Development of a Machine Learning-Based Financial Risk Control System

Zhigang Hu
Utah State University

DEVELOPMENT OF A MACHINE LEARNING-BASED FINANCIAL RISK CONTROL SYSTEM

by

Zhigang Hu

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Curtis Dyreson, Ph.D. Major Professor	Hengda Chen, Ph.D. Committee Member
Rakesh Kaundal, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Interim Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2021

ABSTRACT

Development of A Machine Learning-based Financial Risk Control System

by

Zhigang Hu, Master of Science

Utah State University, 2021

Major Professor: Curtis Dyreson, Ph.D.
Department: Computer Science

With the gradual end of the COVID-19 outbreak and the gradual recovery of the economy, more and more individuals and businesses are in need of loans. This demand brings business opportunities to various financial institutions, but also brings new risks. The traditional loan application review is mostly manual and relies on the business experience of the auditor, which has the disadvantages of not being able to process large quantities and being inefficient. Since the traditional audit processing method is no longer suitable some other method of reducing the rate of non-performing loans and detecting fraud in applications is urgently needed by financial institutions.

This thesis develops a risk control system using real, desensitized loan applicant data provided by financial institutions. The system uses a decision tree model to create a loan user bad debt rate prediction system and a logistic regression model to create a loan user score card system. The Borderline-SMOTE algorithm is used to solve the sample imbalance problem in the dataset. A gradient boost decision tree method is employed to improve the overall performance of the system. The final risk control system improves the approval speed, reduces the risk, and lowers the rate of non-performing loans.

This thesis makes the following innovations.

- Based on the theory of feature engineering methods, a feature derivation method and feature derivation formula are created to meet the project requirements and are applied to the decision tree model with significant results, in that the classification features finally adopted were all derived from the results of feature derivation.

- Based on the idea of logistic regression algorithm, a new method of classifying feature datasets using the reverse of classification result set data is proposed and applied to user credit scoring system with significant classification results.

- Based on the data equal frequency binning algorithm, a new data binning algorithm is proposed by using the chi-square test combined with the method of calculating information value. It makes up for the defect that the equal-frequency binning algorithm cannot calculate the optimal number of bins.

(69 pages)

PUBLIC ABSTRACT

Development of A Machine Learning-based Financial Risk Control System

Zhigang Hu

With the gradual end of the COVID-19 outbreak and the gradual recovery of the economy, more and more individuals and businesses are in need of loans. This demand brings business opportunities to various financial institutions, but also brings new risks. The traditional loan application review is mostly manual and relies on the business experience of the auditor, which has the disadvantages of not being able to process large quantities and being inefficient. Since the traditional audit processing method is no longer suitable some other method of reducing the rate of non-performing loans and detecting fraud in applications is urgently needed by financial institutions.

In this project, a financial risk control model is built by using various machine learning algorithms. The model is used to replace the traditional manual approach to review loan applications. It improves the speed of review as well as the accuracy and approval rate of the review. Machine learning algorithms were also used in this project to create a loan user scorecard system that better reflects changes in user information compared to the credit card systems used by financial institutions today. In this project, the data imbalance problem and the performance improvement problem are also explored.

ACKNOWLEDGMENTS

It was an honor to be invited by JobLogic-X to participate in this development project, which lasted for almost 8 months.During this 8-month project, with the assistance of other partners, we successfully created a “Machine Learning-based Financial Risk Control Model” using daily business data from financial institutions and some data provided by third parties, and achieved good application results in internal testing.

I give special thanks to my supervisor Dr.Curtis, family, friends, and colleagues for their encouragement, moral support, and patience as I worked my way from the initial proposal writing to this final document. I could not have done it without all of you.

Zhigang Hu

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The Internet has transformed traditional financial practices. Internet finance brings convenience, speed and high returns to financial enterprises, but it also brings considerable risks such as confusion in credit management, bad lending and malicious deception. The scale of social financing is growing and private economic activities are becoming more and more active, meanwhile, credit cards, housing loans, auto loans, consumer loans and other personal credit businesses are also growing rapidly. As the total amount of loans is increasing, the number of defaults by users is also rising, and some banks are experiencing an increasing rate of non-performing loans. Therefore, it is very important for finance enterprises to reduce the risk of Internet finance by studying the risk of user default, assessing the risk level, and doing a good job of risk control.

The rapid development of Internet finance has brought serious challenges to the traditional financial model, especially in the risk control system. In the financial sector, the following types of risks are common [1].

- *Market risk* refers to the uncertainties in the value of the company's underlying assets, liabilities, or income due to exposure to a highly dynamic financial market [2, 3]. Market risk includes risks such as interest rate risk, exchange rate risk, and commodity price risk.

- *Credit risk* refers to the uncertainty involving its creditors' ability to perform their contractual obligation (loan defaults or bankruptcy) [4]. This is applicable for both retail lenders (lenders who provide loans to individuals or retail customers) and corporate lenders (lenders who provide loans to businesses).

- *Insurance and Demographic risk,* which is more specific to the insurance industry, refers to the variance in insurance claim experience due to unpredictable events (*e.g.,* catastrophes, car accidents) as well as uncertain ties involved with the demographic profile of its policyholders (*e.g.,* mortality). This risk can be further broken down to mortality risk, catastrophe risk, and non-catastrophe risk [2].

- *Operational risk* refers to the risk of loss due to the unpredictability of business operation or loss of performance due to faulty or fraudulent business practices. Operational risk can be further broken down into business risk and event risk. Business risk indicates the uncertainty related to business performance(*e.g.,* uncertainty in earnings, demand volatility, customer churn, faulty business operations) and event risk includes uncertainty in events that have an adverse effect in business operations (*e.g.,* fraudulent activities, change in regulations).

In contrast to traditional risk control technology, risk control is mostly performed by each financial institution's own credit risk control team in the form of a manual audit. The biggest problem of this audit mode is the slow audit speed, which usually takes more than a week from the audit to the release of funds, which cannot adapt to today's fast-changing financial model. Machine Learning is a computational method that uses past information to improve performance in a specific task(s) or make accurate predictions [5,6]. By building a model using machine learning, we can quickly assess the user's credit rating, loan amount and other related data.

In general, the application of Machine Learning in Financial Risk Management (FRM) falls into the following categories.

- *Loan Risk Assessment Modeling* - This area is by far the most used area of machine learning in FRM. Financial institutions consider both quantitative and qualitative approaches to assign risk level associated with a specific application [7]. Recently, machine learning algorithm have been widely used in credit card default prediction. By constructing several models, it can predict clients' payment precisely and effectively. Additionally, compared with traditional models, machine learning algorithms can address the problem of large-scale data and operate promptly [8]. Several unsupervised learning models used for default or bankruptcy prediction in credit risk management rely on clustering the credit applicant profiles [9].

- *Credit Card User Score System* - In the age of Internet finance, credit is everywhere. In order to judge the quality of customers, the use of credit scoring and the implementation of customer scoring has become a difficult point in the current field of risk control. So it is very important to establish a credit card user score system, through which the lenders can determine risk involved in lending. Shukla built this kind of system by using a parallel social spider algorithm [10]. Sudha used SVM and random forests to detect credit card fraud [11].

This thesis develops a machine learning-based financial risk control system. The system has four parts.

1. The first part is an effective *bad debt prediction system.* Bad debt rate control is a problem that financial institutions face. Current bad debt rate analysis and control systems are based on statistics combined with finance to find out the main risk points in user data, common risk points are monthly income and asset/liability ratio. In this thesis, the main focus is on predicting the bad debt rate of new users by using a classification model. One of the problems in building a classifier is that the financial dataset we use has incomplete information, so an important precursor to creating the classifier is deriving features in original dataset. We experimentally confirm that the classifier we build achieves high accuracy.

2. The second part is a *user credit scoring system.* In FRM, user credit scoring system is a core system. The system must fully reflect the user's credit history, and it is also an important basis for deciding whether to grant a loan to the user, the loan amount, and the interest rate of the loan. The current credit history system that is widely used by major financial institutions is the FICO system, but this system is only a credit record system, not the scoring system that financial institutions need. It can

only score a user's credit based on their age, income status, and whether they have a history of defaults. The system reflects the history of the user, for example, there are a large number of users in the system simply because they have defaulted on their loans for various reasons, but have paid them back in a timely manner. This situation will result in a low FICO score, which will lead to a failed audit and potential loss of a quality customer. The advantage of machine learning lies in the use of historical data to predict the future behavior of users, so machine learning algorithms can be applied to the creation of this system. But the challenge is how to make the final scorecard system interpretable and accurate. It has to be able to accurately rate users based on their history and also explain the basis of the rating to business people. We build a regression based user credit scoring system. Common regression models are binary or multi-classify users based on feature data sets. But we need a system to perform multiple classifications on the feature dataset based on the user classification results, and the classification results must be optimized to reflect the user information as much as possible. Therefore, in this system, we simply borrow the idea of a regression algorithm and create a new algorithm to classify the feature set data based on the result set data, and ensure the interpretability and accuracy of the algorithm.

3. The third part is solving the problem of *data imbalance* in financial risk control system. A natural feature of financial data is data imbalance (fewer users actually have defaulted than paid loans in full), so this reality should be fully considered when dealing with such data. In this thesis several algorithms to deal with the data imbalance problem are tried and the results are given for comparison. We experimentally show that a small improvement in accuracy can be achieved through addressing data imbalance.

4. The fourth part is an integrated algorithm to improve overall performance. It is well illustrated in many papers and kaggle competition practices that integrated algorithms have a great performance improvement compared to single algorithms [TODO - give citations]. We also explore an integrated algorithm in this thesis and experimentally

show that while an integrated algorithm can indeed improve the overall performance of the algorithm, the improvement is not significant.

This thesis makes the following contributions.

- Based on the theory of feature engineering methods, a feature derivation method and feature derivation formula are created to meet the project requirements and are applied to the decision tree model with significant results, in that the classification features finally adopted were all derived from the results of feature derivation.
- Based on the idea of logistic regression algorithm, a new method of classifying feature datasets using the reverse of classification result set data is proposed and applied to user credit scoring system with significant classification results.
- Based on the data equal frequency binning algorithm, a new data binning algorithm is proposed by using the chi-square test combined with the method of calculating information value. It makes up for the defect that the equal-frequency binning algorithm cannot calculate the optimal number of bins.

CHAPTER 2
THE PROCESS OF DEVELOPING A MACHINE LEARNING-BASED FINANCIAL RISK CONTROL SYSTEM

Model development in a machine learning application requires the processes shown in Figure 2.1. The application starts with getting data. The main sources of data in this project are daily business data of financial institutions, data provided by qualified third parties, and log data of websites. The data has been desensitized according to the requirements of personal privacy.

The next step in the process is to clean the data. Data cleaning and feature engineering is an important procedure in machine learning. The quality of the data will directly affect the learning effect of the model. There is a famous idiom in the field of machine learning that "garbage in, garbage out." This step is often neglected in papers because the datasets are *idealized* data obtained after many cleanings. Real-world datasets are often less than ideal, which leads to many algorithmic models failing to achieve the desired effect in practice. The reason for this is not the algorithm itself, but the quality of the data. Data cleaning and feature engineering methods are many and complex, common methods are data type conversion, missing value filling, one-hot coding, biased data correction, and feature derivation and deletion.

Model development comes after feature engineering. Model development is the most central step in machine learning. We use four kinds of machine learning models in this thesis: regression, classification, clustering, and association analysis.

1. A *regression model* predicts a dependent variable from an independent variable or variables. Regression is often a supervised learning method in machine learning, usually divided into univariate versus multivariate regression. The coefficients of the dependent variable are calculated by fitting the data to the regression model during the training process. Regression problems are widely used, examples include stock

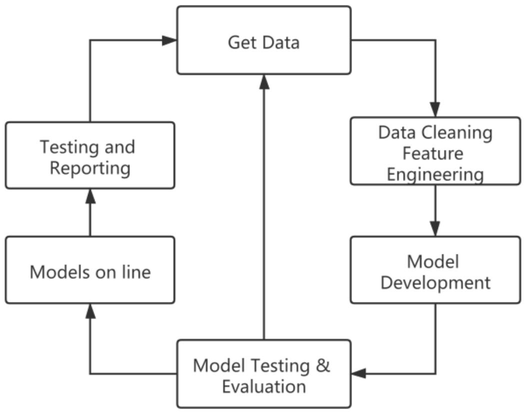


Fig. 2.1: The process of machine learning modeling

price prediction, house price prediction, and temperature prediction. The credit user scoring card system developed in this thesis is a specific application of regression.

2. *Classification* is an important machine learning model. Classification predicts the class of an unknown entity. Classification can be supervised (the classes are known a priori) or unsupervised (the classifier learns the classes). A machine learning model can also output the probability of the classification result while processing the classification. In this thesis classification is used for loan risk predication (into good or bad risk) and credit card user prediction (rating the card risk from A to F).

3. *Clustering* can also be supervised or unsupervised learning. In unsupervised clustering the number of clusters is not known in the initial condition. Clustering is groups similar samples into a cluster. Commonly used methods include density clustering and grid clustering, and the problem of measuring sample distance is also judged by using Euclidean distance and Minkowski clustering. Cluster modeling usually uses K-Means, a simple and easy method, is widely used in pattern recognition, image recognition, and bioinformatics. In this project, the K-Means algorithm is mainly used for abnormal user monitoring.

4. *Correlation analysis* focuses on the correlation between types of data, often as an association rule, which studies the correlation between different data in an event.

After choosing the modeling method and building the model, the next step is testing and evaluation. In this step the training results of the model are tested with the test dataset and the parameters of the model are adjusted based on the test results to achieve the best machine learning results. In addition, an evaluation team involving business people is often needed to assess the feasibility of the modeling results in real business scenarios. When the results of the model do not reach the expected results, we need to re-examine the quality of the input data set when the problems of the model algorithm itself are excluded, so this stage is often an iterative process. The main methods used in this phase are cross-validation, grid-search, and plotting learning curves.

