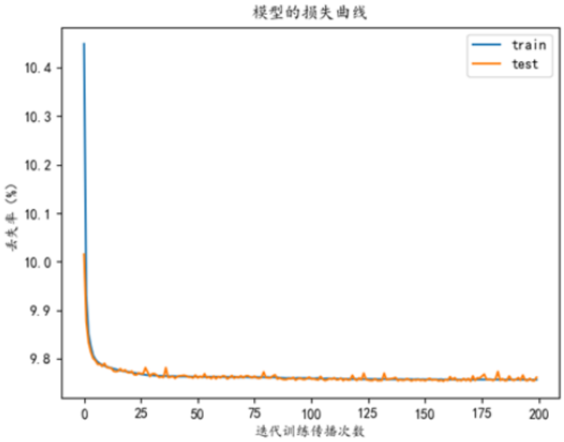
[Figure 6-8] Confusion Matrix of Different Types of Reconstruction Errors for Oversampling

6.5 Robustness Test

In the paragraphs above, when the Model III determines the epochs value, the assumption is 150. Selecting the subjective factors, the paper performs robustness [14] analysis to observe the results of the model when the epochs value is different from the previously assumption. To observe the effect, the function graph is produced to illustrate assumed epochs value and results of oversampling and undersampling. Graph is shown below.



[Figure 6-9] Loss Curve of the Model - Undersampling



[Figure 6-10] Loss Curve of the Model - Oversampling

As shown in Figure 6-10, loss of the undersampling model is basically below 0.145 after stabilization. After 200 epochs, iterations tend to converge better. As shown in Figure 6-10, the loss of the oversampling model is basically maintained below 0.097. After 50~200 epochs, iterations tend to converge better. For the general purpose of the model, choose 150 as the epoch value to achieve better convergence.

6.6 Improvement and Promotion

Over-fitting phenomenon will affect the result of Model III. In order to overcome it, this paper uses the absolute value function as the penalty term to compress the coefficient of the AutoEncoder, and introduces the parameter λ. When the parameter value becomes bigger, the penalty becomes larger, and the training result will become sparse, allowing the model to reach a balance between fitting ability and generalization ability. This is the role of the L1 paradigm regularization. In order to help the model to be quickly sparse, noise can be added to the input layer. And at the same time, the random gradient descent algorithm can be utilized to prevent the model from being affected by individualization or irrelevant input. This can make the training model converge quickly and enhance the balance ability of pair fitting and generalization of the model.

6.7 Model Evaluation & Comparison

Taking the oversampled data set as an example, compare Model II and Model III, including the basic principles, training time, F-score, error rate and so on. Results are shown in Table 6-2.

[Table 6-2] Comparison between Model II and Model III

Take Oversampling as Example	Model II	Model III
Method	**Traditional Machine Learning**	**Deep Learning**
Training Time	276s	3123s
F_score	0.973087	0.9874
Error Rate	2.65%	1.73%
Threshold	Self-adjusted	Manually-adjusted
Data Loss	22.50%	0

VII. Conclusion

This paper carries out the research on credit card fraud risk identification model and related early warning analysis. By using the construction of feature indicators, the credit card fraud early warning model basing on kernel density and logistic is built up constructively. At the same time, ensemble models based on single-layer decision tree classifier and deep neural network are also constructed, for better generalization and preventing over-fitting or under-fitting issues. In the process of model training, the paper focuses on handling the imbalance of data sets by training the model with oversampling and undersampling methods. Results show that the oversampling based training method can effectively improve the prediction accuracy of the model and guarantee a comparatively higher Recall rate. Finally, through the robustness analysis, it is verified that the model can be applied to solve the problems of poor data quality, cold start of the model, etc. It also proves that the model has high resistance to structural changes, high robustness and application values.

However, there is still room for improvement.

(1) Regarding the data: Credit card transaction data have a very high degree of confidentiality. The sample data in this paper are already second-hand processed after PCA with incomplete information, which makes the model training difficult. If original data are available, the effectiveness and robustness of the fraud risk model are expected to be higher with more practical significance.

(2) Regarding the model: In the models proposed in this paper, the construction of classifiers also requires training with training set. In the actual learning process, facing the challenge of more complex and volatile transaction data, identification accuracy can only be improved through learning from various training sets. Model re-learning through manual intervention may not meet the requirements for rapid data growth.

Regarding fraud risk identification and early risk warning, there is also a lot to improve. For example, the self-learning mechanism of the model could be improved so that it can continuously learn and evolve according to the changes in order to avoid the decline of recognition accuracy due to such data changes. Also considering the actual requirements of banks, the parameters are dynamically adjusted in the model for better universality. These issues shall be better addressed with the continuous optimization of data mining technology and deep learning algorithms. Meanwhile, with the continuous improvement of credit card fraud risk prevention and control system, the risk models proposed in the paper shall be more mature and stable.

