



Heterogeneous ensemble learning with feature engineering for default prediction in peer-to-peer lending in China

Wei Li^{1,2} · Shuai Ding^{1,2} · Hao Wang^{1,2} · Yi Chen^{1,2} · Shanlin Yang^{1,2}

Received: 3 July 2018 / Revised: 19 December 2018 / Accepted: 12 March 2019 /

Published online: 19 March 2019

© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

In recent years, peer-to-peer (P2P) lending in China, which is a new form of unsecured financing that uses the Internet, has boomed, but the consequent credit risk problems are inevitable. A key challenge facing P2P lending platforms is accurately predicting the default probability of the borrower of each loan using the default prediction model, which effectively helps the P2P lending platform avoid credit risks. The traditional default prediction model based on machine learning and statistical learning does not meet the needs of P2P lending platforms in terms of default risk prediction because for data-driven P2P lending, credit data have a large number of missing values, are high-dimensional and have class-imbalanced problems, which makes it difficult to effectively train the default risk prediction model. To solve the above problems, this paper proposes a new default risk prediction model based on heterogeneous ensemble learning. Three individual classifiers, extreme gradient boosting (XGBoost), a deep neural network (DNN) and logistic regression (LR), are used simultaneously with a liner weight ensemble strategy. In particular, this model is able to process missing values. After generating discrete and rank features, this model adds missing values to the model for self-training. Then, the hyperparameters are optimized by the XGBoost model to improve the performance of the prediction model. Finally, compared with the benchmark model, the proposed method significantly improves the accuracy of the prediction results. In conclusion, the prediction method proposed in this paper solves the class-imbalanced problem.

Keywords Heterogeneous ensemble learning · Default prediction · Feature engineering · imbalanced data · Hyperparameter optimization

✉ Shuai Ding
dingshuai@hfut.edu.cn

✉ Shanlin Yang
yangsl@hfut.edu.cn

Extended author information available on the last page of the article

1 Introduction

As a typical type of Internet finance in China, P2P lending is a form of personal consumption finance that is made possible by the allocation of funds on the Internet and thus is a novel financing mode. Compared with traditional financing methods, P2P lending matches the supply and demand of funds using the Internet without the participation of traditional financial institutions. Therefore, it is convenient, fast and transparent. P2P lending platforms can assist the development of direct capital connections between investors and borrowers and can ensure that both parties can benefit from this process to the greatest extent. Compared with fixed loan interest rates obtained from banks, P2P lending can reduce the borrowing costs of borrowers due to reduced intermediary fees. Moreover, compared with investors, P2P lending provides convenient channels and high returns. However, due to the above characteristics of P2P lending, traditional default risk prediction methods may not be applicable because the credit data used for P2P lending platforms are mostly obtained from the Internet, which makes them have the characteristics of high dimensionalities and sparse and imbalanced classes. The default risk prediction problem associated with consumer financing, including P2P lending, can be essentially classified as a binary classification problem. In view of this, an accurate prediction model of P2P lending default risk with high dimension, sparseness and category imbalance is the core of P2P lending industry development.

In general, the prediction method used to determine loan default risk includes a statistical method and a machine learning method, which are used to distinguish good loan applicants, namely, those able to repay loans within the specified period [20, 34] from other applicants. Default risk prediction is mainly used to determine the possibility that an applicant will default. Loan default risk predictions can be visually generated by converting the problem into a binary classification issue. Although P2P lending platforms evaluate the credit of borrowers using a complicated assessment mechanism, P2P lenders are more likely to suffer when they issue unsecured loans. In addition, default may occur because the lenders of many platforms lack the ability to engage in credit risk management and may be unable to effectively evaluate the risk level of borrowers when they establish their credit portfolios [10]. Various machine learning-based loan default risk prediction models, e.g., random forests, decision trees, artificial neural networks and kernel methods [28, 44], have been used when lenders decide whether to provide funding for a certain loan and are critical for P2P lending platforms. The assessment of prediction results relies on accuracy-based measurements, for example, overall accuracy, error rates and the area under the receiver operating characteristic curve (AUC), all of which are used to evaluate the performance of the model proposed in this paper [6, 21].

In the real world, the majority of borrowers are “good” customers, and there is only a small portion of “bad” customers, that is, the sample data used for the prediction model have an imbalanced distribution. The so-called imbalanced distribution of the sample data class means that the data are centralized in one or some classes, i.e., there are much more data in this or these classes than there are in other classes. We call the nondefault sample class the majority class, and the default sample class, which has a relatively small amount of data, will be called the minority class. It can thus be seen that prediction errors may occur because minority samples have so little data that the traditional prediction model may sometimes predict them as positive samples. Financial institutions, therefore, will suffer a much greater loss when they mistake a “bad” target customer for a “good” one while predicting the customer’s default behavior than if they mistake a “good” customer for a “bad” one [6, 41].

In previous research, researchers have attempted to optimize prediction performance when using highly imbalanced datasets. One commonly used method is the sampling method, which, however, may change the class distribution of the original training set. Oversampling and undersampling are two sampling techniques, but previous studies on default prediction or credit scoring have tended to use the random sampling method [41]. Another method is improving the algorithm by using techniques such as ensemble learning, and cost-sensitive learning [8, 22, 33], all of which adapt to an imbalance by improving the algorithm, i.e., assigning the various samples with different weight values, adjusting penalty coefficients, changing the probability density, regulating the class boundary, etc. The last strategy is focused on simplifying uncorrelated or redundant features in data to decrease differences, improve the classification accuracy of the model and decrease the time complexity [12].

In addition to high-dimensional and class-imbalanced problems in data, data deficiency can also result in the poor performance of the default prediction model. Therefore, this paper proposes a more accurate default prediction model for P2P lending that can be used for high-dimensional data samples with imbalanced classes and missing values. To improve prediction performance, this paper adopts a new way to address missing values in data processing and feature engineering; this paper uses this method as a kind of feature and introduces it to learning model for self-learning. Next, while we address engineering this new feature, we pretrain the three individual classifiers we select for the benchmark models and compare the results with those obtained through the original feature training. The machine learning model with the feature engineering treatment dramatically improves prediction accuracy. For hyperparameter optimization, the extreme gradient boosting (XGBoost) model has many parameters that will directly affect the performance of the prediction model. Three hyperparameter optimization methods are adopted and compared in this paper to obtain the range of optimal hyperparameter values suitable for the prediction model. Finally, three individual classifiers are ensembled through linear weighted fusion to acquire the heterogeneous ensemble learning-based default prediction model. According to the results of the test, the heterogeneous ensemble learning-based default prediction model with feature engineering, which we develop, performs better than other machine learning models and traditional ensemble learning models.

The remainder of this paper is organized as follows. In Section 2, the relevant literature is briefly reviewed. In Section 3, the three individual classifiers used in the heterogeneous ensemble learning-based prediction model and the contents of the model pretraining and ensemble strategies are introduced. Section 4 describes the analysis and the results of the tests conducted in this paper. Finally, Section 5 concludes this paper.