Once the model has passed evaluation and testing, it is ready for live application. However, in practice, we find that after the model has been online for a period of time, there is often a phenomenon of model degradation, that is, the model works better for a period of time when it is just online, but when it runs for a period of time, the effect is much worse than when it was just online. After analyzing the reasons, we found that the user data is relatively concentrated in certain types over a period of time, for example, there is a period of time when most of the loan applications are for home loans, and another period of time when most of the users are for car loans. In this case, the original distribution of the data set is affected thus leading to degradation of the model. Then, we need to add the new data distribution to the model and retrain the model. Therefore, it is not the end of the project after the model goes online, but we need to continuously monitor the effect of the online model, and analyze the reasons for the degradation once the model is found, and retrain the model.

CHAPTER 3
A DECISION TREE-BASED LOAN REVIEW SYSTEM

An important part of a financial institution's daily business is the review of loan applicants. At present, with the increasing economic activity and lower interest rates on loans, the number of applications for loans of all kinds has risen sharply. While these loans bring lucrative profits to financial institutions, they also pose a relatively large risk. Traditional loan applications are reviewed by auditors based on their work experience, and this method has two key drawbacks.

1. *Low speed* - An auditor can review a limited number of applications per day. Generally, an experienced auditor can only review about 30-50 applications a day. However, the number of applications has increased dramatically and the number of experienced auditors is limited, so many financial institutions have a large backlog of applications to be reviewed. This can result in a large number of quality customers being lost to other financial institutions. The longer audit cycles also lead to a slowdown in economic activity for the entire community.

2. *Low accuracy* - Traditional auditors can only rely on limited information such as credit score, income status, and marital status to analyze and make judgments with previous work experience. For example, when most financial institutions review home loans, they often set the monthly mortgage payments to be no more than 1/3 of the net income. However, personal income is a dynamic data, and the income proof available to financial institutions is only the applicant's current income, which leads to a large number of qualified applicants who cannot get the loan amount they need and miss out on the desired home, while some applicants default on the loan due to lower income. All of this can result in losses to the loan originator.

To address both drawbacks, we built a decision tree-based loan review system. A decision tree is a non-parametric supervised learning method that summarizes decision rules from a series of data with features and labels, and presents these rules in a tree-like graph structure to solve classification and regression problems. Decision tree algorithms are easy to understand, applicable to all kinds of data, and perform well in solving various problems, especially various integrated algorithms with tree model as the core, which are widely used in various industries and fields.

This chapter describes the decision tree-based loan review system starting with a description of the data set used to train the system.

3.1 The Data Set and Feature Engineering

We use user oil consumption data provided by a third-party, desensitized and integrated. The details of the data-set are as shown in Figure 3.1. The data set has a total of 50609 records and 18 features, where *bad_ind* is an expert provided label of whether or not to violate the contract, which we use here as the y-label.

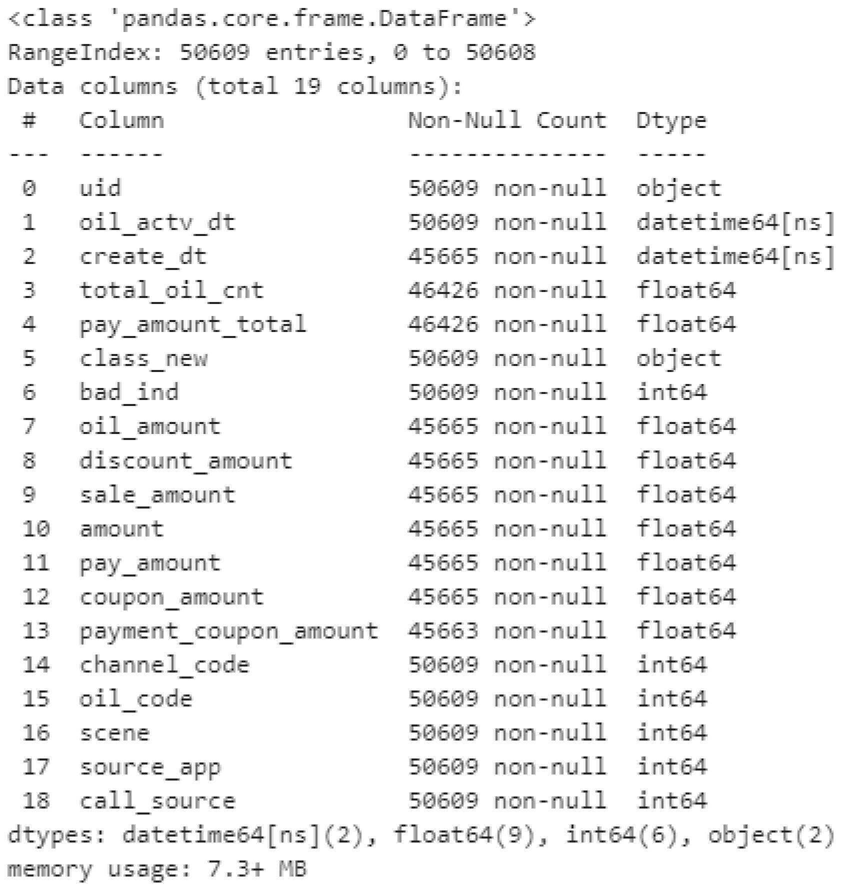
Some of the features have missing values. The missing values are shown in Figure 3.2. The missing values account for nearly 10% of the total data volume, so we cannot simply delete records with missing values or fill them with 0. We need to adopt different filling strategies for missing values.

Based on the nature of the features, we divide the feature set into three categories.

1. *org_lst* - No need to do a special transformation, instead remove duplicates. This category contains features such as *'uid'*, *'create_dt'*, *'oil_actv_dt'*, *'class_new'*, *'bad_ind'*.

2. *agg_lst* - Aggregate numeric values. This category contains features such as *'oil_amount'*, *'discount_amount'*, *'sale_amount'*, *'amount'*, *'pay_amount'*, *'coupon_amount'*, and *'payment_coupon_amount'*.

3. *dstc_lst* - Count text variables. This category contains features such as *'channel_code'*, *'oil_code'*, *'scene'*, *'source_app'*, and *'call_source'*.



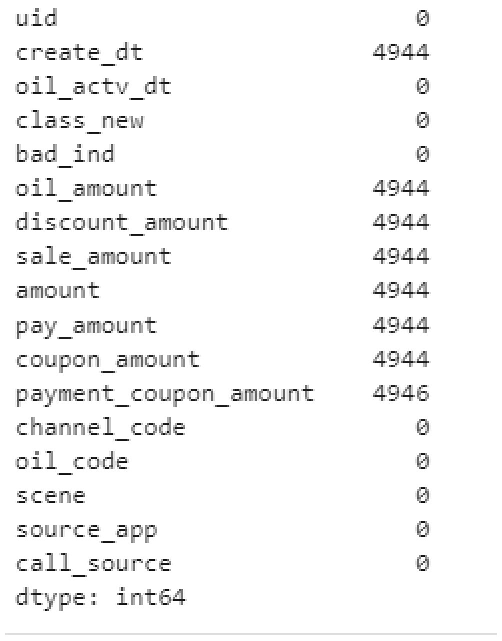
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50609 entries, 0 to 50608
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   uid                    50609 non-null  object
 1   oil_actv_dt            50609 non-null  datetime64[ns]
 2   create_dt              45665 non-null  datetime64[ns]
 3   total_oil_cnt          46426 non-null  float64
 4   pay_amount_total       46426 non-null  float64
 5   class_new              50609 non-null  object
 6   bad_ind                50609 non-null  int64
 7   oil_amount             45665 non-null  float64
 8   discount_amount        45665 non-null  float64
 9   sale_amount            45665 non-null  float64
 10  amount                 45665 non-null  float64
 11  pay_amount             45665 non-null  float64
 12  coupon_amount          45665 non-null  float64
 13  payment_coupon_amount  45663 non-null  float64
 14  channel_code           50609 non-null  int64
 15  oil_code               50609 non-null  int64
 16  scene                  50609 non-null  int64
 17  source_app             50609 non-null  int64
 18  call_source            50609 non-null  int64
dtypes: datetime64[ns](2), float64(9), int64(6), object(2)
memory usage: 7.3+ MB
```

Fig. 3.1: The detail information of the data-set

3.1.1 Handle missing values

Inevitably, missing values appear in the raw data for various reasons. However, in machine learning algorithms, if there are missing values in the dataset, the program cannot be executed. Therefore, one of the important tasks in feature engineering is to fill in the missing values. When the number of missing values is small, the easiest way is to remove the missing values. Other methods include median/mean/zero value filling, etc.

From the Figure 3.2,we can know that there are a total of 4944 data items with missing values. The number of missing values accounts for 9.8% of the total data volume, and if these missing values are deleted directly, it will inevitably cause information loss and thus affect the final prediction results. Looking further, we can reveal that the missing values exist in the following feature sets: *'create_dt' 'oil_amount' 'discount_amount'*