Reference

[1] http://www.pbc.gov.cn/goutongjiaoliu/113456/113469/3607110/index.html 《2018 年第二季度支付体系运行总体情况》；
http://www.pbc.gov.cn/zhifujiesuansi/128525/128545/128643/2884430/index.html 《2010 年第二季度支付体系运行总体情况》.

[2] Hanagandi V, Dhar A, Buescher K. Density-based clustering and radial basis function modeling to generate credit card fraud scores[C]// Computational Intelligence for Financial Engineering, 1996. Proceedings of the IEEE/IAFE 1996 Conference on. IEEE, 1996:247-251.

[3] Chan P K, Stolfo S J. Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection[C]// International Conference on Knowledge Discovery and Data Mining. AAAI Press, 1998:164-168.

[4] Jurgovsky J, Granitzer M, Ziegler K, et al. Sequence Classification for Credit-Card Fraud Detection[J]. Expert Systems with Applications, 2018, 100.

[5] 肖琴. 基于互联网数据的个人信用风险评估的研究与应用. 2017-03-01.

[6] https://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/

[7] https://classeval.wordpress.com/introduction/basic-evaluation-measures/

[8] https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[9] 冯广庆，杨扬. 基于 Logistic 模型的大学生信用卡风险研究 [J]. 知识经济，2011(14): 55-55.

[10] https://www.quora.com/What-is-kernel-density-estimation

[11] 黄铃. 基于 AdaBoost-LC 的微博垃圾评论识别研究[D].重庆大学,2014.

[12] 邓俊锋，张晓龙. 基于自动编码器组合的深度学习优化方法[J]. 计算机应用，2016, 36(03): 697-702.

[13] 刘勘,袁蕴英. 基于自动编码器的短文本特征提取及聚类研究[J].北京大学学报(自然科学版),2015,51(02): 282-288.

[14] 吕大刚,宋鹏彦,崔双双,王闽雄. 结构鲁棒性及其评价指标[J]. 建筑结构学报，2011, 32(11): 44-54.