2 Literature review

Compared with the extensive literature on risk prediction or the credit scoring of traditional banks and other financial institutions, research on default prediction or the credit scoring of P2P lending is limited. Lenders and borrowers, using a P2P lending platform, can engage in fund allocation and circulation without the help of banks and other financial intermediaries, but because these loans are unsecured, credit risk is generated, ultimately leading to default. Therefore, to avoid loss to the greatest extent possible, a random forest is used for the borrowers' credit classification to predict the probability of default. However, the performance of this model declines under a high acceptance level. Although cost sensibility is taken into

consideration, the ability of this method to distinguish borrowers that will default cannot be improved greatly [25]. Emekter et al., by establishing logistic regression (LR) and survival-analyzed credit risk evaluation models, discovered that the debt-to-income ratio (DTI), FICO score and revolving line utilization are highly correlated with loan default [7]. Guo et al. established an example-based loan evaluation model with five features (the FICO score, an inquiry of the last six months, the fund amount, housing ownership and the DTI ratio), which is able to evaluate the return and risk of every P2P loan [10]. The traditional behavior scoring model, which relies on the classification method, will generate a static probability of default, but the repayment of the borrowers varies. Based on the analysis discussed above, Z. Wang et al. developed a new behavior scoring model with a mixed survival analysis framework to predict dynamic default probability in P2P lending [37]. Serrano-Cinca and Gutiérrez-Nieto proposed using a profit scoring method rather than credit scoring in P2P lending: they focused on expected earning capacity for investing in P2P loans using a famous financial ratio - the internal investment ratio [29]. Nonetheless, in studies on P2P lending, class imbalance, high dimension, missing data and other problems associated with credit data have not been effectively solved.

For data with such a high imbalance rate, problems regarding the differences in the size of the different classes in samples cannot be solved, which means that one class will be far larger than another class in the sample. Specifically, whether using default prediction or credit scoring, there are more applications where the prospective borrowers have good credit than those with bad credit, and applications for which the prospective borrowers have bad credit should be rejected. Many researchers have exerted great effort to solve that problem and put forward some effective imbalanced learning methods. The sampling technique is widely applied to solving imbalanced data problems [2]. This method tries to balance the original data using a series of sampling algorithms that adjust the number of different classes in samples and then trains the new “balanced” data using the classification algorithm [6]. Undersampling, which is a common sampling technique, occurs when majority samples are selected randomly and then included into minority classes to form a new training dataset. Different from undersampling, oversampling mainly means randomly duplicating minority samples to increase the sample size of the minority class and then make the minority class roughly equal to other minority classes in the sample. However, this process may cause overfitting of the learning classifier, i.e., the classifier performs well in the training set but badly in the test set. To overcome the weaknesses of random oversampling, a synthetic minority oversampling technique (SMOTE) is proposed; this is a technique that inserts new samples in the minority sample and its adjacent samples rather than duplicating the minority sample directly. Nevertheless, some inevitable problems still exist, e.g., adjacent samples are selected, and the current sample may be in a different class.

The cost-sensitive learning method can be used by increasing the cost matrix, which is composed of the penalty coefficient for the class judgment error, to improve the cost weight of the default sample judgment error. For example, $C(+, -)$ refers to the cost of misjudging a negative sample as a positive one; by the same token, this variable also stands for the cost of misjudging a positive sample as a negative one. Using this method, the classification of original imbalanced data can be simplified into an optimization problem to minimize the classification error of all data samples. In general, it is more important to identify the default sample than to distinguish a general sample. Therefore, the cost of misjudging a positive sample as a negative sample is higher than misjudging a negative sample as a positive sample, i.e., a negative sample. $C(-, +) > C(+, -)$ [24, 38]. B. Krawczyk et al. proposed an original cost-sensitive ensemble based on a decision tree, which solves the imbalanced classification problem effectively [18]. In

most traditional loan evaluation models, a balanced misclassification cost is assumed, which is very different from real-life business. Therefore, Y. Xia et al. developed an improved cost-sensitive tree-style loan evaluation model to improve the ability to distinguish potential default borrowers through cost-sensitive learning and extreme gradient boosting [39].

Because loan data have changed remarkably in the last few decades, both in structure and feature, a single classifier cannot solve the problem effectively [8]. In most previous research, random sampling was adopted to generate a training subset to establish base classifiers, and classifier diversity could not be ensured, which might make the overall classification performance decline. H. Xiao et al. adopted a monitoring cluster to divide every class of data into several clusters, combined them into different classes to form a series of training subsets and established a specific base classifier in each training subset to predict the class tag of the sample. The output of these base classifiers is based on weighted voting. According to the test, the method improves credit scoring accuracy [42]. In light of the fact that there are two data issues: noisy and redundant data occur under the condition of data scoring, two dual-strategy trees are used, a random search (RS)-bagging decision tree (DT) and a bagging-RS DT, both of which use two ensemble strategies (bagging and random subspace) to reduce noise data influence and data redundancy and realize high classification accuracy [23]. Meanwhile, the ensemble method combining multiple algorithms can combine different assumptions to form a better assumption, which can usually yield a better prediction result [26]. Y. Xia et al. provided an original heterogeneous ensemble credit model with an ensemble bagging algorithm with a stacking method, and the effectiveness of this model was verified as [41]. Lessmann et al. also conducted a large-scale empirical analysis and determined that a heterogeneous ensemble usually performs better than individual classifiers [20]. DTs are widely used as a base classifier in boosting; in some classical ensemble algorithms, the DT is used as a base classifier, e.g., AdaBoost, GBDT (gradient boosting DT) and XGBoost [8, 16, 40].

Although many researchers have made many contributions to imbalanced data processing methods and ensemble learning strategies for default prediction [5, 27, 32, 35], few considered these together, and missing value problems in data concentration have not been solved effectively in the relevant literature discussed above. Brown and Mues compared several credit scoring algorithms for imbalanced data concentration: the random forest and gradient boosting have good performance, but there is a parameter optimization still exists according to test results [10]. Therefore, in this paper, an original heterogeneous ensemble learning model is proposed to predict the default of borrowers under P2P lending. We select XGBoost, DNN and LR as the individual classifiers for training. In data preprocessing and feature engineering, we regard a missing value as a kind of feature that can be used to better solve the preceding problems. Considering that the XGBoost model includes many hyperparameters, we fine tune these hyperparameters during the pretraining to optimize the prediction model more effectively. Finally, this study concludes that the heterogeneous ensemble learning-based default prediction model improves the predictive effect dramatically compared to the default model, as shown by the testing results.

3 Methodology

3.1 A heterogeneous ensemble learning framework

By integrating three different individual classifiers, this paper proposes a default risk prediction method based on heterogeneous ensemble learning, which is used to predict the customers'

default level under P2P lending. First, three individual classifiers are described: XGBoost, a deep neural network (DNN), and LR. The most significant characteristic of XGBoost is that it can automatically use the multi-threading of the CPU for parallelism while improving the accuracy of the algorithm. By designing different structures, DNN can change the method it uses to extract the features of the network. This method can also have different effects on the fitting objective function. Moreover, the features extracted by representation learning have stronger generalization performance than the specific settings. This model can be applied to multiple fields. LR is simple, efficient, fast, interpretable and uses a simple calculation. Therefore, combining these methods into an ensemble classifier that can process complex data and has good performance of the prediction is practical for technology.