```
uid                          0
create_dt                 4944
oil_actv_dt                  0
class_new                    0
bad_ind                      0
oil_amount                4944
discount_amount           4944
sale_amount               4944
amount                    4944
pay_amount                4944
coupon_amount             4944
payment_coupon_amount     4946
channel_code                 0
oil_code                     0
scene                        0
source_app                   0
call_source                  0
dtype: int64
```

Fig. 3.2: Missing values information

'sale_amount' 'amount' 'pay_amount' 'coupon_amount' 'payment_coupon_amount'.In terms of data types, the missing values are both date and numeric data, so we need to use different filling algorithms.

Do a filling to *'creat_dt'*, fill it with *'oil_actv_dt'*, and intercept 6 months of data. When constructing the variables you cannot do a direct accumulation of all the historical data. Otherwise, the variable distribution will change significantly over time.

The algorithm for missing value filling is described as follows.

First filter out the values with maximum time interval >180

$$\text{dtn=oil\_actv\_dt-create\_dt}$$

Then for numerical data, find the value of the maximum time interval less than 180 and take their means:

$$\text{missing\_value=mean(max(}\textit{agg\_list}\text{),where dtn<180)}$$

3.1.2 Feature Derivation

In the feature engineering approach of machine learning, we often compute the correlation between features and labels to delete those feature sets that have little or no impact on the final learning results but consume a lot of computing resources. However, the deletion of these feature sets will inevitably lead to the loss of data information, which will affect the final machine learning results. However, after some transformation or combination of such features, they often have strong information value and can be helpful for data sensitivity and machine learning experience, so we need to do some derivative work on the basic features.

In this dataset, the set of features that can actually be used in the decision tree for classification is *agg_lst*, which has a total of seven features. The number of features is on the low side to achieve a good classification result. Therefore, feature derivation can be considered for this feature set. There are seven numerical features in this feature set, and most of the information contained in this dataset should be contained in this data set. In order to further explore the hidden data information in the data set to ensure the classification effect of the decision tree model, feature derivation can be performed on this feature set.

Feature construction work is not entirely dependent on technology, it requires us to have rich knowledge or practical experience in related fields, to observe and analyze the raw data based on actual business scenarios, to think about the potential form of the problem and data structure, and to find out some physically meaningful features from the raw data.

Kanter proposed the concept and method of Deep Feature Synthesis [12] and verified the effectiveness of the method. Based on the ideas provided in the paper and previous work experience, the feature set was derived as shown in Figure 3.3. After feature engineering, the dimensionality of our dataset is 11099*70.

3.2 Going from a Data Table to a Decision Tree

There are an exponential number of decision trees that can be constructed from a table. Decision-tree algorithms prune decision trees by measuring some property of the tree. The method we chose uses *impurity*. Generally speaking, the lower the impurity, the better

$$DFea\_num = \{count(x_i), x_i \neq 0, i = i...n\} \qquad (1)$$

$$DFea\_mean = \frac{\sum_{i}^{n} x_i}{n} \qquad (2)$$

$$DFea\_\max = \max\{x_i, x_i \neq nan, i = 1...n\} \qquad (3)$$

$$\begin{gathered} DFea\_\max = \max\{x_i, x_i \neq nan, i = 1...n\} \\ DFea\_cnt = \{count(x_i), i = 1...n\} \end{gathered} \qquad (4)$$

$$DFea\_\text{var} = \frac{\sum_{i=1}^{n} (x_i - DFea\_mean)^2}{n} \qquad (5)$$

$$DFea\_gap = DFea\_\max - DFea\_\min \qquad (6)$$

$$DFea\_coe = \frac{DFea\_mean}{DFea\_\text{var}} \qquad (7)$$

Fig. 3.3: Kanter's Deep Feature Synthesis

the decision tree fits the training set. Many decision tree algorithms are centered around optimizing some impurity-related metric.

Impurity is calculated based on nodes, each node in the tree will have an impurity, and the impurity of child nodes must be lower than that of the parent node. That is, the leaf node must have the lowest impurity in the same decision tree.

In the decision tree algorithm, there are two ways to calculate impurity.

1. *Entropy* - This equation was proposed by Quinlan [13].

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t)\ \log_2(p(i|t))$$

2. Gini impurity - This equation was proposed by De'ath [14].

$$Gini(t) = 1 - \sum_{i=0}^{c-1} p(i|t)^2$$

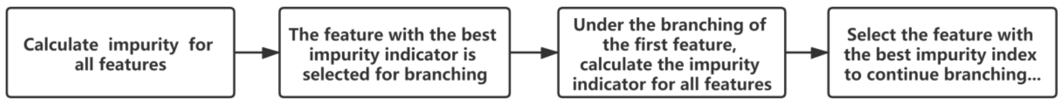

Calculate impurity for all features
The feature with the best impurity indicator is selected for branching
Under the branching of the first feature, calculate the impurity indicator for all features
Select the feature with the best impurity index to continue branching...

Fig. 3.4: The basic process of Decision Tree

In the equation t represents the given node and i represents any classification of the label. $p(i|t)$ represents the proportion of label classification i on node t.

Information entropy is more sensitive to impurity than the Gini coefficient and penalizes impurity the most. However, in practical use, the effects of information entropy and Gini coefficient are similar. The calculation of information entropy is slower than the Gini coefficient because the calculation of the Gini coefficient does not involve logarithms. Information entropy is more sensitive to impurity, so the growth of the decision tree is more refined when the information entropy is used as an indicator. For high-dimensional data or data with a lot of noise, the information entropy can be easily over-fitted, and the Gini coefficient tends to work better in this case. When the model is under-fitted, *i.e.,* when the model does not perform well on both the training and test sets, it is better to use the information entropy coefficient. When the model does not fit well, *i.e.,* when the model does not perform well on both the training and test sets, use the Gini coefficient.

The basic process of decision trees can actually be briefly summarized as shown in Figure 3.4:

Without restrictions, a decision tree grows until the measure of impurity is optimal, or until no more features are available. Such a decision tree tends to over-fit, which means that it will perform well on the training set and poorly on the test set. The sample data we collect cannot be identical to the overall situation, so when a decision tree has an overly good interpretation of the training data, the rules it finds must contain the noise in the training sample and make it sensitive to unknown data.

In order to have better generalization of the decision tree, we have to prune the decision tree. The pruning strategy has a huge impact on the decision tree, and the right pruning strategy is the core of the optimization the core of the decision tree algorithm.

There are two common pruning strategies: max_depth and min_samples_leaf.

1. max_depth - Limit the maximum depth of the tree, and cut off all branches that exceed the set depth. This is the most widely used pruning parameter and is very effective at high dimensionality and low sample size. Growing the decision tree by one more layer doubles the demand on the sample size, so Therefore, limiting the tree depth can effectively limit over-fitting.

2. min_samples_leaf and min_samples_split - *min_samples_leaf*, each child of a node after branching must contain at least min_samples_leaf training samples, otherwise branching not happen, or the branching will happen in the direction of satisfying min_samples_leaf for each child node. It is usually used with max_depth, which has the magic effect of smoothing out the model in the regression tree. Setting the number of this parameter too small will induce over-fitting, and setting it too large will prevent the model from learning the data. If the sample size contained in the leaf nodes is very variable it is recommended to enter a floating point number as a percentage of the sample size to be used. Also, this parameter ensures the minimum size of each leaf and can be used in regression problems It can avoid the appearance of low variance, over-fitting leaf nodes in the regression problem. *min_samples_split*, a node must contain at least min_samples_split training samples for this node to be allowed to be branched, otherwise branching will not happen.

We can find the optimal parameters for the decision tree by plotting the learning curve as shown in Figures 3.5, 3.6 and 3.7. From the learning curves, it appears that the optimal combination of parameters is *max_depth*=2, *min_samples_leaf*=500, *min_samples_split*=5000.

3.3 Decision tree result output and related decisions

After training the decision tree model using the optimal parameters the resulting tree is shown in Figure 3.8. We can see that the average bad debt rate is 4.7% when pay_amount_tot $\leq$ 240387.5 is used as the first level screening condition, while the bad

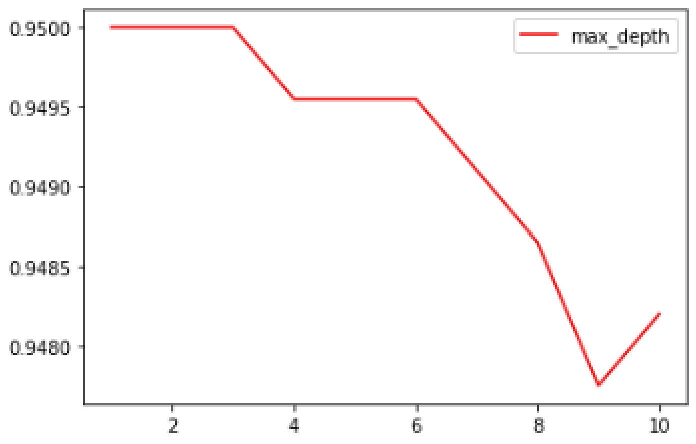


Fig. 3.5: The learning curve of *max_depth*

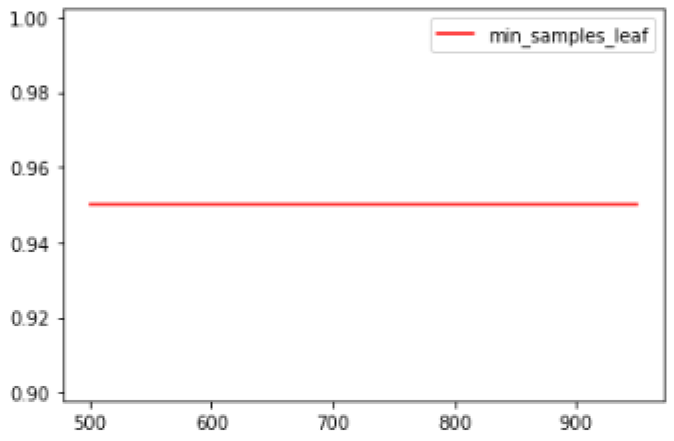


Fig. 3.6: The learning curve of *min_samples_leaf*

debt rate is 2.1% when pay_amout_num $\leq$ 3.5 is used as the second level screening condition, both of which meet the project target requirement of no more than 5% bad debt rate.

Using the decision tree, users can be classified into A, B and C categories as illustrated in Table 3.1.

3.3.1 Analysis and comparison of results

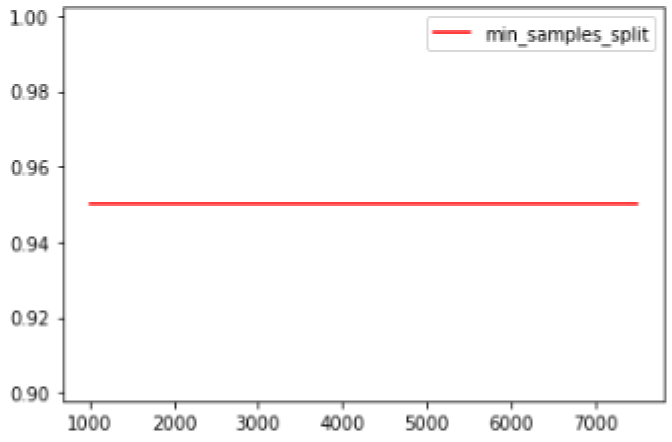


Fig. 3.7: The learning curve of *min_samples_split*

Classification condition	**Classification Result**
pay_amount_tot>240387.5 and *pay_amount_num*>3.5	Card_A
pay_amount_tot>240387.5 and *pay_amount_num*<=3.5	Card_B
pay_amount_tot<=240387.5	Card_C

Table 3.1: User Categories

[TODO: explain these three tables. The previous table had only three categories: A, B, C. But these tables have grades A through F.]

Through the results comparison table we can find: the number of applications approved increased from 1,901 to 5,052, and the approval rate increased from 17.1% to 45.5%, while the bad debt rate decreased from 2.9% to 1.6%.

This fully demonstrates that the system can significantly improve the pass rate while also significantly reducing the bad debt rate, and the accuracy and efficiency of the audit is greatly improved compared to the original manual audit method.

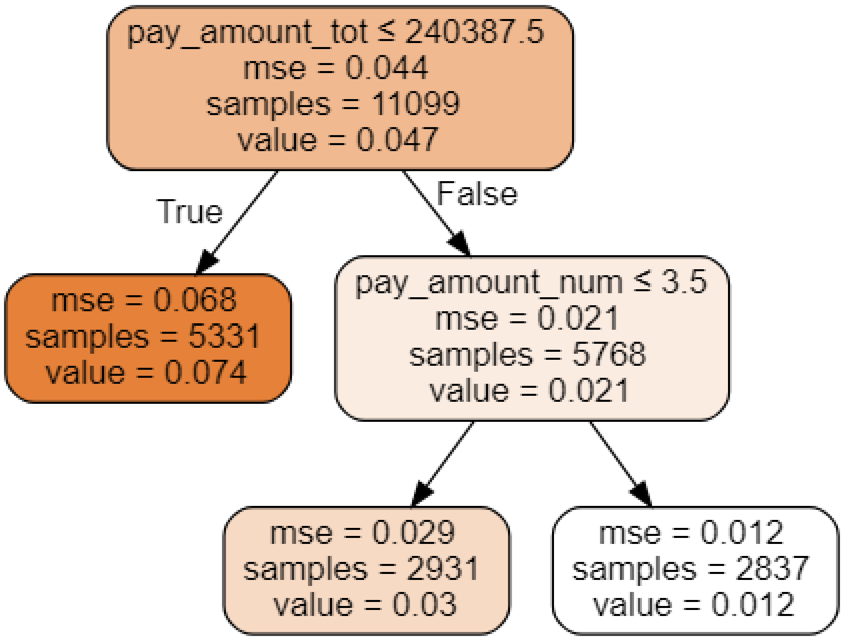
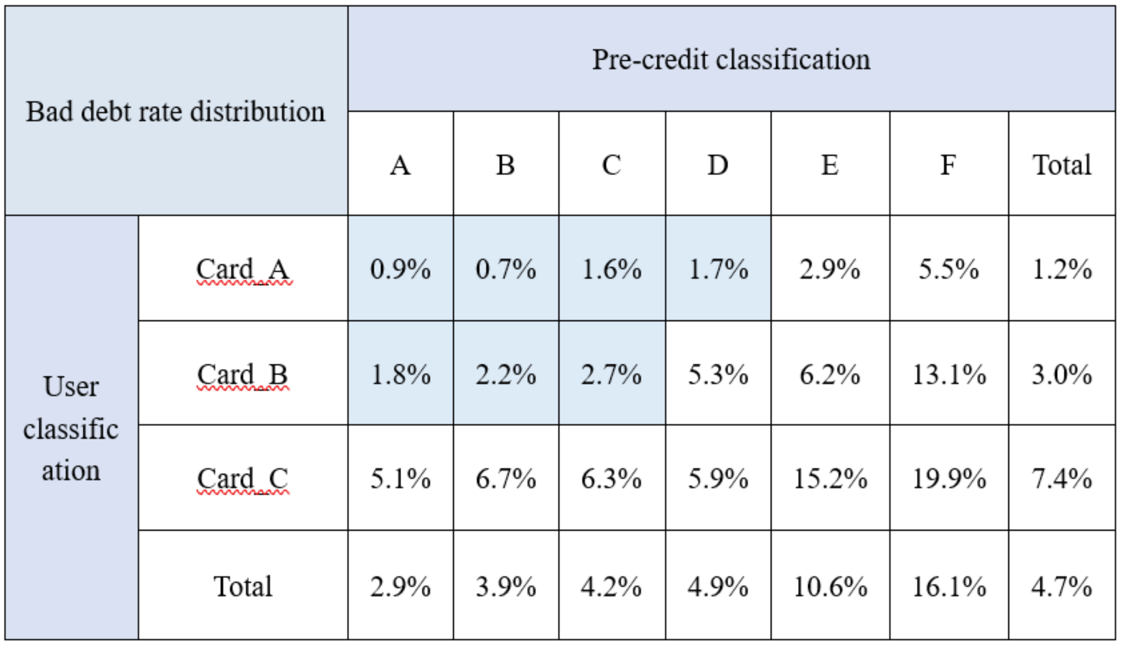


Fig. 3.8: The output of decision tree model



Bad debt rate distribution		Pre-credit classification						
		A	B	C	D	E	F	Total
User classification	Card_A	0.9%	0.7%	1.6%	1.7%	2.9%	5.5%	1.2%
	Card_B	1.8%	2.2%	2.7%	5.3%	6.2%	13.1%	3.0%
	Card_C	5.1%	6.7%	6.3%	5.9%	15.2%	19.9%	7.4%
	Total	2.9%	3.9%	4.2%	4.9%	10.6%	16.1%	4.7%

Table 3.2: Bad debt rate distribution

Number of people distribution		Pre-credit classification						
		A	B	C	D	E	F	Total
User classification	Card_A	4.9%	12.6%	3.9%	2.6%	0.9%	0.7%	25.6%
	Card_B	5.0%	12.5%	4.1%	3.2%	0.9%	0.8%	26.4%
	Card_C	7.3%	21.6%	7.5%	6.8%	2.4%	2.4%	48.0%
	Total	17.1 %	46.7%	15.5%	12.6%	4.3%	3.8%	100.0%

Table 3.3: User distribution

	Applicants approved	Approval rate	Bad rate
Current plan	5052	45.5%	1.6%
Original Plan	1901	17.1%	2.9%

Table 3.4: Comparison of results

CHAPTER 4
LOGISTIC REGRESSION-BASED CREDIT SCORING MODEL

Credit scoring model is an advanced technical tool in consumer credit management, and is one of the most central management techniques for banks, credit card companies, personal consumer credit companies, telecommunication companies, utility service companies, insurance companies and other business entities involved in consumer credit. It is widely used in credit card life cycle management, auto loan management, housing loan management, personal loan management, and other consumer credit management. It plays a very important role in marketing, credit approval, risk management, account management, customer relationship management and other areas.

Credit scoring models use advanced data mining techniques and statistical analysis methods to systematically analyze a large amount of data such as consumers' demographic characteristics, credit history records, behavior records, and transaction records, to mine the behavior patterns and credit characteristics embedded in the data, capture the relationship between historical information and future credit performance, and develop a predictive model to comprehensively assess consumers' future with a credit score of a certain credit performance. For example, the probability of incurring a bad debt loss in the next year, the potential to generate revenue for the bank. or the probability that the consumer will accept credit card marketing as a basis for consumer credit management decisions. If developed scientifically and applied correctly, credit scoring models can provide consumer credit managers with a large amount of highly predictive information that can help them develop effective management strategies to effectively develop markets, control risks, and exploit revenues with a high degree of accuracy, thus achieving high efficiency in consumer credit business.

4.1 Key approaches to credit score modeling