Appendix

```
import pandas as pd

#All_Data=pd.read_csv("resampled_addtitle.csv")
All_Data=pd.read_csv("../creditcard_data.csv")
Data= All_Data
Train=Data
Target=Train['Class']
#V1=Train['V1']
Train.drop('Class',axis=1,inplace=True) #使用 pandas 包中 drop（）函数用于删除 class 列
Train.drop('Amount',axis=1,inplace=True)
Train.drop('Time',axis=1,inplace=True)
correlationall = Train.corrwith(Target)
#correlational_V1 = V1.corr(Target)
print(correlationall)
correlationall.to_csv('correlation_v28.csv')

import pandas as pd
import matplotlib.pyplot as plt


data_corr_result = pd.read_csv("correlation_v28.csv", header = None)
col0=data_corr_result.iloc[:,0]
col1=data_corr_result.iloc[:,1]

names = col0.values
datas = col1.values
datas = list(map(abs, datas))

name_larges, data_larges = [], []
for name, data in zip(names, datas):
    if data >=0.3:
        name_larges.append(name)
        data_larges.append(data)

#设置输出的图片大小
figsize = 15,9
figure, ax = plt.subplots(figsize=figsize)
plt.tick_params(labelsize=18)
plt.bar(name_larges, data_larges, facecolor='blue', width=0.8)
plt.savefig('figure_3.png')
```

```
# -*- coding: utf-8 -*-
"""
"""


import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

#%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.float_format', lambda x: '%.4f' % x)

font1 = {'family' : 'Times New Roman',
         'weight' : 'normal',
         'size'   : 12,
}
font2 = {'family' : 'Times New Roman',
         'weight' : 'normal',
         'size'   : 18,
}
data_cr = pd.read_csv('../creditcard_data.csv')
data_cr.info()
data_cr.shape
data_cr.describe().T

data_cr.groupby('Class').size()
data_cr['Hour'] =data_cr["Time"].apply(lambda x : divmod(x, 3600)[0])
data_cr['Count'] =1
data_cr[data_cr["Class"]== 0]["Amount"].describe()
data_cr[data_cr["Class"]== 1]["Amount"].describe()
linenumber = data_cr.shape[0]
maxhour = (int)(data_cr['Hour'][linenumber-1])

data_hour1 = np.zeros(maxhour+1)
for hourindex in data_cr['Hour']:
    data_hour1[int(hourindex)] += 1

index = np.arange(maxhour+1)
```

```
fig=plt.figure(dpi=128,figsize=(8,4))
plt.xlabel('Hour',    font1)
plt.ylabel('Count', font1)

my_x_ticks = np.arange(0, maxhour+1, 1)
plt.xticks(my_x_ticks)
plt.xticks(fontsize=6, fontname = "Times New Roman")
plt.yticks(fontname = "Times New Roman")
plt.plot(index, data_hour1)
plt.savefig('linefig1.png', dpi=128)
plt.show()

data_hour2 = np.zeros(maxhour+1)
for item, hournumber in zip(data_cr["Class"], data_cr["Hour"]):
        if item == 1:
                data_hour2[int(hournumber)] += 1
index = np.arange(maxhour+1)
fig=plt.figure(dpi=128,figsize=(8,4))
plt.xlabel('Hour', font1)
plt.ylabel('Count', font1)
my_x_ticks = np.arange(0, maxhour+1, 1)
plt.xticks(my_x_ticks)
plt.xticks(fontsize=6, fontname = "Times New Roman")
plt.yticks(fontname = "Times New Roman")
plt.plot(index, data_hour2)
plt.savefig('linefig2.png', dpi=128)
plt.show()

data_hour3 = np.zeros(maxhour+1)
for item, hournumber in zip(data_cr["Class"], data_cr["Hour"]):
        if item == 0:
                data_hour3[int(hournumber)] += 1
index = np.arange(maxhour+1)
fig=plt.figure(dpi=128,figsize=(8,4))
plt.xlabel('Hour', font1)
plt.ylabel('Count', font1)
my_x_ticks = np.arange(0, maxhour+1, 1)
plt.xticks(my_x_ticks)
plt.xticks(fontsize=6, fontname = "Times New Roman")
plt.yticks(fontname = "Times New Roman")
plt.plot(index, data_hour3)
plt.savefig('linefig3.png', dpi=128)
plt.show()
```

```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(15, 6))
ax1.scatter(data_cr["Hour"][data_cr["Class"] == 1], data_cr["Amount"][data_cr["Class"] == 1],s=75,marker='s')
ax1.set_title('Fraud',font2)
ax1.set_ylabel('Amount', font2)

ax2.scatter(data_cr["Hour"][data_cr["Class"] == 0], data_cr["Amount"][data_cr["Class"] == 0],s=75,marker='s')
ax2.set_title('Normal',font2)
ax2.set_ylabel('Amount', font2)

labels1 = ax1.get_xticklabels() + ax1.get_yticklabels()
[label.set_fontname('Times New Roman') for label in labels1]
[label.set_fontsize(14) for label in labels1]
labels2 = ax2.get_xticklabels() + ax2.get_yticklabels()
[label.set_fontname('Times New Roman') for label in labels2]
[label.set_fontsize(14) for label in labels2]

plt.xlabel('Hour', font2)
#plt.ylabel('Amount', font2)
my_x_ticks = np.arange(0, maxhour+1, 1)
plt.xticks(my_x_ticks, fontname = "Times New Roman")

plt.savefig('fig4.png', dpi=128)
plt.show()



f, (ax1, ax2) = plt.subplots(2, 1, figsize=(15,6))
ax1.hist(data_cr["Amount"][data_cr["Class"]== 1], bins = 100,range=(0,2500),log=True)
ax1.set_title('Fraud', font2)
ax1.set_ylabel('Number of\nTransactions', font2)
ax2.hist(data_cr["Amount"][data_cr["Class"] == 0], bins = 100,log=True)
ax2.set_title('Normal', font2)
ax2.set_ylabel('Number of\nTransactions', font2)

labels1 = ax1.get_xticklabels() + ax1.get_yticklabels()
[label.set_fontname('Times New Roman') for label in labels1]
[label.set_fontsize(14) for label in labels1]

labels2 = ax2.get_xticklabels() + ax2.get_yticklabels()
```

```
[label.set_fontname('Times New Roman') for label in labels2]
[label.set_fontsize(14) for label in labels2]

plt.xlabel('Amount', font2)
#plt.ylabel('Number of\nTransactions', font2)
plt.savefig('fig5.png', dpi=128)
plt.show()



import random
from sklearn import preprocessing
from sklearn.neighbors import NearestNeighbors