Second, model training includes feature selection and hyperparameter optimization. Because there are some missing values in the credit dataset used in this paper, to improve the accuracy of the prediction model, the processing of the missing values is not based on the traditional filling method, but the information provided by the missing values is retained to the maximum extent. The feature selection method used by the learning model is regarded as a feature. The rank feature and the discrete feature are generated based on the original feature and then included in the model for training. Because the learning of the model and the selection of the features can be performed simultaneously, the operational efficiency of the predictive model is improved. Then, the feature ranking method based on the learning model is used to reduce the dimensionality to avoid the curse of dimensionality. In addition, the XGBoost model has many hyperparameters that need to be optimized to improve the accuracy of the prediction model. Finally, the ensemble strategy is described. This paper adopts a simple and transparent linear weighted fusion strategy. A flowchart of the approach used in this paper to predict default risk proposed is shown in Figure 1.

3.2 Constructing individual classifiers

3.2.1 Extreme gradient boosting (XGBoost)

The XGBoost model is a highly efficient system implementation based on gradient boosting. On one hand, feature sampling is used to prevent overfitting through the regular terms used to control model complexity, similar to the random forest. On the other hand, using the self-defined data matrix class DMatrix, data are preprocessed to improve the efficiency of each iteration. An approximate algorithm that finds split nodes is used to accelerate and reduce memory consumption. The problem of insufficient computer memory is solved by using off-

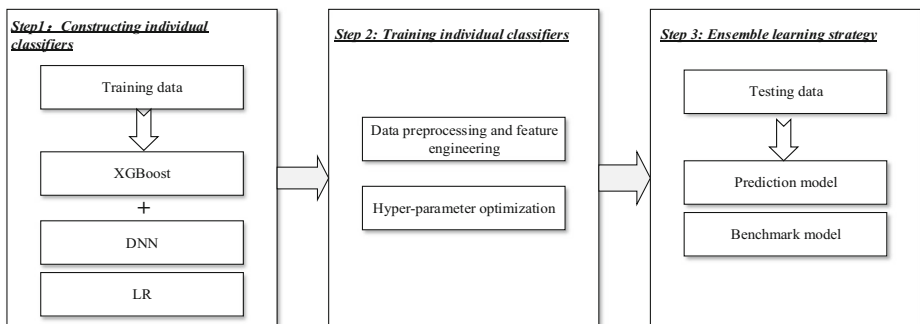


Figure 1 Flowchart of the default prediction model based on heterogeneous ensemble learning

core computing, and the data are sorted in advance and stored in block form, which is conducive to CPU multi-thread parallel computing. In addition, this model has strong expansibility and can be used in combination with other models; as a result, the adaptability of the model is further improved [3, 4].

In the default prediction model, for a given set of credit data $D = \{x_i, y_i\}$ with n samples and m characteristics, x and y represent feature variables and label variables, respectively; similar to a gradient lift, XGBoost uses a K additive function and $f_k(x)$ to approximate the function (namely, $F_K(x) = \sum_{k=1}^K f_k(x)$, $f_k \in \mathcal{F}$, where $\mathcal{F} = \{f(x) = \omega_{q(x)}\}$), and $F_K(x)$ represents the prediction of the k^{th} lift.

The core problem of XGBoost is determining the optimal tree structure. An intuitive search algorithm is summarized as follows: Enumerate all possible tree structures, calculate the score, and then find the optimized structure. However, there is an infinite tree structure; therefore, the calculation cost of enumeration is very high. XGBoost adopts a greedy search algorithm to find the optimized structure. This algorithm is the so-called *exact greedy algorithm for splitting finding*. XGBoost starts from the root node and iterates across the decomposition characteristics. For each node in the tree, XGBoost tries to add a split.

3.2.2 Deep neural network

DNN, which is a typical deep learning model, is the deepest of neural networks. The traditional single hidden layer of the multi-layer feedforward network already has a very strong learning ability, but increasing the number of hidden layers is clearly more effective than increasing the number of hidden layer neurons because adding the number of hidden layers increases the number of neuron activation functions and increases the activation function of net layers. Therefore, a typical DNN is composed of an input layer, no less than two hidden layers and an output layer to form the network topology. The input layer matches the feature space, so there are as many input neurons as there are predictors. The output layer can be either a classification layer or a regression layer to match the output space. All layers are made up of neurons as the basic unit of a model. Using a classic feedforward structure, each neuron in the previous layer L is fully connected with all neurons in the subsequent layer $L + 1$ through the directed edge, and each connection represents a certain weight. Moreover, each nonoutput layer of the network has a bias unit that acts as an activation threshold for neurons in subsequent layers. Similarly, each neuron receives a neuron in the previous layer L as an input, which has a weighted combination α of n_L ,

$$\alpha = \sum_{i=1}^{n_L} w_i x_i \quad (1)$$

where w_i represents the weight of the output x_i and b represents the bias. The weighted combination α of equation (1) is transformed by a certain activation function f , so the output signal $f(\alpha)$ is transmitted to the neurons in the subsequent layers.

This paper uses maxout as the activation function of the DNN. The output expression of the hidden layer node is:

$$h_i(x) = \max_{j \in [1, k]} z_{ij} \quad (2)$$

where $z_{ij} = x^T W_{\dots ij} + b_{ij}$ and $W \in \mathbb{R}^{d \times m \times k}$. Here, w is 3-dimensional, and the size is $d \times m \times k$, where d represents the number of input layer nodes, m represents the number of hidden layer nodes, and k indicates that each hidden layer node corresponds to the nodes of k hidden layers. The nodes of the k hidden layer are all linearly provided, and each node of maxout takes the

largest of the k “hidden layer” node output values. Because there is a maximum operation in the excitation function, the entire maxout network is also a nonlinear transformation.

The ability of maxout to fit is very good, and it can fit any convex function. The most intuitive explanation is that any convex function can be fitted with a piecewise linear function with arbitrary precision, and maxout uses the maximum of k hidden layer nodes. These “hidden layer” nodes are also linear, so under different values, the maximum value can also be regarded as piecewise linear (the number of segments is related to the k value). The only exception is the last layer, using ReLU (rectified linear unit) as the activation function [17].

Overfitting is a common problem in the field of machine learning. Dropout [30] combines the advantages of single and multiple models and is a powerful method that can be used for overfitting in current depth learning [13]. If you do not consider time complexity, you can integrate the different network models to integrate the lifting effect. The greater the difference between the network structures is, the more obvious the lifting effect will be. The idea of dropout is that during each iteration, some nodes (neurons) of the network are randomly prevented from participating in the training, and all the edges related to these temporarily discarded neurons are removed; therefore, the corresponding weights will not be used this time [14]. The weights are updated in the iteration, and this operation is repeated for each iteration of the training. It should be noted that these discarded neurons are only temporarily not updated and will re-participate in the randomized dropout in the next iteration.

In addition, deep learning using DNNs has become a hot topic in the field of artificial intelligence research and application, and many excellent deep learning frameworks have emerged. The Keras framework used in this paper has been favored since its inception due to its good modularity, minimal design and rapid prototyping. Keras is a high-level neural network API (application program interface). Keras is written in pure Python and is based on TensorFlow and Theano; it is a highly modular neural network library that supports CPU and GPU acceleration.

3.2.3 Logistics regression

LR, which is a typical classification method in machine learning, is a model based on statistical learning that has a solid statistics foundation and high interpretability. In this paper, the output variable is binary, and only two values are obtained: $Y = 1$ (loan defaulted) or $Y = 0$ (loan not defaulted). In addition, more approximate values can be obtained through LR, and a regression model can be used to predict the probability of the specific event default. The general form of the LR model is shown below:

$$\text{logit}(P(Y = 1)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$$

where β_0 is nodal increment Y -intercept, $\beta_1, \beta_2, \dots, \beta_k$ are regression coefficients, and $\text{logit}(P(Y = 1))$ is called the logarithm of probability.