The first is credit analysis with customer classification (profiling and segmentation) as the core. This is to use some descriptive analysis (such as mean, variance, frequency, probability distribution, etc.) and exploratory analysis (such as class clustering analysis, factor analysis, correlation analysis, etc.) to conduct a relatively simple analysis of customer credit information, to observe and refine customer behavior characteristics. We also conduct preliminary assessment of customers' future performance and classify customers into categories on this basis. For example, customers are classified according to the income of consumers or customers, according to the good or bad historical performance, according to the amount of revolving credit usage and the activity of transactions, etc. The characteristic is to get some insight as a basis for decision making through non-systematic analysis (ad hoc analysis). This method uses a simple algorithm, small amount of data, and gives only a very limited basis for decision making, so the method is gradually being replaced.

The second is the credit scoring model with predictive modeling as the core. This is a major breakthrough in credit analysis technology, which extracts a large number of derivative variables reflecting consumer behavior and creditworthiness through deep mining of external information (such as consumer credit history information collected by credit bureaus) and internal information (such as credit card transaction information and behavioral information) of consumer credit institutions, and uses advanced mathematical and statistical techniques to synthesize the information of various variables, so as to It also uses advanced mathematical and statistical techniques to synthesize the information of various variables, so as to systematically predict the future credit performance of customers in a certain area and rank them by a score as a basis for decision making. This method has been applied earlier and has been used for more than 20 years, and major companies and organizations have accumulated a lot of relevant experience in using it, so it is still a relatively mainstream modeling method. Typical methods include data mining techniques using empirical research [15], using network-based models to improve credit score accuracy [16], and using fuzzy theory to create credit scoring models [17].

The third is a credit scoring model with decision modeling (DM) at its core. This is a step beyond pure predictive models because it not only predicts the future credit performance of a consumer from information about his or her behavioral characteristics, but also predicts that the credit decision itself has an impact on future performance. The fundamental difference between decision models and predictive models is that predictive models predict the future credit performance of consumers based on their historical behavioral characteristics, and future credit performance is only a function of consumers' own credit characteristics; whereas decision models aim to quantify the impact of decisions with different future credit service profiles. Therefore, the sensitivity to the decision varies. This is the so-called interaction effect. In recent years, decision modeling techniques have been applied to optimize consumer credit management decisions by predicting, measuring, and comparing the impact of different decisions on business management objectives and then selecting the management decision that is most beneficial to achieve the objectives.

Machine learning techniques, which have become more and more widely used in recent years, are also widely used in this field. Mhina [18] proposed collecting information related to cell phone users, and using K-Means clustering method to score the credit of users who apply for loans through cell phones. Zhang [19] uses deep learning to solve the problem that the single classification method used in other methods is only good for classifying one type of data, but poor for classifying other types of data. And the algorithm's performance has been significantly improved to enable real-time online analysis of data. More research and applications use the classical algorithm of logistic regression and its improvements [20–23]. In these approaches, it is common to set the credit scoring model as a classification model, the difference being whether it is a dichotomous or multi-classification problem. In this project, a new classification approach is explored.

4.2 Logistic Regression

Linear regression is the simplest regression algorithm in machine learning. Its equation is expressed as follows.

$$z = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$

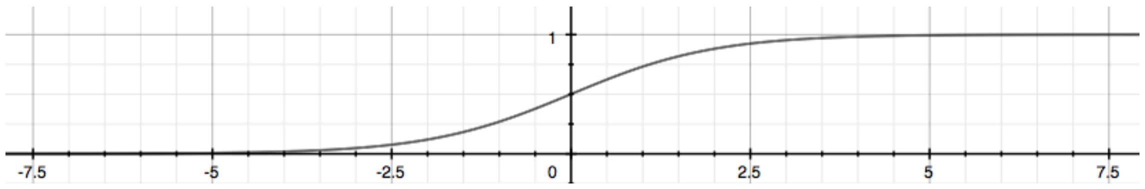


Fig. 4.1: The curve of Sigmoid Function

θ are collectively referred to as the parameters of the model, where θ_0 is known as the *intercept* and $\theta_1 \ldots \theta_n$ is known as the coefficient. If this equation is represented by the matrix, x and θ can all be viewed as a column matrix, yielding the following.

$$z = [\theta_0, \ldots, \theta_n] \times \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} = \theta^T \; x(x_0 = 1)$$

The task of linear regression is to construct a prediction function that maps the linear relationship between the input feature matrix x and the label value y. The core of constructing a prediction function is to find the parameters of the model θ^T and θ_0. Through the function z, Linear regression uses the input feature matrix X to output a continuous set of label values *y_pred*, to accomplish various tasks of predicting continuous type variables. Then, if our labels are discrete variables, in particular, if they are discrete variables that satisfy the 0-1 distribution variables, what do we do?

Transform the linear regression equation z to $g(z)$ and let the values of $g(z)$ are distributed between (0,1) and the sample is labeled as category 0 when $g(z)$ is close to 0, and category 1 when $g(z)$ is close to 1. This gives a classification category model. And this link function for logistic regression, is Sigmoid function:

$$g(z) = \frac{1}{1 + e^{-2}}$$

In linear regression, we replace with $z = \theta^T x$, then we can get the general form of the binary logistic regression model:

$$g(x) = y(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Linear regression has strict requirements on data, such as labels must meet normal distribution, co-linearity between features needs to be eliminated.

Logistic regression is a good fit for linear relationships. Data with extremely strong linear relationships between features and labels, such as those for credit card fraud, score card modeling, marketing prediction in e-commerce, and other related data are all strong points of logistic regression. Although there is now a gradient boosting GDBT, which is more effective than logistic regression and is also used by many data consulting companies, but the dominance of logistic regression in the financial field, especially in the banking industry, is still unshakable.

Computation of logistic regression is quickly. For linear data, logistic regression is very fast to fit and compute, and the computational efficiency is better than SVM and Random Forest.

The classification results returned by logistic regression are not fixed 0 and 1, but class probability numbers presented as decimals. We can therefore use the results of logistic regression as continuous data. For example, when creating a scorecard, we not only need to determine whether a customer will default, but also need to give a definite "credit score". The calculation of this credit score requires logarithmic odds calculated using class probabilities, while classifiers like decision trees and random forests can produce classification results but cannot help us calculate the credit score. decision trees and random forests can produce classification results, but they cannot help us calculate the score. Logistic regression is essentially a classifier that returns log odds and performs well on linear data, and it is mainly used in the field of finance. Its mathematical purpose is to solve for the values of the parameters θ that allow the model to fit the data the best, to construct a prediction function, and then to input the feature matrix into the prediction function to calculate the logistic regression result y.

4.2.1 Loss function of binary logistic regression

In machine learning modeling, we have to observe how the model performs on the training set, and how it performs on the test set. We model for optimal model performance on the test set, so the model evaluation metrics are often used to measure the performance of the model on the test set. However, logistic regression has the need to solve parameters θ based on training data and wants the trained model to fit the training data as closely as possible, that is, the closer the model's prediction accuracy on the training set is to 100%, the better.

The loss function of the logistic regression is derived from the Great Likelihood Estimation, and the specific results can be written as follows.

$$J(\theta) = -\sum_{i=1}^{m}(y_i \times \log(y_\theta(x_i) + (1 - y_i) \times \log(y_\theta(x_i)))$$

Where θ is a set of parameters solved for, m is the number of samples, y_i is the true label on sample i, $y_\theta(x_i)$ is the value of the logistic regression on sample i based on the parameters θ calculated the logistic regression return value from x_i is the value of each feature of sample i. Our goal is to solve for the value θ that minimizes $J(\theta)$. Since we pursue the minimum of the loss function to make the model perform optimally on the training set, another problem may be raised: if the model represents excellent on the training set but performs poorly on the test set, the model will be over-fitting. Although logistic regression and linear regression are inherently under-fitting models, we still need techniques to control over-fitting to help us tune the models, and control of over-fitting in logistic regression is achieved through regularization.

4.2.2 Regularization in logistic regression

Regularization is the process used to prevent over-fitting of the model. Two options are commonly used, L1 regularization and L2 regularization, which are implemented by adding multiples of the L1 and L2 paradigms of the parameter vector θ after the loss function, respectively. This increased paradigm is called the “regular term” and also the “penalty term”. We use this to adjust the degree of model fit. The L1 paradigm is expressed as the sum of the absolute values of each parameter in the parameter vector, and the L2 paradigm is expressed as the squared sum of the squares of each parameter in the parameter vector.

$$J(\theta)_{L1} = C \times J(\theta) + \sum_{j=1}^{n} |\theta_j| (j \geq 1)$$

$$J(\theta)_{L2} = C \times J(\theta) + \sqrt{\sum_{j=1}^{n} (\theta_j)^2} (j \geq 1)$$

Although both L1 regularization and L2 regularization can control over-fitting, they do not have the same effect. As the regularization strength gradually increases (C gradually becomes smaller), the parameter values will gradually become smaller, but L1 regularization will compress the parameter to 0, and L2 regularization will only make the parameter as small as possible and will not take it to 0. In the process of gradual strengthening of L1 regularization, the parameters of features that carry little information and contribute little to the model will become zero faster than those of features that carry a huge amount of information and contribute a significant amount to the model. Therefore, L1 regularization is essentially a feature selection process, which governs the “sparsity” of parameters. The stronger the L1 regularization, the more parameters in the parameter vector become 0, the more sparse the parameters are, and the fewer features are selected to prevent over-fitting. Thus, if the feature size is large and the data dimension is high, we tend to use L1 regularization. Due to this property of L1 regularization, feature selection for logistic regression can be done by an embedding method. In contrast, L2 regularization will try to make each feature contribute to the model in some small way during the strengthening process, but features that carry little information and do not contribute much to the model will have parameters very close to 0. Usually, if our main purpose is only to prevent over-fitting, choosing L2 regularization is sufficient. However, if the model still over-fits after

Features / Tags	**Meaning**
SeriousDlqin2yrs	Showing 90 days or more of past due behavior (*i.e.,* defining good and bad customers)
RevolvingUtilizationOfUnsecuredLines	Loan and credit card available amount to total amount ratio
age	*Borrower age*
NumberOfTime30-59DaysPastDueNotWorse	Number of times in the past two years when there were 35-59 days past due but did not progress to worse
DebtRatio	Monthly debt repayment, alimony, living expenses divided by monthly gross income
MonthlyIncome	Monthly income
NumberOfOpenCreditLinesAndLoans	Number of open-end loans and credits
NumberOfTimes90DaysLate	Number of times 90 days past due or worse in the last two years
NumberRealEstateLoansOrLines	Number of mortgages and real estate loans, including home equity lines of credit
NumberOfTime60-89DaysPastDueNotWorse	Number of times in the past two years when there were 60-89 days past due but did not progress to worse
NumberOfDependents	Number of dependents in the family excluding oneself (spouse, children, etc.)

Table 4.1: The data dictionary of Borrower Information datasets

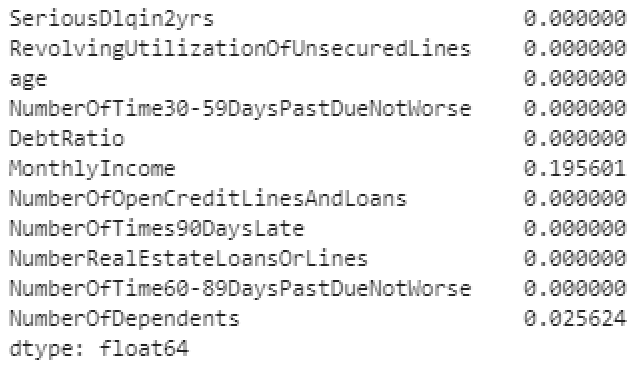
choosing L2 regularization and performs poorly on the unknown datasets, L1 regularization can be considered.

4.3 Datasets and Feature engineering

The data set is obtained from the daily business data of financial institutions, after data desensitization and integration, and the data dictionary is as shown in Table 4.1.

One problem that can exist with realistic data, especially in the banking industry, is sample duplication. Sometimes it may be input duplicate by accident, sometimes it may be a system entry duplicate, in short, we must remove duplicates.

The second problem to face is the missing values. The features we need to fill in here are *MonthlyIncome* and *NumberOfDependents.* For the *NumberOfDependents* feature, only about 2.5% is missing, which can be considered to be removed directly or filled using the