import numpy as np
import pandas as pd
import csv
class Smote:
    """
    SMOTE 过采样算法.
    Parameters:
    -----------
    k: int
    选取的近邻数目.
    sampling_rate: int
    采样倍数, attention sampling_rate < k.
    newindex: int
    生成的新样本(合成样本)的索引号.
    """
    def __init__(self, sampling_rate=5, k=5):
        self.sampling_rate = sampling_rate
        self.k = k
        self.newindex = 0
    def fit(self, X, y=None):
        if y is not None:
            negative_X = X[y==0]
            negative_X = np.array(negative_X)
            X = X[y == 1]
            X = np.array(X)

        n_samples, n_features = X.shape
        # 初始化一个矩阵, 用来存储合成样本
        self.synthetic = np.zeros((n_samples * self.sampling_rate, n_features))
        synth_samples, synth_features = self.synthetic.shape
        # 找出正样本集(数据集 X) 中的每 一个样本在数据集 X 中的 k 个近邻
        knn = NearestNeighbors(n_neighbors=self.k).fit(X)
        for i in range(len(X)):
            k_neighbors = knn.kneighbors(X[i].reshape(1,-1),return_distance=False)[0]
            # 对正样本集 (minority class samples)中每个样本 ， 分别根据其 分别根据其 k 个
近邻生 成
            # sampling_rate 个新的样本
            self.synthetic_samples(X, i, k_neighbors)

        if y is not None:
            for i in range(61):
                self.synthetic=np.delete(self.synthetic, 0, axis=0)
            synth_samples, synth_features = self.synthetic.shape
#            return (np.concatenate((self.synthetic, X, negative_X), axis=0),
#                    np.concatenate(([1] * (len(self.synthetic) + len(X)), y[y == 0]), axis=0))
            return np.concatenate((self.synthetic, X, negative_X), axis=0)
        return np.concatenate((self.synthetic, X), axis=0)

    # 对正样本集 (minority class samples) 中每个样本 ， 分别根据其 k 个近邻生成
sampling_rate 个新的样本 个新的样本
    def synthetic_samples(self, X, i, k_neighbors):
        hang = len(self.synthetic) # 201809
        for j in range(self.sampling_rate):
            # 从 k 个近邻里面随机选择一个近邻
            neighbor = np.random.choice(k_neighbors)
            # 计算样本 X[i]X[i] X[i]与刚选择的近邻差
            diff = X[neighbor]   - X[i]
            # 生成新的数据
            if self.newindex < hang : # 201809
                self.synthetic[self.newindex] = X[i] + random.random() * diff
            self.newindex += 1
#            newindex_g = self.newindex
# ------ 通过采样获取 calss 为 1 的数据

#newindex_g = 0

dataf=pd.read_csv('../creditcard_data.csv')

print(dataf.shape)
count_class = pd.value_counts(dataf['Class'],sort= True).sort_index()
print (count_class)
"""
#数据标准化
```

```
from sklearn.preprocessing import StandardScaler #导入数据预处理模块
dataf['Amount'] = StandardScaler().fit_transform(dataf['Amount'].values.reshape(-1,1)) # -1 表示系统自动计算得到的行， 1 表示 1 列

#dataf = dataf.drop("Time", axis=1)
"""
columnname = dataf.columns.values.tolist()
print(columnname)

data0 = []
data1 = []
data3 = []
for i, element in enumerate(dataf['Class']):
    if element == 1:
        data1.append(dataf.iloc[i, :])
    else:
        data0.append(dataf.iloc[i, :])
X = np.array(data1)
smote = Smote(sampling_rate=577, k=492)

#data1=smote.fit(X).tolist()
#data2=smote.fit(df, df['Class']).tolist()
data2=smote.fit(dataf, dataf['Class'])

#将过采样数据顺序打乱
m2 = len(data2)
index2 = np.array(range(0, m2))
np.random.shuffle(index2)
data22 =[]
for i in range(m2):
    data22.append(data2[index2[i]])

#将过采用数据保存为 csv
csvFile1 = open('data_Over_sampling.csv','w', newline='') # 设置 newline，否则两行之间会 ，否则两行之间会 空一行
writer = csv.writer(csvFile1)
writer.writerow(columnname)
for i in range(m2):
    writer.writerow(data22[i])

csvFile1.close()    #data_Over_sampling.csv
print('data_Over_sampling.csv saved')
```

```
m0 = len(data0)
m1 = len(data1)
index1 = np.random.randint(m0,size=m1)        # np.random.randint(2,size=5)#array([0, 1, 1, 0, 1])
for i in range(m1):
        index = index1[i]
        data3.append(data0[index])


data5 =np.concatenate((data3, data1), axis=0)

#将下采样数据顺序打乱
m5 = len(data5)
index5 = np.array(range(0, m1*2))
np.random.shuffle(index5)
data55 =[]
for i in range(m5):
        data55.append(data5[index5[i]])

#将下采样数据保存为 csv
csvFile2 = open('data_Lower_sampling.csv','w', newline='')
writer = csv.writer(csvFile2)
writer.writerow(columnname)
for i in range(m5):
        writer.writerow(data55[i])
csvFile2.close()    #data_Lower_sampling.csv
print('data_Lower_sampling.csv saved')

# -*- coding: utf-8 -*-
"""

"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedKFold

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
import random
import warnings
```

```
from pylab import mpl
warnings.filterwarnings('ignore')
mpl.rcParams['font.sans-serif' ] = ['SimHei']
def readdata():
    alls =pd.read_csv('data_Over_sampling_LR.csv')
    x=alls.drop('Class',axis=1)
    y=alls['Class']
    x=x.values
    y=y.values
    return x,y

X,y=readdata()

lrn = LogisticRegression()
N = 5
N_iter = 2

for it in range(N_iter):
    skf = StratifiedKFold(n_splits = N, shuffle = True)
    for train_index, test_index in skf.split(X, y):
        X_train, y_train = X[train_index], y[train_index]
        X_test, y_test = X[test_index], y[test_index]
        lrn.fit(X_train, y_train)
        y_pred=lrn.predict(X_test)
        y_prob = lrn.predict_proba(X_test)[:,lrn.classes_[1]]

classes=['0' , '1' ]
y_pred = lrn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