3.3 Training individual classifiers

3.3.1 Feature selection

For the feature selection method, the learning model-based feature ordering method is adopted in this paper. Because the credit dataset has many features, including 1138 dimensions, redundancy feature, for the prediction model, redundancy exists and must be eliminated.

Similar to other tree-based models, XGBoost can output a feature importance score using tree boosting techniques, which is a standard way to measure feature importance in a split. A feature variable with a higher score is more significant in the tree structure. In addition, XGBoost provides three feature importance scores to choose from: “weight”, “gain” and “cover”. In particular, “weight” means how many a feature has been used in a split; “gain” is the average error reduction when a feature is split; and “cover” is the average number of samples influenced by a split. Thus, all features in the initial model have a “gain” score because the model is consistent with the original features of CART. A feature with a high importance score will be regarded as a key variable. A sequential forward search (SFS) is used for feature selection; a series of feature subsets is generated by adding important features into the subset and putting the remaining features in the subset on an iterative basis. Although the tree-based model can select important features automatically, there are two reasons for using the feature selection as an independent step, according to the partitioning criterion. First, feature selection improves XGBoost performance. This likely occurs because the redundancy feature is removed from feature selection. The features may be used for splitting during the building of the model. Second, feature selection ensures that there is a fair comparison of the different methods, as support vector machine (SVM), neural network and other reference models are highly dependent on feature selection.

3.3.2 Hyperparameter optimization

Most machine learning algorithms, including classification algorithms, have almost no parameters, and therefore, they take considerable time to train, while some variables need to be configured in advance of the training. These variables have a great influence on the training result, but there is no group of variables that is applicable to every dataset; therefore, the variables should be configured according to the specific application. The variables that need to be configured are called hyperparameters, e.g., an SVM involves a kernel function, regularization coefficient and other hyperparameters; a DT includes tree building and a pruning strategy. Furthermore, many hyperparameters are applied in neural networks. These hyperparameters, which are able to greatly influence model accuracy, are highly necessary for fine tuning. Hyperparameter optimization refers to searching for a group of parameters from the value range of the variables to realize satisfactory machine learning. Nevertheless, optimization, which is often seen as a type of “black art”, mainly relies on subjective judgment, experience and the use of trial-and-error methods.

For some machine learning models, it is very boring and difficult to tune the parameters to optimize the end-to-end machine learning algorithms. There are two basic methods used for parameter selection: manual selection and automatic selection [40]. Manual selection requires an understanding of what the parameters do and how machine learning models can be generalized effectively. The automatic selection of the hyperparameter algorithm decreases the requirement of understanding these issues, but its calculation cost is higher [40].

Manual parameter tuning is usually adopted to obtain experience in practical operating, but the common hyperparameter search method used is grid search (GS) [15] when there are three or fewer hyperparameters. The user should select a small limited value set to search every hyperparameter, and these hyperparameters, through the Cartesian product, will form a group of hyperparameters. The GS uses every group in the hyperparameter training model [43]. The smallest validation set error is used to form the optimal hyperparameter set.

When there are many hyperparameters (e.g., more than 5), the effect of this parameter tuning method will be inferior to that of RS. In RS, a marginal distribution (e.g., a Bernoulli distribution, category distribution or uniform distribution at a logarithmic scale) will be defined for every hyperparameter. For different GS, the hyperparameter values do not need to be dispersed. Therefore, the search is allowed to be implemented in a larger set, without an additional calculation cost. Bergstra and Bengio [1] conducted a detailed study and discovered a RS; which can decrease the validation set error faster than a GS, in terms of test times, foreach model.

In this paper, the RS hyperparameter optimization method is used to optimize the hyperparameters of the XGBoost model to obtain the hyperparameter configuration space of this model. Therefore, the hyperparameters that need to be parameterized are as follows: the learning rate, maximum tree depth, gamma, number of boosts, subsample ratio, column subsample ratio and minimum child weight.

3.4 Ensemble strategy

The fusion advantage of individual classifiers can be shown from the three aspects described below. First, the assumption space of the learning task is very large in terms of statistics, and using multiple assumptions may lead to performance similar to that of the training set. Poor generalization performance may be caused by a false drop if a single classifier is used at this time, and this risk can be lowered by combining multiple individual classifiers. Second, in terms of the calculation, the learning algorithm usually suffers from local minimization, and the corresponding generalization performance of some local minimal points may be very poor. The combination achieved after multiple operations can lower the risk of the poor local minimal point. Finally, at the level of expression, the true hypothesis of some learning tasks may not be in the hypothesis space that the current learning algorithm considers, and the result must be invalid if a single classifier is used at this time. By combining multiple classifiers, a more effective approximate value may be acquired, as the corresponding consumption space is expanded [31, 36, 42].

Based on the analyses above, we use the linear weighting method as a fusion strategy for individual classifiers because it is simple, transparent and easy to implement and performs well in empirical applications [9]. The mathematical form is $H(x) = \sum_{i=1}^T w_i h_i(x)$, where w_i is the weight of the individual classifier h_i . In general, $\sum_{i=1}^T w_i = 1$ when .

4 Experiments and results analysis

4.1 Data description

The testing data used in this paper were obtained from a P2P lending enterprise in China, and the credit dataset is the credit transaction data of the enterprise's loan customers. In addition, the dataset includes 1138 features and 15,000 data points with a tag that indicates whether the customer is expected to default. The data are characterized by missing values, class imbalance and high-dimensional feature. In a prediction model, class imbalance can cause prediction inaccuracy, and high-dimensional sparse data can cause the dimensionality problem [11]. Finally, to verify the effectiveness and accuracy of the prediction model mentioned in this

paper, we apply 80% of the credit dataset to the training and the rest to the testing, and in the testing, we adopt 10-fold cross validation.

To improve robustness, we circulate the test 30 times while using the XGBoost model for pretraining to acquire the feature importance scores of data concentration and complete ordering. Then, we make 36 iterative loops for individual classifiers, using XGBoost, DNN and LR, and the reference models are used for contrastive analysis. In addition, we average the prediction results of these tests to alleviate the influence of the separate divisions.

4.2 Data preprocessing and feature engineering

This section mainly focuses on addressing missing values. The analysis indicates that the majority of samples used for data concentration have missing values, and the number of samples is large; even worse, some samples are missing thousands of values. The frequently used treatment for missing values is missing data imputation, i.e., imputing the average and middle values of the same form of data. However, because there are also many missing values in the testing set, we adopt a new method for treating missing values as the feature.

We calculate the statistic for the number of missing values in every sample in the training set and sequence samples and rank the number of missing values from high to low to determine the number distribution of data missing values in the training set (as shown in Figure 2) using the ID number for the X-coordinate and the number of missing values for the Y-coordinate. In Figure 2, the black curve is a scatter plot of missing values, and the blue curve (rather than a straight line) is the cumulative number of negative samples (i.e., the default samples), showing the strong relevance between the sample class and the number of missing values.