```
SeriousDlqin2yrs                        0.000000
RevolvingUtilizationOfUnsecuredLines    0.000000
age                                     0.000000
NumberOfTime30-59DaysPastDueNotWorse    0.000000
DebtRatio                               0.000000
MonthlyIncome                           0.195601
NumberOfOpenCreditLinesAndLoans         0.000000
NumberOfTimes90DaysLate                 0.000000
NumberRealEstateLoansOrLines            0.000000
NumberOfTime60-89DaysPastDueNotWorse    0.000000
NumberOfDependents                      0.025624
dtype: float64
```

Fig. 4.2: Percentage of missing values

mean value. We fill the missing dependents with the mean value. *MonthlyIncome* is missing by almost 20%, and we know that *MonthlyIncome* is necessarily an important factor for credit scores, so this feature has to be filled in. Here we can consider using Random Forest to fill in the missing values of this feature. Random forest uses the idea that if A, B, and C can predict Z, then A, C, and Z can be used to fill a missing value for B. For a data with n features, one of the features T has a missing value, we use the feature T as the label, and the other $n-1$ features and the original label form a new feature matrix. Then, for T, the part of it without missing values is our *y_train*, which has both labels and features, and the part with the missing values, only features without labels, is the part we need to predict. The value of feature T that is not missing corresponds to the other $n-1$ features plus the original label: *X_train*. The value of feature T that is not missing: *y_train*. The value of feature T that is missing corresponds to the other $n-1$ features plus the original label: *X_test*. The value of feature T that is missing and we need to predict is *y_test*. This approach works very well for cases where a certain feature is missing but the other features are complete.

4.3.1 Descriptive statistics to handle outliers

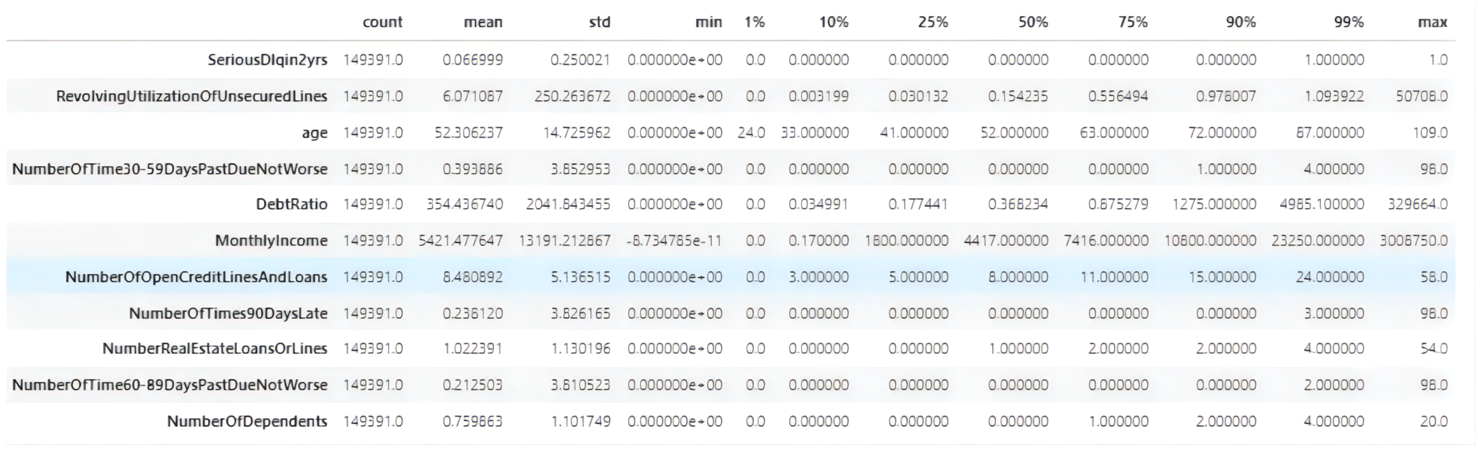
Real data will always have some outliers. We do not want to exclude all outliers, rather many times, the outliers are our focus, for example, the brands with super high purchase volume in a Black Friday sale, or the topics that excite many students in the classroom, these are the ones on which to focus our research and observation.

To deal with outliers on a daily basis, we use box plots or 3δ laws to find outliers. But in the banking data, the outliers we want to exclude are not numbers that are too high or too low, but numbers that do not make sense: for example, income cannot be negative, but a very high level of income is reasonable and can exist. So in the banking industry, we tend to just use ordinary descriptive statistics to see whether the data are abnormal. Note that this approach can only be performed with a limited number of features. If there are hundreds of features and it is not possible to successfully reduce the dimensionality or the feature selection does not work, it is better to use 3δ laws.

As can be seen from the descriptive statistics table above, the *age* characteristic has a minimum value of 0, which is clearly not in line with banking requirements, even for children's accounts, which must be at least 8 years old. If a person's age is found to be 0, it can be judged to be definitely caused by an entry error, and can be treated as a missing value and directly deleted from this sample.

In addition, three indicators seem odd: *NumberOfTime30-59DaysPastDueNotWorse*, *NumberOfTime60-89DaysPastDueNotWorse*, and *NumberOfTimes90DaysLate* which are "the number of times in the past two years that were 35-59 days past due but did not develop worse", "the number of times in the past two years that were 60-89 days past due but did not develop worse", and "the number of times in the past two years that were 90 days past due", respectively. These three indicators are 2 in 99% of the distribution, but the maximum value is 98, which seems very strange. For instance, how could a person be overdue 35-59 days 98 times in the past two years? There are 225 samples where this is the case, and we can basically determine that these samples are some kind of anomalies and they should be removed.

In the descriptive statistics, we can observe that the data are not uniform in magnitude, and there is an extremely skewed distribution. Although logistic regression has no distribution requirement for the data, we know that gradient descent can converge faster if the data obey normal distribution. However, here, we do not standardize the data and do not



	count	mean	std	min	1%	10%	25%	50%	75%	90%	99%	max
SeriousDlqin2yrs	149391.0	0.066999	0.250021	0.000000e+00	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.0
RevolvingUtilizationOfUnsecuredLines	149391.0	6.071087	250.263672	0.000000e+00	0.0	0.003199	0.030132	0.154235	0.556494	0.976007	1.093922	50708.0
age	149391.0	52.306237	14.725962	0.000000e+00	24.0	33.000000	41.000000	52.000000	63.000000	72.000000	87.000000	109.0
NumberOfTime30-59DaysPastDueNotWorse	149391.0	0.393886	3.852953	0.000000e+00	0.0	0.000000	0.000000	0.000000	0.000000	1.000000	4.000000	98.0
DebtRatio	149391.0	354.436740	2041.843455	0.000000e+00	0.0	0.034991	0.177441	0.368234	0.875279	1275.000000	4985.100000	329664.0
MonthlyIncome	149391.0	5421.477647	13191.212867	-8.734785e-11	0.0	0.170000	1800.000000	4417.000000	7416.000000	10800.000000	23250.000000	3008750.0
NumberOfOpenCreditLinesAndLoans	149391.0	8.480892	5.136515	0.000000e+00	0.0	3.000000	5.000000	8.000000	11.000000	15.000000	24.000000	58.0
NumberOfTimes90DaysLate	149391.0	0.238120	3.826165	0.000000e+00	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	3.000000	98.0
NumberRealEstateLoansOrLines	149391.0	1.022391	1.130196	0.000000e+00	0.0	0.000000	0.000000	1.000000	2.000000	2.000000	4.000000	54.0
NumberOfTime60-89DaysPastDueNotWorse	149391.0	0.212503	3.810523	0.000000e+00	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	98.0
NumberOfDependents	149391.0	0.759863	1.101749	0.000000e+00	0.0	0.000000	0.000000	0.000000	1.000000	2.000000	4.000000	20.0

Fig. 4.3: Descriptive statistics of the datasets

perform But here, we do not standardize the data, nor do we standardize the magnitude, why?

Whatever the algorithm has to say, whatever the requirements in statistics, our ultimate goal is to serve the business. Now we have to create scorecards. A scorecard is a card that is used by business people to rate new customers based on various information they fill in. We need to make a "grade" for our data, for example, age 20-30 is a grade, age 30-50 is a grade, monthly income above 10,000 is a grade, 5,000-1,000 is a grade.The score of each grade is different. Once we standardize the data with a uniform magnitude, or after standardization, the data size and range will change, and it may be difficult for business people to understand what it means to have a standardized age between .00328 and .00467 as a grade. And, the information filled in by new customers is inherently inconsistent in magnitude. We can indeed enter all the information, standardize it, and then import it into the algorithm to calculate it. But ultimately, when it comes to the hands of the business people to judge. Due to business requirements, when making the scorecard, we should try to keep the data in its original form, age is the number from 8 to 110, income is a number range from 0 to some very large number, even if the scale is not uniform, we do not standardize the data.

4.3.2 Sample imbalance problem

Some samples are *imbalanced*. For instance, while everyone is trying to prevent credit risk, not many people are actually defaulting on their loans. And, banks don't really refuse all the people who will default. Many people will pay back the money, they just forget the due day. Many people are reluctant to owe money, but they are really struggling at the time and can't turn around their money, so overdue payments happen. But once he has the money, he will pay it back. For a bank, it is willing to lend money to you as long as the borrower can pay it back in the end because it gets income (interest) for lending the money out. So, for banks, what they really want to be identified is the number of people who default in bad faith. But the number of those people is tiny, hence the sample is unbalanced. This has always been a pain point in banking modeling; we always want to capture the minority category.

The most used balancing method in logistic regression is upsampling, which we use to balance the samples.

4.4 Creating bins

To create a score card is to grade the individual features so that the salesperson can score the customer based on the information filled out by the new customer score. Therefore, an important step in the process of creating a scorecard is to divide the bins. It can be said that dividing bins is the most difficult and the core idea of the score card. The essence of dividing bins is to discrete continuous variables, so that people with different attributes can be divided into different categories (with different scores), in fact, the essence is more similar to clustering. We have to answer two questions in dividing bins: First, how many bins are appropriate to divide? Second, what kind of effect should be achieved by splitting the bin?

The number of bins is not known a priori, but since we are discretizing continuous variables, the number of bins must not be too large, and it is better to keep it under ten. For making score cards, it is better to have 4-5 bins. Making a continuous variable discrete must be accompanied by loss of information, and the fewer the bins, the greater the information loss. In order to measure the amount of information on the features and the contribution of the features to the prediction function, the banking industry has defined the concept of *Information Value* (IV).

$$IV = \sum_{i=1}^{N} (good\% - bad\%) \times WOE_i$$

In the above equation N is the number of bins on this feature, i represents each bin, $good\%$ is the proportion of good customers (those with label 0) in this bin to all good customers in the whole feature, $bad\%$ is the proportion of bad customers (that is, those who will default, with label 1) in this bin to all bad customers in the feature, and WOE_i is given by the

IV	**Contribution of features to the prediction function**
<0.03	Features have little to no valid information and contribute nothing to the model, such features can be removed
0.03˜0.09	Little valid information and low contribution to the model
0.1˜0.29	Average valid information, moderate contribution to the model
0.3˜0.49	More valid information and higher contribution to the model
>=0.5	The valid information is very high, and the contribution to the model is super high and suspicious

Table 4.2: Relationship between IV values and the contribution of features to the model

following equation.

$$WOE_i = \ln(\frac{good\%}{bad\%})$$

WOE is an indicator used in the banking industry to measure the probability of default. The essence of this is the logarithm of the ratio of good customers to bad customers; WOE is for a bin, the larger the WOE, the more good customers are in the bin. While IV is for the whole feature, IV represents the meaning of the amount of information on our feature and the contribution of this feature to the model, controlled by Table 4.2.

As can be seen, the larger the IV is not the better. We should find a balance between the value of the IV and the number of bins. The more bins there are, the smaller the IV is bound to be, because the information loss will be very high. Therefore, we will sub-bin the features, and then calculate the WOE value of each feature at every bin, and use the curve of IV values to find the appropriate number of sub-bin.

We want people with different attributes to have different scores, so we want the attributes of people in the same bin to be as similar as possible, and the attributes of people in different bins to be as different as possible, that is, the industry often says "large differences between groups, small differences within groups".For the scorecard, it means that we want the probability of default to be similar for people within a bin, and the probability of default to vary widely across bins, which means that the WOE gap should be large and the proportion of bad customers () in each bin should be different. Then we can use the *chi-square* test to compare the similarity between the two bins, if the *p-value* of the *chi-square* test between the two bins is very large, then they are very similar, then we can combine these two bins into one bin.