if lrn.classes_[0] == 1:
    cm = np.array([[cm[1,1], cm[1,0]], [cm[0,1], cm [0,0]]])
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
thresh = cm.max() / 2.

import itertools
for i , j in itertools .product(range(cm.shape[0]), range(cm.shape[1])):
    plt .text(j , i , cm[i, j ],
    horizontalalignment="center",
    color="white" if cm[i, j ] > thresh else "black")
plt .tight_layout()
plt .ylabel('Actual Class') #真实类别
plt .xlabel('Predicted Class') #预测类别
plt .show()

# -*- coding: utf-8 -*-
"""

"""

import random
from sklearn import preprocessing
from sklearn.neighbors import NearestNeighbors
import numpy as np
import pandas as pd
import csv
class Smote:
    """
    SMOTE过采样算法.
    Parameters:
    -----------
    k: int
    选取的近邻数目.
    sampling_rate: int
    采样倍数, attention sampling_rate < k.
    newindex: int
    生成的新样本(合成样本)的索引号.
    """
    def __init__(self, sampling_rate=5, k=5):
        self.sampling_rate = sampling_rate
        self.k = k
        self.newindex = 0
    def fit(self, X, y=None):
        if y is not None:
            negative_X = X[y==0]
            negative_X = np.array(negative_X)
            X = X[y == 1]
            X = np.array(X)

        n_samples, n_features = X.shape
        # 初始化一个矩阵, 用来存储合成样本
        self.synthetic = np.zeros((n_samples * self.sampling_rate, n_features))
        synth_samples, synth_features = self.synthetic.shape
        # 找出正样本集 (数据集 X) 中的每 一个样本在数据集 X 中的 k 个近邻
        knn = NearestNeighbors(n_neighbors=self.k).fit(X)
        for i in range(len(X)):
            k_neighbors = knn.kneighbors(X[i].reshape(1,-1),return_distance=False)[0]
            # 对正样本集 (minority class samples)中每个样本 ，分别根据其 分别根据其 k 个
近邻生 成
            # sampling_rate 个新的样本
            self.synthetic_samples(X, i, k_neighbors)

        if y is not None:
            for i in range(61):
                self.synthetic=np.delete(self.synthetic, 0, axis=0)
            synth_samples, synth_features = self.synthetic.shape
#            return (np.concatenate((self.synthetic, X, negative_X), axis=0),
#                    np.concatenate(([1] * (len(self.synthetic) + len(X)), y[y == 0]), axis=0))
            return np.concatenate((self.synthetic, X, negative_X), axis=0)
        return np.concatenate((self.synthetic, X), axis=0)

    # 对正样本集 (minority class samples) 中每个样本 ，分别根据其 k 个近邻生成
sampling_rate 个新的样本 个新的样本
    def synthetic_samples(self, X, i, k_neighbors):
        hang = len(self.synthetic) # 201809
        for j in range(self.sampling_rate):
            # 从 k 个近邻里面随机选择一个近邻
            neighbor = np.random.choice(k_neighbors)
            # 计算样本 X[i]X[i] X[i]与刚选择的近邻差
            diff = X[neighbor]   - X[i]
            # 生成新的数据
            if self.newindex < hang : # 201809
                self.synthetic[self.newindex] = X[i] + random.random() * diff
            self.newindex += 1
#            newindex_g = self.newindex
# ------ 通过采样获取 calss 为 1 的数据

#newindex_g = 0

dataf=pd.read_csv('../creditcard_data.csv')

#数据标准化
print(dataf.shape)
count_class = pd.value_counts(dataf['Class'],sort= True).sort_index()
print (count_class)
from sklearn.preprocessing import StandardScaler #导入数据预处理模块
```

```
dataf['Amount'] = StandardScaler().fit_transform(dataf['Amount'].values.reshape(-1,1)) # -1表示系统自动计算得到的行，1表示1列

dataf = dataf.drop("Time", axis=1)
columnname = dataf.columns.values.tolist()
print(columnname)

data0 = []
data1 = []
data3 = []      # class =0 random samnple from data0
for i, element in enumerate(dataf['Class']):
    if element == 1:
        data1.append(dataf.iloc[i, :])
    else:
        data0.append(dataf.iloc[i, :])
X = np.array(data1)
smote = Smote(sampling_rate=577, k=492)


data2=smote.fit(dataf, dataf['Class'])

#将过采样数据顺序打乱
m2 = len(data2)
index2 = np.array(range(0, m2))
np.random.shuffle(index2)
data22 =[]
for i in range(m2):
    data22.append(data2[index2[i]])

#将过采用数据保存为csv
csvFile1 = open('data_Over_sampling.csv','w', newline=") # 设置newline，否则两行之间会 ，否则两行之间会 空一行
writer = csv.writer(csvFile1)
writer.writerow(columnname)
for i in range(m2):
    writer.writerow(data22[i])

csvFile1.close()      #data_Over_sampling.csv
print('data_Over_sampling.csv saved')


#随机选取492个class=0 的数据项
m0 = len(data0)
```

```
m1 = len(data1)
index1 = np.random.randint(m0,size=m1)      # np.random.randint(2,size=5)#array([0, 1, 1, 0, 1])
for i in range(m1):
    index = index1[i]
    data3.append(data0[index])

#下采样数据：将 class=1  和 class=0  合并到一个数组中，   总数据项 966 个
data5 =np.concatenate((data3, data1), axis=0)    # size 492*2

#将下采样数据顺序打乱
m5 = len(data5)
index5 = np.array(range(0, m1*2))
np.random.shuffle(index5)
data55 =[]
for i in range(m5):
    data55.append(data5[index5[i]])

#将下采样数据保存为  csv
csvFile2 = open('data_Lower_sampling.csv','w', newline='')
writer = csv.writer(csvFile2)
writer.writerow(columnname)
for i in range(m5):
    writer.writerow(data55[i])
csvFile2.close()    #data_Lower_sampling.csv
print('data_Lower_sampling.csv saved')



# -*- coding: utf-8 -*-
"""


"""
#模型 3
import pylab
import numpy as np
import pickle
import matplotlib.pyplot as plt
#%matplotlib inline
from scipy import stats

import tensorflow as tf
import seaborn as sns
from pylab import rcParams
from sklearn.model_selection import train_test_split
from keras.layers import Input,Dense