As shown in Figures 2 and 3, regular ladder-like diagrams are formed; in particular, the distribution of the number of missing values in the training set and test set are highly

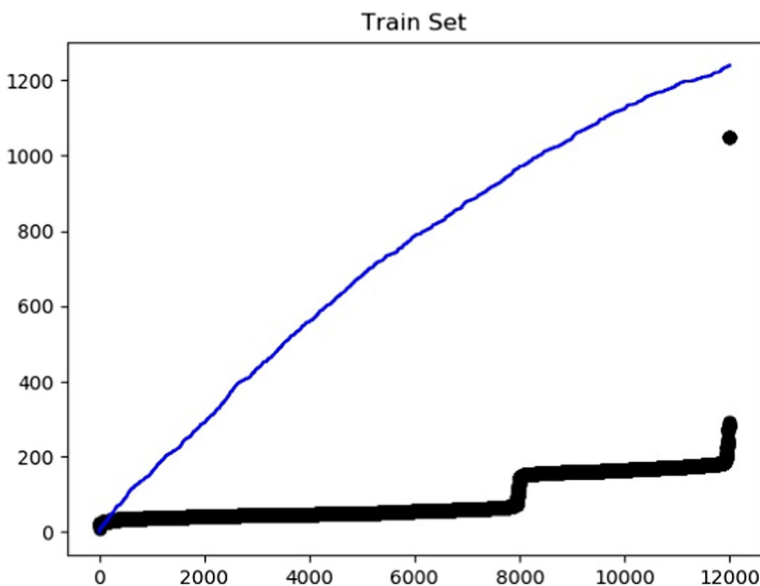


Figure 2 Distribution of the number of missing data values in the training set

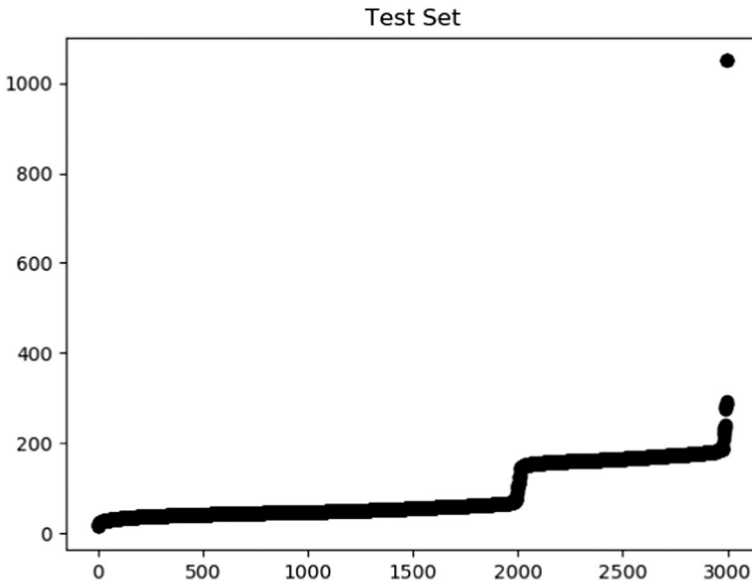


Figure 3 Number distribution of missing data values in the testing set

consistent. Therefore, we divide the dispersed missing values into five ranges to obtain a representational discrete number of missing values (value within 1–5), as shown in Figure 4. We regard this number as a feature here and let the model learn the process.

Furthermore, we delete data that includes more than 194 missing values from the training data, namely, the corresponding sample with a discrete number feature of missing values, as shown in Figure 5. We do this because if the samples include many missing values, model learning becomes difficult and even generates noise and overfitting.

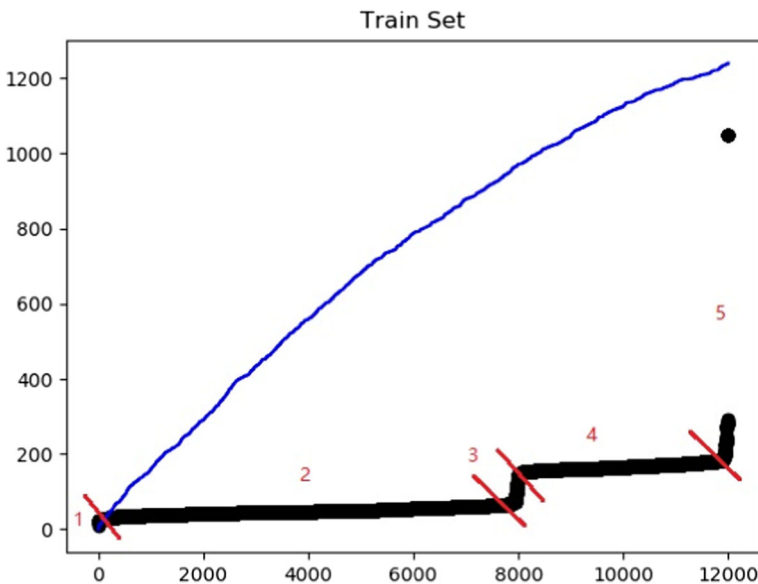


Figure 4 Discrete feature

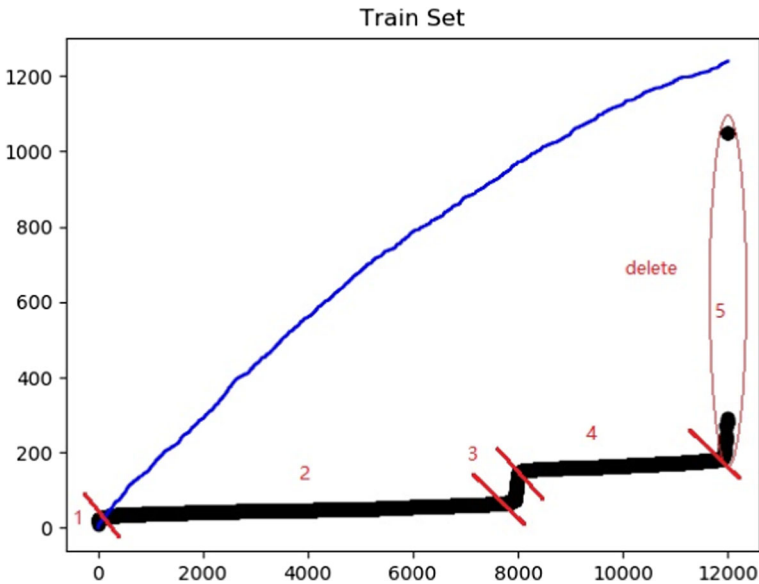


Figure 5 Deletion of incomplete samples

Data preprocessing is closely followed by the feature engineering treatment. There are 1045 dimensions of numeric feature variables in the original data, and 1045 dimensions of rank features are obtained after ordering the data from large to small. The rank features, since they have strong robustness for outliers, make the model steadier, and thus, the risk of overfitting is lowered. We continue to disperse the features after sequencing them. Feature discretization is generally conducted in one of two ways: equal value division (dividing equally as per the value field) and equal size division (dividing equally across the sample size). We adopt an equal size division method to disperse the numeric features: we divide every dimension of features from large to small according to its value and then divide these dimensions into 10 equal ranges; the discretization is 1–10. In addition to the 1045 dimensions of the numeric feature variables, there are also 93 dimensions of categorical feature variables in the original dataset, and many algorithms can only be applied to numeric features. Therefore, we need to code those categorical features. In this paper, one-hot encoding is adopted to obtain 0–1 features, and thus, the problem that the classifiers cannot treat the categorical features is solved.