Based on this idea, we summarize the steps we take to divide a feature into different bins.

1. We first divide the continuous variables into a larger number of sub-type variables, for example, dividing tens of thousands of samples into 100 groups, or 50 groups.

2. Make sure that samples from both categories are included in each group, otherwise the IV values can not be calculated.

3. We perform a *chi-squar*e test on adjacent groups, and groups with large *p-values* from the *chi-square* test are combined until the number of groups in the data is less than the set N bins.

4. We let a feature be divided into twenty bins, observe how the IV value changes under each number of bins, and find the most suitable number of bins.

5. After dividing the boxes, we calculate the WOE and the *bad%* value of each bin, and observe the effect of dividing the bins.

After all these steps are completed, we can pick features by binning each feature and then observing the IV value of each feature.

To create equal frequency bins we use a quantile-based binning function. The quantile-based binning function is essentially a discretization of continuous variables. It is capable of handling one-dimensional data only and returns the upper and lower bounds of the box. In order to be able to calculate the IV values, we have to make sure that there are *good* (*label 1)* and *bad* (*label 0*) samples in each of the bins.

Not all features can use this binning function, for example, some features, like the *NumberOfDependents* cannot be binned into 20 groups. So we separate the features that can be binned into separate groups, and the features that cannot be automatically binned into their own groups are observed and manually binned.

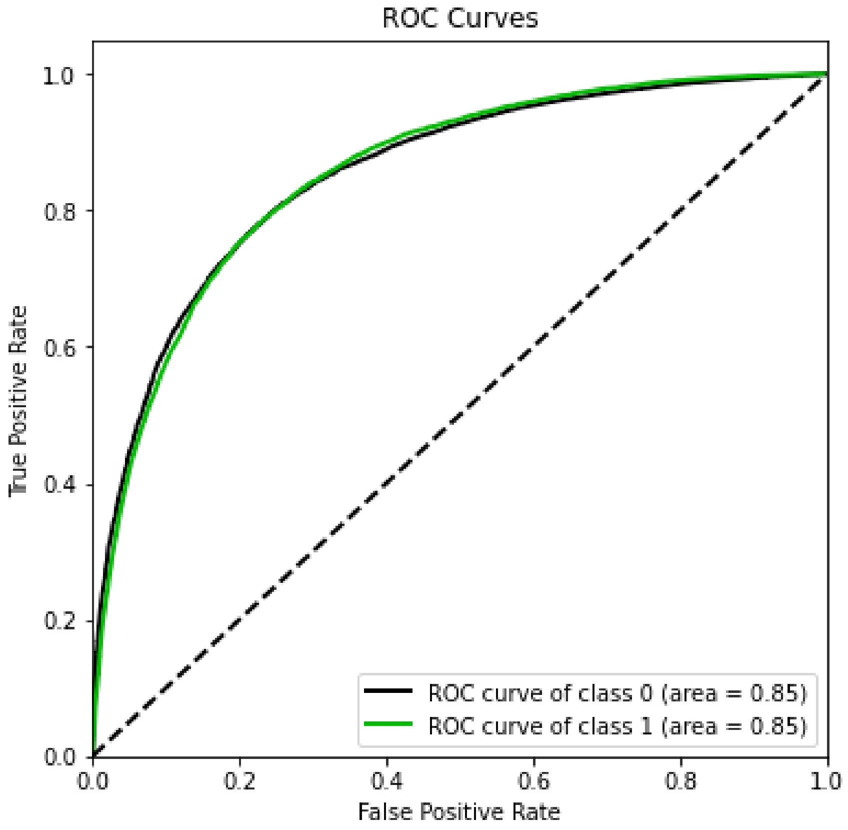


Fig. 4.4: The output of ROC curve

After binning, the next thing we want to do is to calculate the *WOE* for each bin and replace the *WOE* with our original data since we will be using the *WOE* overlay data to model the boxes, we want to get the classification results for each bin, which is the classification results for each rating item on the scorecard.

4.5 Modeling and Model Validation

In the case where there are already bins, the processing of the test set is very simple, we only need to map the already computed *WOE* to the test set.

After the modeling is completed, we use the ROC [24] curve to validate the model. The output of ROC curve is shown in Figure 4.4.

4.6 Making scorecards

After modeling, we verified the predictive power of the model using accuracy and ROC curves. The next step is to convert the logistic regression into a standard scorecard. The scores in the score card are calculated by the following formula.

$$Score = A - B \times \log(odds)$$

where A and B are constants, A is called "compensation", B is called "scale", and $\log(odds)$ represents the probability of a person defaulting. In fact, the logistic regression results in the form of log odds will get $\theta^T x$, that is, our parameter characteristic matrix, so in fact $\log(odds)$ is our parameters. The two constants can be found by bringing the scores of two assumptions into the formula as follows.

1. Expected score for a particular probability of default

2. Specified fraction of doubled probability of default (PDO)

Assuming that the specific score set at log odds of $\frac{1}{60}$ is 600 and PDO = 20, then the score at log odds of $\frac{1}{30}$ is 620. Bringing in the above linear expression, we get.

$$600 = A - B \times \log(\frac{1}{60})$$

$$620 = A - B \times \log(\frac{1}{30})$$

Once the program is run, we can get the scoring grid shown in Table 4.3. Once this form is generated, it can be handed over to a salesperson or entered into an online scoring system to score the customer's credit.

base_score	**[481.97097754]**
age	**Score**
($-\infty$, 43.0]	-3.665845531
(43.0, 54.0]	-1.810777965
(54.0, 61.0]	1.901590047
(61.0, 74.0]	8.185424582
(74.0, ∞]	13.71242121
RevolvingUtilizationOfUnsecuredLines	**Score**
($-\infty$, 0.0987]	46.87958301
(0.0987, 0.297]	14.26470276
(0.297, 0.464]	-2.571738342
(0.464, 0.984]	-22.86940576
(0.984, 1.0]	-9.966645567
(1.0, ∞]	-43.96356247
DebtRatio	**Score**
($-\infty$, 0.0175]	20.03214998
(0.0175, 0.402]	0.476293894
(0.402, 1.485]	-5.102614672
(1.485, ∞]	2.279750114
MonthlyIncome	**Score**
($-\infty$, 0.45]	15.3832127
(0.45, 6165.555]	-3.94588206
(6165.555, ∞]	3.324572836
NumberOfOpenCreditLinesAndLoans	**Score**
($-\infty$, 1.0]	-4.90083415
(1.0, 3.0]	-1.849819887
(3.0, 5.0]	-0.292095707
(5.0, 17.0]	0.723184691
(17.0, ∞]	2.503375994
NumberOfTime30-59DaysPastDueNotWorse	**Score**
($-\infty$, 0.0]	4.954982724
(0.0, 1.0]	-12.21854402
(1.0, 2.0]	-19.3844553
(2.0, ∞]	-21.68284915
NumberOfTimes90DaysLate	**Score**
($-\infty$, 0.0]	3.436794926
(0.0, 1.0]	-25.49014445
(1.0, 2.0]	-32.95065569
(2.0, ∞]	-35.09913698
NumberRealEstateLoansOrLines	**Score**
($-\infty$, 0.0]	-6.866739983
(0.0, 1.0]	3.456113181
(1.0, 2.0]	10.61364782
(2.0, 4.0]	6.804299046
(4.0, ∞]	-5.348785486
NumberOfTime60-89DaysPastDueNotWorse	**Score**
($-\infty$, 0.0]	0.899457188
(0.0, 1.0]	-10.00793895
(1.0, 2.0]	-12.71662421
(2.0, ∞]	-13.06033089
NumberOfDependents	**Score**
($-\infty$, 0.0]	12.75165972
(0.0, 1.0]	-9.951302125
(1.0, 2.0]	-10.4749613
(2.0, ∞]	-9.278451218

Table 4.3: Final output scorecard

CHAPTER 5
DEALING WITH DATA IMBALANCE

In many real-world scenarios, data has *unbalanced* category distribution. That is, one category in the data contains much more data than the other categories [25]. In the loan scenario, we focus on the category imbalance problem in binary classification since a creditworthy customer often has far more data than a fraudulent customer.

Consider a simple example, 100,000 positive samples (normal customers labeled 0) with 1,000 negative samples (fraudulent customers labeled 1), the ratio of positive to negative samples is 100:1, if brought directly into the model to learn, each gradient descent if using the full amount of samples, the weight of negative samples is less than 1/100, even if the information of negative samples is not learned at all, the accuracy rate is more than 99%. So obviously we must not measure the effectiveness of the model in terms of accuracy. But in practice, we actually know that even if we use KS or AUC to measure the performance of the model, there is still no guarantee that the model can learn the negative samples well. Instead, we actually need to get a classifier that has a high accuracy for the positive cases without affecting the accuracy of the negative cases.

In financial risk control business scenarios there are three primary ways to handle unbalanced data sets.

1. Use human experience - Force data balancing approach based on business experience.
2. Use an algorithm - From the perspective of the algorithm, the algorithm is optimized considering the variability of the cost of different misclassification cases, mainly based on Cost-Sensitive-Learning (CSL) algorithm.
3. Use (non-random) sampling - Sampling can be used to skew the dataset.

We consider each scenario in detail below.

5.1 Handling Imbalance

Forcing data balancing mainly involves artificially modifying the sample distributions in the data based on business experience. The advantage of this method is that it is simple and easy to be understood by business staff. The disadvantage is also obvious, that is, it lacks a theoretical basis for support and relies heavily on the business experience of data processors. It tends to work well on some data sets, but performs poorly on others, and tends to lead to over-fitting.

Downward probing is the most direct solution to the imbalance of samples in the risk control scenario. The so-called downward probing is to lend to people with lower scores who are rejected, sacrificing part of the revenue to accumulate bad samples for subsequent model learning. This is also the most direct and effective of all methods. But not every company is willing to take on this part of bad debt. In addition, as we mentioned before, the samples used in subsequent model iterations are biased as the business develops, and downward probing can solve this problem as well.

In semi-supervised learning the data of rejected customers are gradually generated by semi-supervised methods to generate labels and then brought into the model for training. The more typical sub-methods are described below.

- Rejection deduction - Also called rejection inference, is a method of sampling the percentage of low scoring customers based on experience. For example, fifty percent of the lowest scoring customers are considered bad guys, followed by forty percent, etc. The effect is not as good as down-probing. But does not have any additional overhead.

- Violent semi-supervision - A more crude approach is to exhaustively enumerate each way the sample is labeled and bring it into the model to see if it helps the model. This approach is less efficient and prone to over-fitting.

- Model Screening - The unlabeled samples are labeled with other models that have been trained, and then the model is trained. Many companies will use the current model to make predictions on top, and then bring in the model to continue training. This is highly discouraged and the results are generally very poor. Consider unsupervised algorithms or using very old samples for training and then making predictions.

[TODO - describe which method is used in this thesis of the three methods given above.]

5.2 Treatment of imbalanced data sets from an algorithmic perspective

Typical of this type of approach is cost-sensitive-learning. The study of algorithms for cost-sensitive-learning can be divided into three categories according to the solution to the problem.

1. The first category is *direct construction*. This class of approaches focuses on how to directly construct a cost-sensitive learning model, and researchers have proposed different solutions for different classifier models, which include:

    - Decision trees: Knoll [26] and Bradford [27] proposed a cost-sensitive pruning method for decision trees. Bradford studied how to prune decision trees under cost-sensitive conditions to minimize the loss, and showed that the pruning method based on Laplace's method could achieve the best results. Holte investigated node splitting for decision trees with cost-sensitive-learning method [28]
    - Boosting: Fan and others [29] have proposed the cost-sensitive Boosting algorithm Ada-Cost.
    - Neural Networks:Geibel and Wysotzki [30] proposed a cost-sensitive learning approach based on the Perceptron classification algorithm, in which the authors propose cost-sensitive parameter update rules for non-separable classes. For example, Kukar and Kononenko [31] propose a new backward propagation algorithm for neural networks for neural networks to enable them to meet the requirements of cost-sensitive learning.

2. The second category is cost-sensitive learning based on *post-processing of classification results*. This class of methods involves learning a classification model according

to traditional learning methods and then adjusting the results of its classification according to Bayesian risk theory in order to minimize losses. Compared to the first class of cost-sensitive learning methods, this approach has the advantage that it does not depend on the specific classifier used. Domingos [32] proposed a process called MetaCost, which treats the underlying classifier as a black box without making any assumptions or changes to the classifier, and MetaCost can be applied to any number of base classifiers and any form of cost matrix.

3. The third category is based on *traditional learning models.* Such methods are trained to obtain cost-sensitive models by changing the distribution of the original training data. Chan and Stolfo [33] proposed a hierarchical model (Stratification) to adjust the unevenly distributed training data into uniformly distributed data with positive and negative examples. Zadrozny [34] propose a cost-proportionate idea to adjust the weights to the training data, which in practice is similar to the Boosting algorithm, and can be implemented by adjusting the weights to the classification model, and by subsampling.

[TODO - describe which method this thesis uses]