from keras.models import Model,load_model
from keras.callbacks import ModelCheckpoint,TensorBoard
from keras import regularizers
import pandas as pd
from pylab import mpl
import time
mpl.rcParams['font.sans-serif'] = ['KaiTi'] #  指定默认字体
mpl.rcParams['axes.unicode_minus'] = False #   解决保存图像是负号 '-'显示为方块的问题
#data =pd.read_csv('creditcard_data.csv')
data =pd.read_csv('data_Lower_sampling.csv')
#查看数据格式
# print(df.head())
#查看数据结构
# print(df.shape)
count_classes = pd.value_counts(data['Class'],sort = True).sort_index()
count_classes.plot(kind='bar',rot=0)
plt.title('class 的分布情况')
plt.xlabel('Class')
plt.ylabel('数目')
plt.show()
#
RANDOM_SEED=42
x_train,x_test=train_test_split(data,test_size=0.3,random_state=RANDOM_SEED)
y_train =x_train['Class']
#x_train=x_train[x_train.Class==0]
x_train=x_train.drop(['Class'],axis=1)

y_test=x_test['Class']
x_test=x_test.drop(['Class'],axis=1)

x_train=x_train.values
x_test=x_test.values
y_train=y_train.values
y_test=y_test.values

print(x_train.shape)
input_dim=x_train.shape[1]
encoding_dim=14
input_layer=Input(shape=(input_dim ,))
encoder=Dense(encoding_dim,activation='tanh',activity_regularizer=regularizers.l1(10e-5))(input_layer)
encoder=Dense(int(encoding_dim/2),activation='relu')(encoder)
```

```
decoder=Dense(int(encoding_dim/2),activation='tanh')(encoder)
decoder=Dense(input_dim,activation='relu')(decoder)
autoencoder=Model(inputs=input_layer,outputs=decoder)
nb_epoch=150
batch_size=32
autoencoder.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])
checkpointer=ModelCheckpoint(filepath='data_Lower_sampling.h5',verbose=0,save_best_only=True)
tensorboard=TensorBoard(log_dir='.logs',histogram_freq=0,write_graph=True,write_images=True)
history=autoencoder.fit(x_train,
x_train,epochs=nb_epoch,batch_size=batch_size,shuffle=True,validation_data=(x_test,x_test),verbose
=1,callbacks=[checkpointer,tensorboard]).history
print(history)
#
plt.plot(history['loss'])
plt.plot(history['val_loss'])
plt.title('Loss Curve of the Model') #型的损失曲线
plt.ylabel('Loss Rate（%）') #丢失率
plt.xlabel('Epoch') #迭代训练传播次数
plt.legend(['train','test'],loc='upper right')
plt.show()
# data_Over sampling  过采样模型
# data_Lower sampling  下采样模型
autoencoder=load_model('data_Lower_sampling.h5') # 201809
#autoencoder=load_model('creditcard_data.h5')      # 201809
prediction=autoencoder.predict(x_test)
mse=np.mean(np.power(x_test -prediction,2),axis=1)
error_df=pd.DataFrame({'reconstruction_error':mse,'true_class':y_test})
print(error_df.describe())
from sklearn.metrics import (confusion_matrix,precision_recall_curve,auc,roc_curve,
recall_score,classification_report,precision_recall_fscore_support)

fpr,tpr,thresholds=roc_curve(error_df.true_class,error_df.reconstruction_error)
# ROC  曲线
roc_auc = auc(fpr, tpr)
plt.title('ROC Curve') #ROC 曲线
plt.plot(fpr, tpr, label='AUC = %0.4f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r -- ')
plt.xlim([-0.001, 1])
plt.ylim([0, 1.001])
plt.ylabel('True Positive Rate ') #真阳性率
plt.xlabel('False Positive Rate ') #假阳性率
plt.show()
```

```
#精确率和召回率
precision, recall, th = precision_recall_curve(error_df.true_class, error_df.reconstruction_error)
plt.plot(recall, precision, 'b', label='Precision-Recall curve')
plt.title('Recall vs Precision')
plt.xlabel('Recall ') #召回率
plt.ylabel('Precision ') #精准率
plt.show()
#准确率
plt.plot(th, precision[1:], 'b', label='阈值-精准率曲线')
plt.title('不同阈值的精准率')
plt.xlabel('阈值')
plt.ylabel('精准率')
plt.show()
#召回率 f
plt.plot(th, recall[1:], 'b', label='阈值-召回率曲线')
plt.title('不同阈值的召回率')
plt.xlabel('阈值')
plt.ylabel('召回率')
plt.show()
threshold=2.9
y_pred = [1 if e > threshold else 0 for e in error_df.reconstruction_error.values]
conf_matrix = confusion_matrix(error_df.true_class, y_pred)

groups = error_df.groupby('true_class')
fig, ax = plt.subplots()
for name, group in groups:
    ax.plot(group.index, group.reconstruction_error, marker='o', ms=3.5, linestyle='', label= 'Legal ' if name == 1 else 'Fraudulent') #正常交易  欺诈交易
ax.hlines(threshold, ax.get_xlim()[0], ax.get_xlim()[1], colors='r', zorder=100, label='Threshold') #阈值
ax.legend()
#plt.title('不同类别的重建误差')
plt.ylabel('Reconstruction Errors') #重建误差
plt.xlabel('Data Point Index') #数据点索引
plt.show()
LABELS=['Legal','Fraudulent'] #'正常','欺诈'
#
plt.figure()
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt='d')
plt.title('Confusion Matrix') #混淆矩阵
plt.ylabel('Actual Class') #实际的分类
plt.xlabel('Predicted    Class') #预测的分类
plt.show()
```

```
# #
TP=0
FN=0
FP=0
TN=0
f1data=[]
thresho=[]
for i in range(0,500):
    threshold = i*0.01
    thresho.append(threshold)
    y_pred = [1 if e > threshold else 0 for e in error_df.reconstruction_error.values]
    conf_matrix = confusion_matrix(error_df.true_class, y_pred)
    for T, P in zip(y_pred, y_test):
        if T == 1 and P == 1:
            TN += 1
        if T == 1 and P == 0:
            FN += 1
        if T == 0 and P == 1:
            FP += 1
        if T == 0 and P == 0:
            TP += 1

    #
    # print('TP' + str(TP))
    # print('FN' + str(FN))
    # print('FP' + str(FP))
    # print('TN' + str(TN))
    p = (TP + TN) / (TP + FN + FP + TN )
    print(str(p))
    f1data.append(p)
plt.xlabel("阈值") # X 轴的文字
plt.ylabel("F1") # Y 轴的文字
plt.title("阈值 -F1 曲线") # 图表的标题
plt.plot(thresho, f1data)
plt.show() # 显示图片
```