The rank and discrete features generated based on the original features exceed 3000 dimensions after adding them up. On one hand, having these many features may cause a dimensionality problem but, on the other hand, it may make overfitting easier; therefore, the dimensionality reduction is required. Common dimensionality reduction methods include principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) (with high computer complexity). For this project, PCA dimensionality reduction did not have a good effect. The reasons are as follows: Most of the features of data concentration include missing values, and there is a large number of missing values. PCA requires the data to have a Gaussian distribution, and the testing dataset does not meet this condition [19]. In addition to the dimensionality reduction algorithm, feature selection can be used to decrease the number of features. Commonly used feature selection methods use the maximum information coefficient (MIC), the Pearson correlation coefficient (measuring linear correlation among variables), the regularization method (L1 and L2) and the model-based feature ranking

method [11]. The latter is the most efficient method. One advantage of using this approach is that model learning and feature selection are synchronous. We therefore choose to use this method. A DT-based algorithm (e.g., random forest and boosting tree) can output feature importance after model training has completed. In this paper, we adopt XGBoost, which a boosting tree that is highly efficient and accurate and is widely used for diverse data mining competition, for feature selection.

In Table 1 and Figure 6, we list the top 20 original features trained using the XGBoost model. The column heading “Feature” refers to the name of the feature (there are 1138 dimensions of features in the dataset, of which, 1045 are numeric features, and the rest are categorical features). The feature score indicates the score of the trained feature (in this paper, the first 20 features are shown from high to low). The minimum value is has the lowest score (−1 stands for missing value), and the maximum value has the highest score. The column heading “Number of null values” reflects the number of missing values, and “Type” indicates whether the feature is numeric or categorical (note: in this paper, we only sequence the numeric features).

4.3 Evaluation metric

With respect to classification, particularly regarding the imbalance of the dataset class distribution, a reasonable prediction result cannot be obtained without using an appropriate evaluation standard and a prediction model. In default prediction, one commonly used evaluation criterion is average accuracy (ACC), which is defined as the ratio of the samples and test samples for correct classification. ACC is not the optimal evaluation to use in this paper because it cannot effectively distinguish between good and bad applicants. This occurs because when using an imbalanced dataset, the prediction model may predict all the default customers as nondefault customers when ACC is used as an evaluation criterion, and consequently, default customers cannot be accurately distinguished.

Table 1 Ranking of feature importance

Feature	Score	Min	Max	Number of null values	Type
×29	2810	−1	88.8	62	numeric
×148	2769	−1	22.45	15	numeric
×2	2657	−1	56.41	15	numeric
×486	2372	−1	2,028,938	16	numeric
×554	2356	−1	312	533	numeric
×767	2227	−1	302,150	15	numeric
×171	2135	−1	22.41	15	numeric
×53	2083	−1	100	71	numeric
×951	2010	−1	14,748.62	105	numeric
×944	1993	−1	3,223,695.97	218	numeric
×137	1933	−1	100	62	numeric
×226	1917	−1	24.62	15	numeric
×168	1906	−1	24.11	15	numeric
×606	1888	−1	62,745	15	numeric
×193	1845	−1	50.32	15	numeric
×6	1814	−1	18	15	numeric
×82	1805	−1	86.59	71	numeric
×114	1757	−1	34.79	15	numeric
×791	1720	−1	29,982	15	numeric
×10	1689	−1	78.23	15	numeric

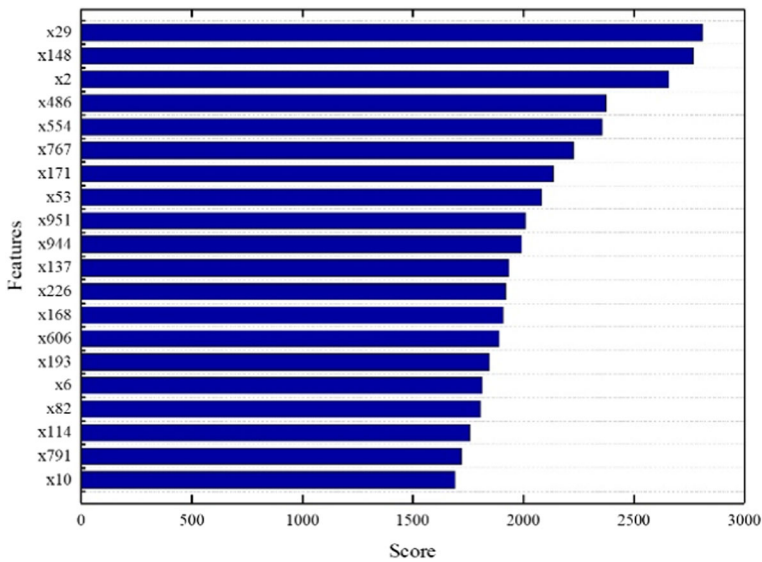


Figure 6 Feature importance scores

The area under the ROC (AUC) is determined by an alternative capacity judgment measurement based on the receiver operating characteristic (ROC) curve. The ROC curve is the complete sensitivity/specificity report used for model evaluation. In the ROC curve, the FPR (false positive rate) is the horizontal axis, and the TPR (true positive rate) is the vertical axis; $FPR = \frac{FP}{FP+TN}$, and $TPR = \frac{TP}{TP+FN}$. For one classifier, obtain a group (FPR, TPR) for the previously mentioned coordinate by adjusting the threshold value of the classifier; then, connect these points to form an ROC curve. The threshold value of the classifier is obtained from the probability output of the classifier. Because the ROC curve fails to evaluate the classifier on a quantitative basis, the AUC score in the area below the ROC curve is usually used as an evaluation standard. The AUC score ranges from 0 to 1, and a higher value indicates a better classification effect. In alignment with the discussion above, the AUC score is used as the evaluation criteria of the prediction model.

4.4 Results and analysis

This section compares the proposed model with different reference models using an actual P2P lending credit dataset. The credit dataset is divided into two parts for verification, i.e., a training set and a testing set; 80% of the data will be used for training, and the remaining 20% will be used for the testing method described in this paper. In addition, to improve robustness, we adopt 10-fold cross validation in the model training and use 30 circulations for hyperparameter optimization. During the testing, the open-source programming language Python (Version 3.6.4) is used. Python is an object-oriented computer programming language that has a rich and powerful library. We use the NumPy and Pandas packages for Python for data preprocessing and feature engineering, and we use the Scikit-learn, Keras, Matplotlib packages for the hyperparameter optimization and model training. All experiments were performed on a notebook computer with a 2.8 GHz Intel i7 CPU, 16 GB RAM, and Windows 10 operating system.

4.4.1 Hyperparameter optimization

Nearly all machine learning algorithms used for default risk prediction are designed with parameters, and these parameters must be optimized to improve the prediction capacity of the algorithms. In a default risk prediction model with hyperparameters, GS and RS are two common parameter optimization strategies that can be used, as well as optimization methods (MS) manual parameter tuning. RS, which was discussed by Bergstra and Bengio, is a random trial-and-error process that is more effective than GS. The GS-based hyperparameter optimization method is adopted in this paper, and the preceding hyperparameter MS and GS are used for the baselines.