5.3 Treatment of imbalanced data sets from a data perspective

This processing method focuses on the imbalance of the data set by using different sampling methods in the data set through statistical methods. This method does not require much modification to the existing algorithm, is simple to implement, and can achieve more desirable application results. Therefore, in this project, this type of method is mainly used to deal with the imbalanced datasets problem.

For imbalanced data, one of the easiest ways is to use *random sampling*, that is, to generate a sample of a small number of classes. There are two main approaches to generating minority classes.

1. Over-sampling. Over-sampling is random sampling from a small number of classes to add new samples.

2. Under-Sampling. In contrast to up-sampling, down-sampling is the process of selecting a small number of samples at random from the majority class samples and merging the original minority class samples into a new training data set.There are two types of random down-sampling: with and without put-back.

The biggest advantage of random sampling is its simplicity, but the disadvantage is also obvious. Some samples will be repeated in the datasets after up-sampling, and the trained model will have some over-fitting; and the obvious disadvantage of down- sampling is that the final training set loses data and the model learns only a part of the overall pattern.

To avoid over-sampling we employ SMOTE [35]. The basic idea of SMOTE is to analyze the minority class samples and add new samples to the datasets manually based on the minority class samples. The algorithm flow is as follows.

1. For each sample in the minority class $x_i \in S_{\min}$ where $S_{\min} \in S$, calculate the distance between that point and the other sample points in the minority class to get the nearest k nearest neighbors (that is, the KNN algorithm for the minority class points)

2. A sampling ratio is set according to the sample imbalance ratio to determine the sampling multiplier, and for each minority class sample x_i, a number of samples are randomly selected from its k nearest neighbors, assuming that the selected nearest neighbor is x'.

3. For each randomly selected nearest neighbor x', a new sample x_{new} is constructed separately from the original sample according to the following formula.

$$x_{\text{new}} = x_i + (x' - x) \times \delta$$

   where $x_i \in S_{\min}$ is a sample of a few classes, x' is a close neighbor of x, and $\delta \in [0, 1]$ is a random number.

The SMOTE method is an oversampling method that overcomes some of the drawbacks of oversampling and enhances the original data. However, the SMOTE algorithm has

	train_ks	**evl_ks**
Before	0.4482453222991063	0.4198642457760936
After	0.4859866821876423	0.44085108654818894

Table 5.1: Comparison of results before and after processing of imbalanced data sets

obvious drawbacks: on the one hand, it increases the possibility of overlap between classes (since a new sample is generated for each minority class sample, it is prone to generate sample overlap), and on the other hand, it generates some samples that do not provide useful information.

We improve on SMOTE by utilizing Borderline-SMOTE [36]. The difference between Borderline-SMOTE and SMOTE is that the original SMOTE generates new samples for all minority class samples. The improved method, on the other hand, first determines the borderline samples of a few classes according to the rules, and then generates new samples for these samples.

A simple rule for determining the borderline is that the minority class with more than half majority class samples in the K-nearest neighbors is the border sample. Intuitively, new samples are generated only for those minority class samples that are surrounded by mostly majority class samples.

Suppose $x_i \in S_{\min}$ is a sample in the minority class, at this time the samples in the minority class are divided into three categories.

1. NOISE - All nearest neighbor samples of that minority class are from other classes different from sample x_i.
2. DANGER - At least half of the nearest neighbor samples are from the same category (different from the category of x_i).
3. SAFE - All nearest neighbor samples are from the same class.

5.4 Comparison of results

Table 5.1 shows the results. As can be seen, there is a 3.5 percentage point improvement on the final cross-time validation set. The about 4 percentage point improvement on the

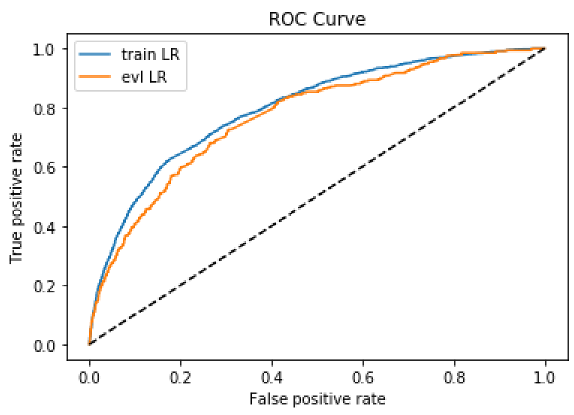


Fig. 5.1: The ROC Curve before processing

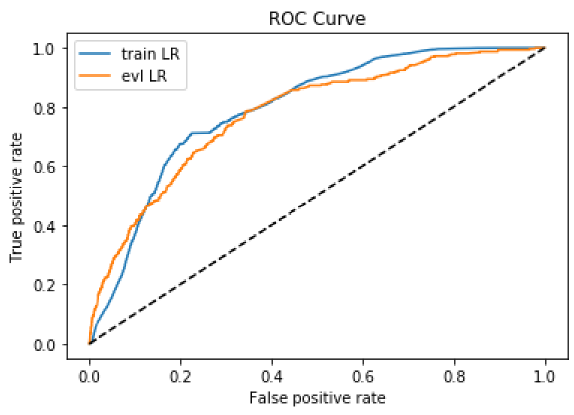


Fig. 5.2: ROC curve after oversampling with SMOTE algorithm

training set is more in line with expectations. However, as can be seen from the ROC curves in Figures 5.1 and 5.2, after oversampling with the SMOTE algorithm, the performance of the model increases on both the training and test sets while the performance of the model improves. That is, the performance improvement on the training set is better than that on the test set, which may bring the risk of overfitting, but this risk is within the manageable range.

CHAPTER 6
PERFORMANCE IMPROVEMENT

In this chapter we present how the models were improved using two methods: building an ensemble model and employing a gradient boost decision tree (GBDT).

6.1 Ensemble Models

An ensemble model is a compilation of several independent algorithms whose predictions are gathered to work as a whole [37]. There are several ways to create an ensemble. In this section we consider several methods: simple Voting/Averaging (for classification and regression problems respectively), Stacking, Boosting and Bagging [38]

Voting or *averaging* the results predicted by each different model directly, without changing the model, is a simple but effective way of ensemble. For example, for the classification problem, suppose there are three mutually independent models, each with a 70% correct rate, and a minority-majority voting is used. Then the final correct rate will be given by the following equation.

$$0.7^3 \times 0.3 \times 3 = 0.784$$

That is, the results were simply voted on, resulting in an 8% improvement in the correctness rate. This is a simple probabilistic problem - if more models are voted on, then obviously the results will be better. But the prerequisite is that the models are independent of each other and the results are not correlated with each other. The more similar the models are fused, the worse the ensemble will be.

Another ensemble method is *boosting* [39]. Boosting is an integrated learning approach that concatenates various weak classifiers in series, where the training of each classifier depends on the results of the previous classifier. Boosting is shown in Figure 6.1. As with all fusion approaches, it does not consider what the structure of the individual weak

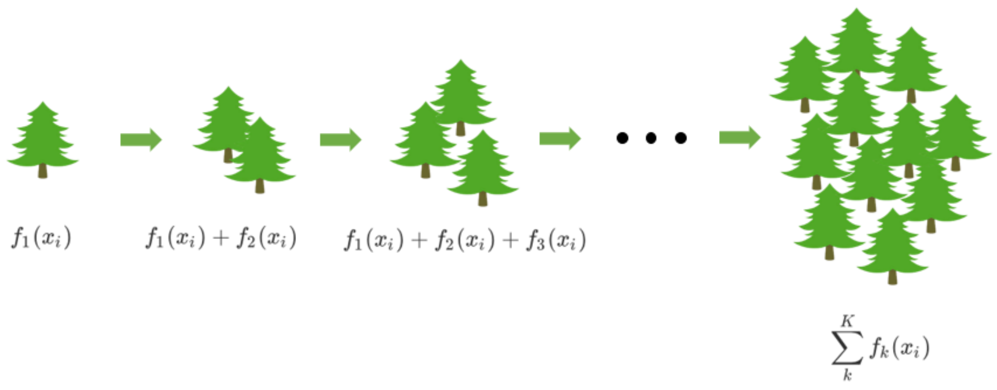


Fig. 6.1: General steps for building a Boosting ensemble model

	Individual Model		Ensemble Model		
	Decision Tree	SVM	Voting	AdaBoost	Random Forest
Cross_val_score	0.708343	0.751235	0.765589	0.760377	0.768196

Table 6.1: Comparison of the results of three model ensemble methods

classifier models themselves are, but rather manipulates the training data (sample set) and the connection method to obtain smaller errors. However, in order to equalize the errors of the final strong classifiers, the previously selected classifiers are generally relatively weak ones, because once a classifier is stronger it will make the subsequent results too affected. Common Boosting methods are Adaboost, GBDT, and XGBOOST [40].

Bagging (also known as *Bootstrap Aggregation*) [41] also does not operate on the model itself, but acts on the sample set. It uses a randomized and put-back selection of training data then constructs the classifiers and finally performs the combination. Unlike the Boosting method in which the classifiers are interdependent and run serially, the Bagging method in which the base learners do not have strong dependencies on each other and are generated simultaneously and run in parallel.

To demonstrate the impact of an ensemble model, we created an ensemble model from a Decision Tree and SVM using Voting, AdaBoost, and Random Forest. Table 6.1 shows the result. Each ensemble method improved the accuracy over that of the individual classifiers.

6.2 Gradient Boost Decision Tree

Gradient Boost Decision Tree (GDBT) is a commonly used nonlinear model [42, 43]. It is based on the idea of boosting in ensemble learning. Each iteration creates a new decision tree in the direction of reducing the gradient of the residuals, and as many iterations as it takes to generate a decision tree. A GBDT has the advantage of finding multiple distinguishing features and feature combinations, and the paths of the decision tree can be used directly as Logistic Regression (LR) input features, eliminating the need to manually find features and feature combinations. This way of generating LR features by GBDT (GBDT+LR), which has been practiced in the industry (Facebook [44]).

In GBDT+LR GBDT is used to extract features from the training set as new training input data, and LR is used as a classifier for the new training input data in the following three steps.

1. GBDT first does training on the original training data to get a binary classifier, and it also needs to use grid search to find the best combination of parameters.

2. In contrast to the usual practice, when GBDT is trained to make predictions, the output is not the final binary probability value, but the leaf node position belonging to the predicted probability value calculated for each tree in the model is to be recorded as 1. In this way, a new training data is constructed. In constructing new training data with GBDT, it is the One-hot method that is used. In a GBDT with n weak classifiers and m leaf nodes, each training data is transformed into a $1 * m$-dimensional sparse vector with n elements being 1 and the remaining $m - n$ elements being 0.

3. After the new training data is constructed, the next step is to input the label (output) data from the original training data into the Logistic Regression classifier for training the final classifier.

The procedure is diagrammed in Figure 6.2. Tree1(The light blue nodes in the image) and Tree2(The red nodes in the image) are two trees learned by GBDT model, x(The black node in the image) is an input sample, after traversing the two trees, x samples fall on

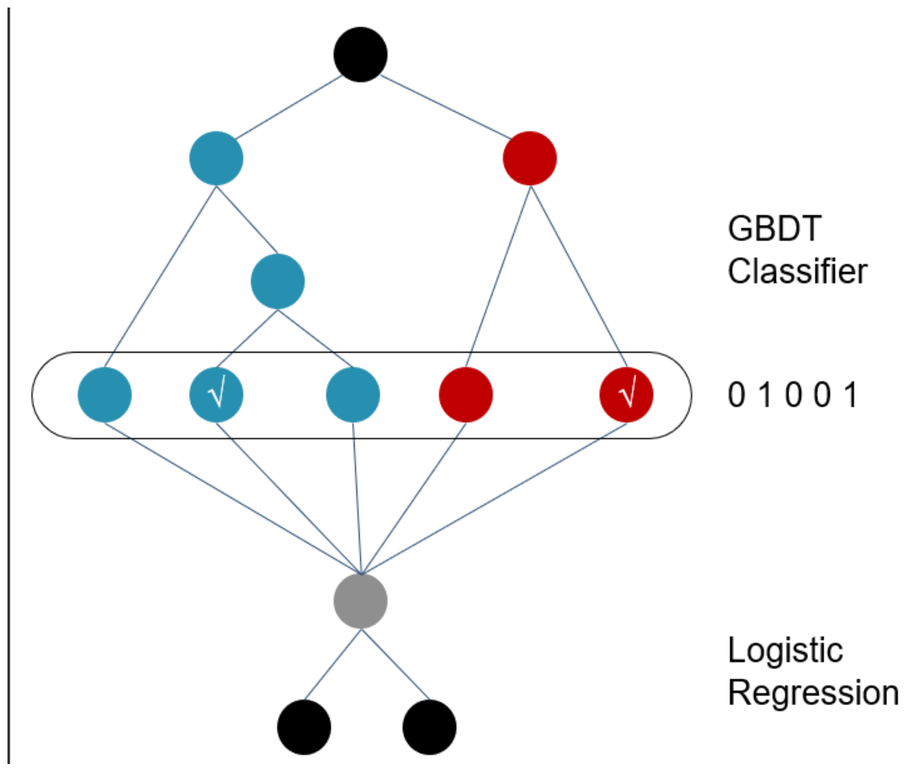


Fig. 6.2: The procedure of GBDT ensemble LR

	train_ks	**evl_ks**
Before	0.4482453222991063	0.4198642457760936
LGB+LR	0.4687230745337274	0.44510149222090417

Table 6.2: Results Comparison Table

the leaf nodes(The light blue nodes and red nodes) of the two trees respectively, each leaf node corresponds to LR one-dimensional features, then by traversing the tree, all LR features corresponding to the sample are obtained. Since each path of the tree, which is a distinguished path finally partitioned by minimizing the mean squared deviation and other methods, the features and feature combinations obtained according to the path are relatively distinguished, and the effect is theoretically not inferior to the manual empirical processing method.

6.2.1 Comparison and analysis of results

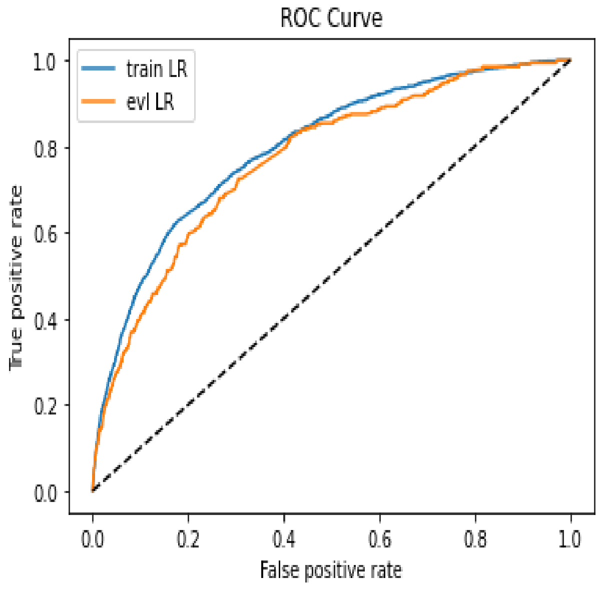


Fig. 6.3: The The ROC Curve before model ensemble

The Figure 6.3 and Figure 6.4 and comparison Table 6.2 show that with model ensemble, the performance of the model improves by about 3% on both the training set and the test set data. Also the difference in performance between the training and test sets is shrinking. This shows that the model fusion technique also plays a role in improving the model effect and reducing overfitting.

Combined with the results achieved by using the Borderline-SMOTE algorithm on the unbalanced dataset in the previous section, the use of model fusion techniques is also a way to be considered in order to improve the training effect of the financial risk control training model, in addition to the corresponding processing measures on the unbalanced dataset. And since the financial risk control dataset is inherently characterized by data imbalance, it can be suggested that processing of imbalanced data can be preferred, and if the effect is still poor after processing, then the use of model fusion can be considered for performance improvement.