matlab code:

AdaBoostTestmain.m

```
 csv_data = csvread('creditcard.csv',1,0);
 downsampling(csv_data, 'downsampling_data.csv');
 csv_alldata = csvread('downsampling_data.csv');
 fastPCA(csv_alldata(:,1:30),csv_alldata(:,31) ,8 );
 AdaBoostTest();
```

AdaBoostTest.m

```
function [same] = AdaBoostTest()
    fprintf(' 读取数据中 ... \n');
    csv_data = csvread('下采样 8 个指标乱序.csv', 1, 0);
    fprintf('读取完成.\n');
    same = zeros(8,1);
    len = fix(length(csv_data(:,1)));
    train_len = fix(len* 0.7);
    train_data = csv_data(1:train_len,1:8);
    train_label = csv_data(1:train_len,9);
    test_data = csv_data(train_len+1:len,1:8);
    test_label = csv_data(train_len+1:len,9);
    fprintf('下采样\n')
    for i=1:8
        %length(train_data)
        ens = fitensemble(train_data(:,1:i),train_label,'AdaBoostM1' ,100,'tree','type','classification');
        predict_label = predict(ens, test_data(:,1:i));
        TP = 0;
        FN = 0;
        FP = 0;
        TN = 0;
        for i2=1:length(predict_label)
            if test_label(i2,1) == 1 && predict_label(i2,1) == 1
                TP = TP+ 1;
            end
            if test_label(i2,1) == 1 && predict_label(i2,1) == 0
                FN = FN + 1;
            end
            if test_label(i2,1) == 0 && predict_label(i2,1) == 1
                FP = FP + 1;
            end
            if test_label(i2,1) == 0 && predict_label(i2,1) == 0
                TN = TN + 1;
            end
            if test_label(i2,1) == predict_label(i2,1)
                same(i,1) = same(i,1) + 1;
            end
        end
        same(i,1)=same(i,1)/length(predict_label)*100;
        fprintf('%d 列： \n',i);
        fprintf('TP: %f FN: %f\nFP: %f TN: %f\n',TP,FN,FP,TN);
        P = TP / (TP+ FP);
        R = TP / (TP+ FN);
        fprintf('P: %f R:%f\nF1: %f\n',P,R,2*P*R/(P+R));
        fprintf('准确率  : %f %%\n',same(i,1));
    end
end
```