This comparison is mainly aimed at evaluating the effectiveness of the RS hyperparameter optimization method relative to that of the other two methods. The RS optimizer is an iterative process, and in this study, the number of iterations is set to 30 ($CV = 5$) to realize a favorable balance between accuracy and complexity. The hyperparameter optimization method will be evaluated from two aspects: prediction accuracy after optimization (measured as per AUC score) and optimization speed (measured as per time (in seconds)). The details on the search space are summarized in Table 2.

In regard to prediction accuracy after optimization, Figure 7 shows the accuracy of using different classifiers for the credit dataset. The X-axis represents the method adopted, and the Y-axis represents accuracy. As shown in Figure 7, greater accuracy is obtained by using RS, which is a method that calculates extreme points through using a random number and then obtaining the optimal solution of function approximation. RS is superior to the other two methods. Table 3 shows the optimization results of the five largest groups of parameters in order; these groups are obtained using RS and GS methods. According to Tables 2 and 3 and Figure 6, prediction accuracy is highest after tuning the parameters with the RS method (with a range of AUC scores for [0.6496, 0.6794]) and a process that lasted 940 s, RS yields the best results at the lowest cost. Manual parameter tuning requires considerable time and effort, and the computable space is limited. If every parameter combination is covered, the process will take a considerable amount of time, and effective optimization will be less likely to be realized. In summary, RS performs between GS and MS in terms of the parameter tuning effect, but its efficiency is far higher than those of the other two methods.

4.4.2 Comparison of the experiments

To verify our method and compare it with the current credit prediction model, we compare the heterogeneous ensemble learning-based default prediction model and various benchmark

Table 2 Search space of the hyperparameter optimization approach

Parameter	Symbol	Random search	Grid search	Manual search
learning rate	learning_rate	(0.001, 0.1)	[0.001, 0.01, 0.1]	0.1
maximum tree depth	max_depth	(1, 12)	[6, 21, 41]	8
gamma	gamma	(0, 0.5)	[0.1, 0.2, 0.3]	0.1
number of boosts	num_boost_round	1000	1000	1000
subsample ratio	subsample	(0.5, 1)	[0.6, 0.7, 0.8]	0.7
column subsample ratio	colsample_bytree	(0.2, 0.5)	[0.2, 0.3, 0.4]	0.3
minimum child weight	min_child_weight	(2, 5)	[10, 28, 34, 44]	3
time	seconds	940	23,600	3900

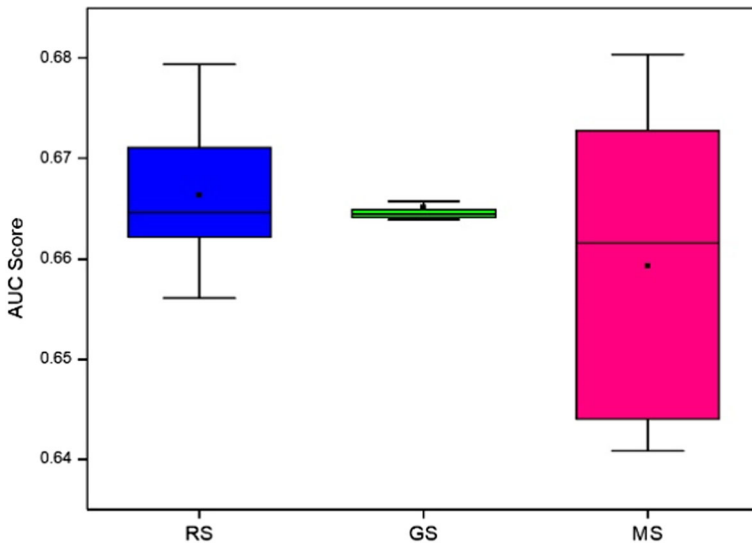


Figure 7 Boxplot of AUC for three hyperparameter optimization approaches

models. Nine models were compared, including six benchmark models and three individual classifiers: the three individual classifiers are XGBoost, DNN and LR; the three traditional machine learning models are SVM, DT and KNN (k-nearest neighbor); and the three ensemble learning-based models are AdaBoost (adaptive boosting), GBDT and RF (random forest). The prediction results are summarized in Table 4, which shows that the heterogeneous ensemble learning-based credit prediction model proposed provides an excellent prediction result for the credit dataset.

Table 3 Tuning performance of the RS and GS methods

Method	Rank	Mean score	Std	Parameter combination
RS	1	0.6794	0.0067	{'subsample': 0.9, 'min_child_weight': 2, 'max_depth': 3, 'learning_rate': 0.069, 'gamma': 0.3, 'colsample_bytree': 0.4}
RS	2	0.6786	0.0109	{'subsample': 0.9, 'min_child_weight': 4.5, 'max_depth': 4, 'learning_rate': 0.054, 'gamma': 0.4, 'colsample_bytree': 0.3}
RS	3	0.6772	0.0101	{'subsample': 0.8, 'min_child_weight': 2, 'max_depth': 5, 'learning_rate': 0.03, 'gamma': 0.3, 'colsample_bytree': 0.3}
RS	4	0.6731	0.0093	{'subsample': 0.9, 'min_child_weight': 2.5, 'max_depth': 2, 'learning_rate': 0.065, 'gamma': 0.2, 'colsample_bytree': 0.2}
RS	5	0.6726	0.0083	{'subsample': 0.5, 'min_child_weight': 4, 'max_depth': 5, 'learning_rate': 0.04, 'gamma': 0.4, 'colsample_bytree': 0.3}
GS	1	0.6688	0.0057	{'colsample_bytree': 0.3, 'gamma': 0, 'learning_rate': 0.1, 'max_depth': 6, 'min_child_weight': 5, 'subsample': 0.7}
GS	2	0.6663	0.0090	{'colsample_bytree': 0.3, 'gamma': 0.1, 'learning_rate': 0.1, 'max_depth': 6, 'min_child_weight': 5, 'subsample': 0.7}
GS	3	0.6649	0.0095	{'colsample_bytree': 0.5, 'gamma': 0, 'learning_rate': 0.1, 'max_depth': 6, 'min_child_weight': 5, 'subsample': 0.7}
GS	4	0.6645	0.0119	{'colsample_bytree': 0.4, 'gamma': 0.3, 'learning_rate': 0.01, 'max_depth': 8, 'min_child_weight': 4, 'subsample': 0.6}
GS	5	0.6641	0.0116	{'colsample_bytree': 0.4, 'gamma': 0.2, 'learning_rate': 0.01, 'max_depth': 8, 'min_child_weight': 4, 'subsample': 0.6}

Table 4 Predicting performance

Models	AUC score using raw features	AUC score using ranked features	AUC score using feature engineering	Basic algorithm
Ensemble	–	–	0.7256	–
XGBoost	0.7038	0.7035	0.7169	ensemble
DNN	0.5284	0.6614	0.7041	neural network
LR	0.6356	0.6896	0.7005	linear model
AdaBoost	0.6616	0.6672	0.6688	ensemble
GBDT	0.6664	0.6776	0.6955	ensemble
RF	0.6522	0.6514	0.6742	ensemble
DT	0.6525	0.6508	0.6660	tree
KNN	0.5029	0.6041	0.5899	tree
SVM	0.5143	0.6898	0.6781	kernel function

0.7256 is the final best predicted AUC score; 0.7038 is the optimal predicted AUC score of XGBoost for using raw features; 0.7035 is the optimal predicted AUC score of XGBoost for using ranked features

In the first group of comparison tests, the prediction result, shown in the second column of Table 4, is obtained after using the original features to train the three individual classifiers and the six benchmark models. The XGBoost model yields the best prediction result, with the AUC score predicted reaching 0.7038. This result shows that the prediction model will yield a better result using ensemble with gradient boosting. This occurs because the feature's value undergoes incomplete sample treatment and XGBoost can learn to its split direction automatically. In addition, the other three ensemble learning-based models AdaBoost, GBDT and RF have prediction results that are higher than 0.65, which reflects good performance and shows that they have a high adaptive ability to address missing values. The KNN model, which includes the untreated original feature, has a very poor prediction result that is only slightly better than a random guess. This result occurs because when there is class imbalance in the sample, large-capacity samples will be in the majority among the k adjacent samples, as in this sample; this model inputs a new sample if the default sample size is very small and the nondefault size is very large. Moreover, the DNN and SVM models perform poorly in terms of addressing the missing values and abnormal values, and these models require a feature engineering treatment to adapt to new data.