At the same time, because GDBT is faster and suitable for online analysis scenarios, if there is a demand for online analysis, you can first consider using GDBT+LR for performance improvement.

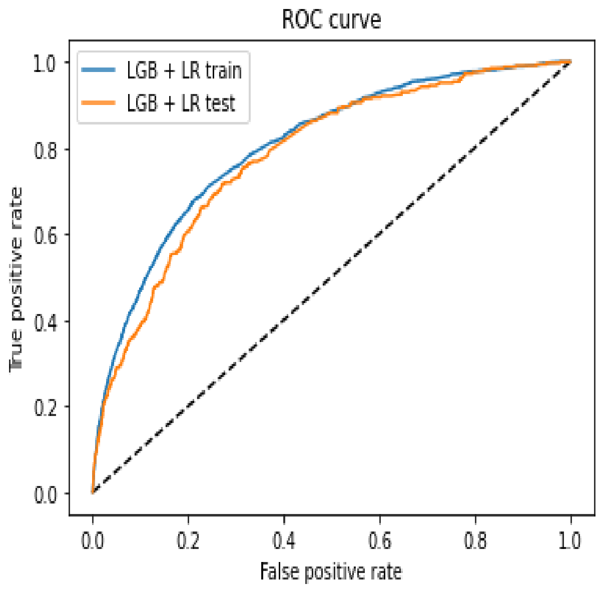


Fig. 6.4: The The ROC Curve after model ensemble

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

This thesis presents a financial risk control system build using machine learning methods. The system uses daily business data from financial institutions and some data provided by third parties. The system was deployed in the financial industry and achieved good results in internal testing. After a period of testing and optimization, the project will be officially launched in early 2022, which will improve the efficiency and accuracy of the daily loan business of the company. The system has the following features.

- Developed a decision tree based loan review system, which performs the review of loan applications by predicting the bad debt rate of users. For the data set selection of this model, we did not use traditional personal credit data, but used oil consumption data, which is closely related to the applicants' daily life, as the basis of modeling. Various data pre-processing strategies were also used to clean the dataset. In response to the actual situation that the original dataset had few effective features, feature derivation was carried out, and the original 18 features were derived to 70 effective features, which greatly improved the classification effect of the model. While the bad debt rate is reduced, the pass rate is significantly improved.

- Developed a credit score card system for individual users. The traditional credit scoring system is mainly based on credit data provided by third-party credit bureaus (FICO). However, this data can only be used for credit scoring based on the user's historical consumption data and income status, which has the disadvantage of slow information update and cannot fully reflect the user's credit status. This scoring system combines personal information data and consumption data on top of credit data, and uses logistic regression algorithm to perform credit scoring. In the data set processing, for the characteristics of logistic regression algorithm, the continuous type data is converted into discrete type data by using split-box technology. The binning method creatively uses equal frequency binning + calculation of WOE value, which achieves good results. The score card is also generated for the use of operational staff. The model has the advantages of strong interpretability, timeliness and high accuracy.

- For data imbalance, Borderline SMOTE based on SMOTE algorithm is used to balance.

- Based on the comprehensive comparison of several model fusion algorithms, the GBDT+LR-based model fusion algorithm is used to improve the overall performance of the model.

In future, the following improvements will be made around enhancing performance and improving functionality

The current infrastructure of this system is built based on AWS public cloud system, in which AWS's EC2, S3, Aurona, DynamoDB, RDS and other services are mainly used to obtain the required computing and storage capabilities. However, as the project progressed, the amount of data acquired and required for daily processing became larger and larger. Therefore, we plan to build our own OLAP eco-system due to the cost, performance, data sensitivity and data security considerations. The design is shown in Figure 7.1

In this system, Hadoop distributed file system HDFS and Spark distributed computing framework are used to solve the problem of large data sets and improve the security of the data.

- Using the distributed storage HDFS to avoid data loss and and provide computing resource.

- Using Zookeeper to achieve high availability and stability of the distributive computing platform, with the ability to tolerate network attack.

- Building a hybrid warehouse system using Hive and Hbase to facilitate faster data query (using Hive) and distributive storage of T-byte data (using Hbase) .

- Using Apache Sqoop to convert the relational database to Hadoop data.

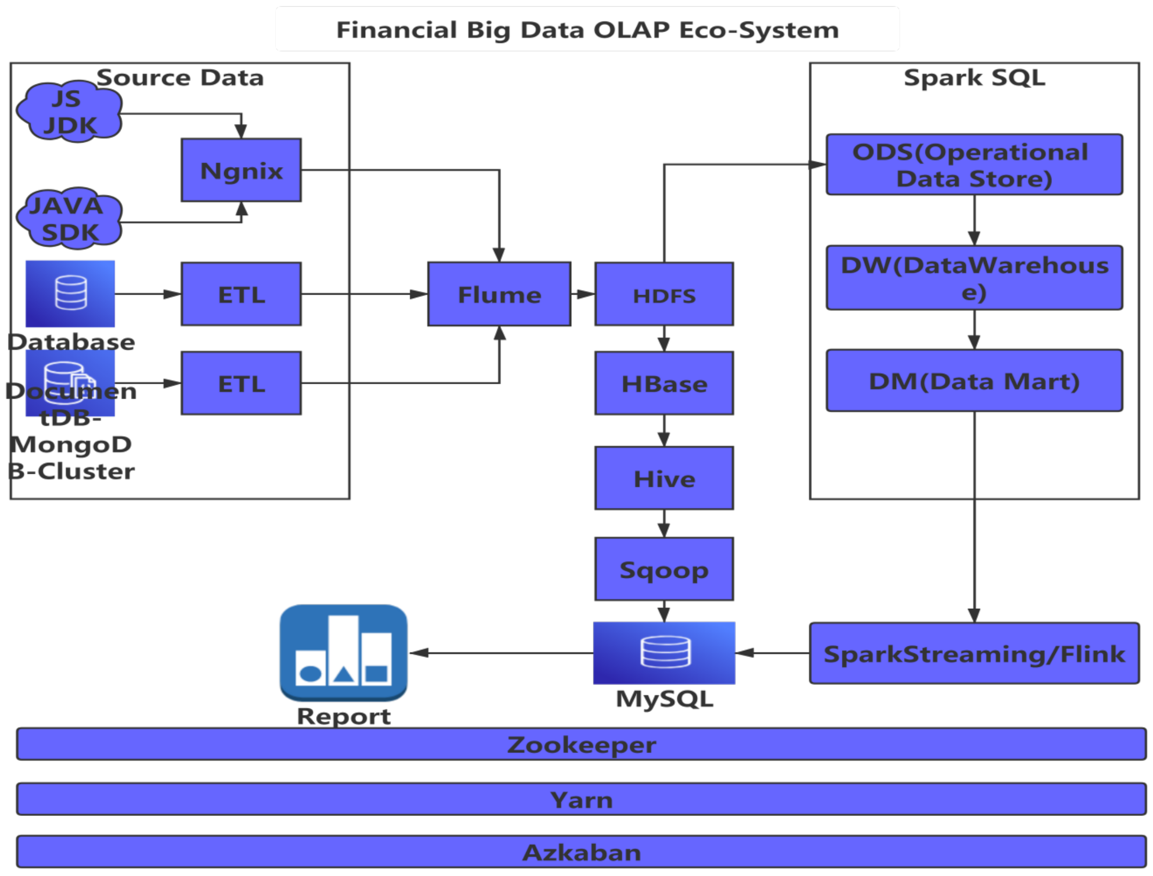


Fig. 7.1: The Design of Financial Big Data OLAP Eco-System

- Building MongoDB clusters and implement data migration from the relational database to MongoDB
- Building an Apache Flume-based system for multi-source data collection with high efficiency and organization

Almost all financial institutions have a need for anti-fraud. And fraud is becoming more and more sophisticated, with each different type of fraud treated as a separate category. In addition to the diverse and continuously changing fraud schemes, fraud detection generally faces the following problems

- In most cases, the data is not labeled, and there is no use for various mature supervised learning.

- It is very difficult to distinguish between noise and anomaly, and even requires a little imagination and intuition.
- When multiple scam data are mixed together, it is more difficult to distinguish between different scam types. The reason for this is that we don't know the definition of each type of fraud.
- Even if we do have historical data on frauds, that is, using supervised learning with labels, there is a significant risk. A model learned with such historical data can only detect scams that have appeared similar to historical scams, while our model will be powerless for variants of scams and scams that have never been seen before.

We plan to use unsupervised learning to build and model and have domain experts to validate our predictions and provide feedback so that the model can be adjusted in a timely manner.

Bibliography

[1] A. Mashrur, W. Luo, N. A. Zaidi, and A. Robles-Kelly, “Machine learning for financial risk management: A survey,” *IEEE Access*, vol. 8, pp. 203 203–203 223, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3036322

[2] P. O. Kelliher, D. Wilmot, J. Vij, and P. J. Klumpes, “A common risk classification system for the actuarial profession,” *British Actuarial Journal*, vol. 18, no. 1, pp. 91–121, 2013.

[3] M. Beyhaghi and J. P. Hawley, “Modern portfolio theory and risk management: assumptions and unintended consequences,” *Journal of Sustainable Finance & Investment*, vol. 3, no. 1, pp. 17–37, 2013.

[4] W.-Y. Lin, Y.-H. Hu, and C.-F. Tsai, “Machine learning in financial crisis prediction: a survey,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 421–436, 2011.

[5] T. Mitchell, “Machine learning,” 1997.

[6] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning.* MIT press, 2018.

[7] S. Anglekar, U. Chaudhari, A. Chitanvis, and R. Shankarmani, “Machine learning based risk assessment analysis for smes loan grant,” in *2021 International Conference on Communication information and Computing Technology (ICCICT).* IEEE, 2021, pp. 1–5.

[8] Y. Yu, “The application of machine learning algorithms in credit card default prediction,” in *2020 International Conference on Computing and Data Science (CDS).* IEEE, 2020, pp. 212–218.

[9] G. Kou, Y. Peng, and G. Wang, "Evaluation of clustering algorithms for financial risk analysis using mcdm methods," *Information Sciences*, vol. 275, pp. 1–12, 2014.

[10] U. P. Shukla and S. J. Nanda, "Designing of a risk assessment model for issuing credit card using parallel social spider algorithm," *Applied Artificial Intelligence*, vol. 33, no. 3, pp. 191–207, 2019.

[11] C. Sudha and D. Akila, "Credit card fraud detection system based on operational & transaction features using svm and random forest classifiers," in *2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM)*. IEEE, 2021, pp. 133–138.

[12] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 2015, pp. 1–10.

[13] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[14] G. De'ath and K. E. Fabricius, "Classification and regression trees: a powerful yet simple technique for ecological data analysis," *Ecology*, vol. 81, no. 11, pp. 3178–3192, 2000.

[15] W. Li and J. Liao, "An empirical study on credit scoring model for credit card by using data mining technology," in *2011 seventh international conference on computational intelligence and security*. IEEE, 2011, pp. 1279–1282.

[16] B. H. Misheva, P. Giudici, and V. Pediroda, "Network-based models to improve credit scoring accuracy," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2018, pp. 623–630.

[17] J. Ignatius, A. Hatami-Marbini, A. Rahman, L. Dhamotharan, and P. Khoshnevis, "A fuzzy decision support system for credit scoring," *Neural Computing and Applications*, vol. 29, no. 10, pp. 921–937, 2018.

[18] M. C. Mhina and F. Labeau, “Using machine learning algorithms to create a credit scoring model for mobile money users,” in *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 2021, pp. 000 079–000 084.

[19] Z. Zhang, K. Niu, and Y. Liu, “A deep learning based online credit scoring model for p2p lending,” *IEEE Access*, vol. 8, pp. 177 307–177 317, 2020.

[20] F. Shen, R. Wang, and Y. Shen, “A cost-sensitive logistic regression credit scoring model based on multi-objective optimization approach,” *Technological and Economic Development of Economy*, vol. 26, no. 2, pp. 405–429, 2020.

[21] W. H. Eko, M. Heti, and S. I. Teguh, “Improving credit scoring model of mortgage financing with smote methods in sharia banking,” *Russian Journal of Agricultural and Socio-Economic Sciences*, vol. 92, no. 8, 2019.

[22] H. Sutrisno and S. Halim, “Credit scoring refinement using optimized logistic regression,” in *2017 International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT)*. IEEE, 2017, pp. 26–31.

[23] R. Y. Goh, L. S. Lee, H.-V. Seow, and K. Gopal, “Hybrid harmony search–artificial intelligence models in credit scoring,” *Entropy*, vol. 22, no. 9, p. 989, 2020.

[24] W. J. Krzanowski and D. J. Hand, *ROC curves for continuous data.* Crc Press, 2009.

[25] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[26] U. Knoll, G. Nakhaeizadeh, and B. Tausend, “Cost-sensitive pruning of decision trees,” in *European Conference on Machine Learning*. Springer, 1994, pp. 383–386.

[27] J. P. Bradford and J. A. Fortes, “Characterization and parallelization of decision-tree induction,” *Journal of Parallel and Distributed Computing*, vol. 61, no. 3, pp. 322–349, 2001.

[28] R. C. Holte and C. Drummond, "Cost-sensitive classifier evaluation using cost curves," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2008, pp. 26–29.

[29] Z. Fan and M. Ni, "Individualized boosting learning for classification," *Optik*, vol. 126, no. 24, pp. 5733–5739, 2015.

[30] P. Geibel, U. Brefeld, and F. Wysotzki, "Perceptron and svm learning with generalized cost models," *Intelligent Data Analysis*, vol. 8, no. 5, pp. 439–455, 2004.

[31] M. Kukar and I. Kononenko, "Reliable classifications with machine learning," in *European Conference on Machine Learning.* Springer, 2002, pp. 219–231.

[32] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 155–164.

[33] P. K. Chan and S. J. Stolfo, "On the accuracy of meta-learning for scalable data mining," *Journal of Intelligent Information Systems*, vol. 8, no. 1, pp. 5–28, 1997.

[34] B. Zadrozny and C. Elkan, "Learning and making decisions when costs and probabilities are both unknown," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 204–213.

[35] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[36] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing.* Springer, 2005, pp. 878–887.

[37] J. R. Campos, E. Costa, and M. Vieira, "Improving failure prediction by ensembling the decisions of machine learning models: A case study," *IEEE Access*, vol. 7, pp. 177 661–177 674, 2019.

[38] S. Nagi and D. K. Bhattacharyya, "Classification of microarray cancer data using ensemble approach," *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 2, no. 3, pp. 159–173, 2013.

[39] R. E. Schapire and Y. Freund, "Boosting: Foundations and algorithms," *Kybernetes*, 2013.

[40] D. Sarkar and V. Natarajan, *Ensemble Machine Learning Cookbook: Over 35 practical recipes to explore ensemble machine learning techniques using Python.* Packt Publishing Ltd, 2019.

[41] H. Kim and Y. Lim, "Bootstrap aggregated classification for sparse functional data," *Journal of Applied Statistics*, pp. 1–12, 2021.

[42] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[43] ——, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[44] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 2014, pp. 1–9.