downsampling.m

```
function [] = downsampling(data, resultFilename)
    result = [];
    randomNUM = [];
    len = length(data(:,1));
    for i=1:483
        while(1)
            num = fix(rand(1) * len) + 1;
            fprintf('%d\n',num)
            if any(randomNUM==num)==0
                break
            end
        end
        result = [result;data(num,:)];
        csvwrite(resultFilename,result)
    end
end
```

fastPCA.m

```
function pcaA = fastPCA( A,B, k )
[r c] = size(A);
meanVec = mean(A);
Z = (A-repmat(meanVec, r, 1));
covMatT = Z * Z';
[V D] = eigs(covMatT, k);
V = Z' * V;
for i=1:k
     V(:,i)=V(:,i)/norm(V(:,i));
end
pcaA = Z * V;
pcaA1=[pcaA,B]
csvwrite('下采样 8 个指标乱序.csv',pcaA1)
```

Acknowledgement

Nanjing Foreign Language School
Teacher Qing Yan
My Parents
People who helped me during this research project

Biography

Zhongnian Tao, student in Senior 3 (1) Advanced Science Class in Nanjing Foreign Language School.

- *National Finalist (Top 15)* –2017 Dongrun-Yau Science Award (Computer)
- *First Prize* – National Olympiad in Informatics in Provinces (Senior Group) from Grade 8 to Grade 11 consecutively;
- *Champion, Gold Medal* – National Olympiad in Informatics 2017 Winter Camp;
- *Gold Medal* – The 29th International Olympiad in Informatics China Team Selection Competition;
- *Silver Medal* – The 11th Asia Pacific Informatics Olympiad;
- *Jiangsu Provincial First Prize* – 2017 National Olympiad in Mathematics in Provinces;
- *Global Top 1%* – American Mathematics Competition 12;
- Student of 2018 China Talent Program administered by China Association for Science and Technology and China Ministry of Education;
- Student of 2018 Sakura Science Program, a Japan-Asian Youth Exchange Program.

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员：陶重年　　　　指导老师：尹青

2018 年 9 月 15 日