In the second group of comparison tests, the prediction result, shown in the third column of Table 4, is obtained after using the ordered feature to train the nine models. The XGBoost model still has the best prediction result, and this result is not very different from the prediction result of the first group of XGBoost models (using raw features). The reason for this result is that upon treating a missing value, the XGBoost model will treat the missing value of an incomplete sample as a sparse matrix. Missing values that are not considered upon a node split are divided into a left subtree and a right subtree to calculate the value, and the superior f will be selected. The tree-base model is less sensitive to missing values, and it is mostly used in cases of missing data. Missing data will become more significant if the distance between two points is calculated when the distance measurement is calculated. This occurs because when the concept of “distance” is involved, the effect will become very small if the missing value is treated inappropriately, e.g., the KNN and SVM models perform very poorly in this aspect. The distance calculation that the loss function of the linear model usually uses involves a different calculation for the prediction value and the actual value and can easily establish effective substitutes for the missing values. Therefore, in the second group, the prediction effect of the LR model is superior to that of the first group. The three ensemble learning-based models AdaBoost, GBDT and RF do not differ much in terms of performance.

Finally, we retrain the nine models after engineering treatment for the original, rank and discrete series of features to obtain the prediction result, which is shown in the fourth column of Table 4. The prediction performance of the three models (XGBoost, DNN and LR) is 0.7169, 0.7041 and 0.7005, respectively. Then, we complete a heterogeneous ensemble for the three individual classifiers using a linear weighted fusion strategy and using 0.70: 0.10: 0.20 as the ensemble weights of the models. The final predicted AUC score is 0.7256, which represents good prediction performance. In this way, we verify the applicability of the prediction model mentioned in this paper for credit data after feature engineering treatment. Additionally, the other three ensemble learning-based models AdaBoost, GBDT and RF have good prediction ability according to their prediction results. After feature engineering, the DT model performs better than those in the first two groups, which shows that the tree-based DT model has good adaptive capacity. The KNN and SVM models have worse prediction performance than those in the second group with ordered features, which means that KNN and SVM models cannot adapt to discrete features well after adding this feature. Figure 8 presents the ROC curve of the prediction result of the heterogeneous ensemble learning-based default prediction model, as well as those of the XGBoost model, the DNN model and the LR model. Figure 9 compares the ROC curves of the above four models. Figure 9 compares the ROC curves of the above four models.

4.5 Discussion

The analysis and credit risk modeling in this paper show that the ensemble learning approach has become a popular modeling technique. The XGBoost algorithm was introduced to enhance

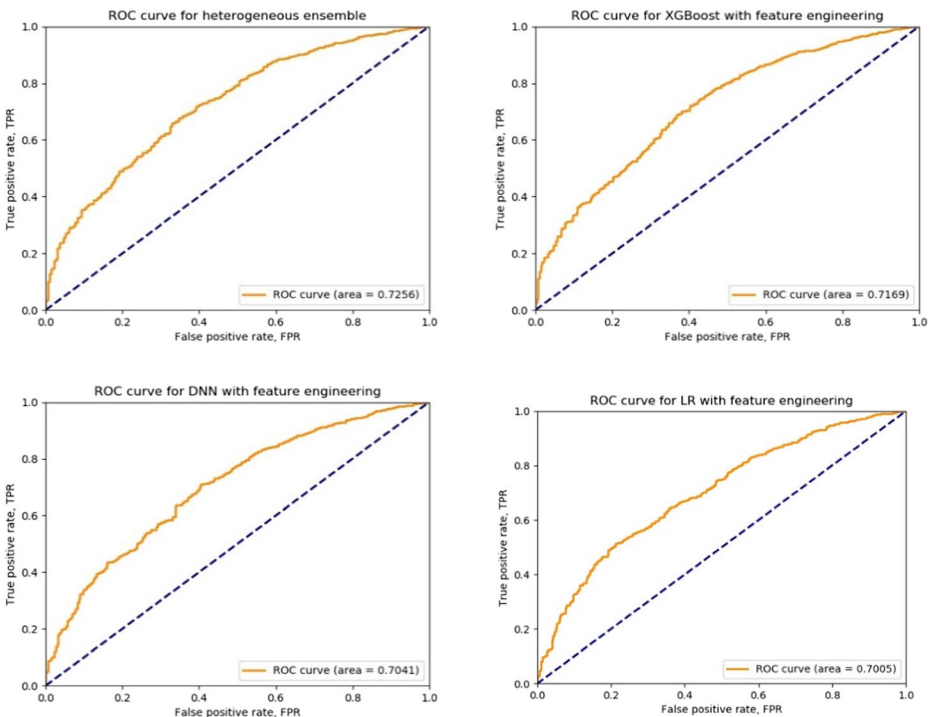


Figure 8 ROC curve for the ensemble, XGBoost, DNN, and LR prediction models

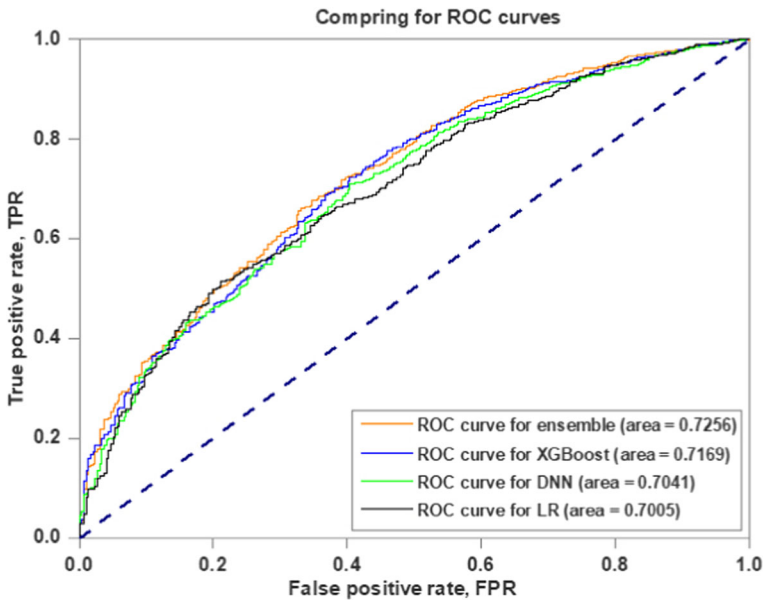


Figure 9 Comparison of the ROC curves

the robustness of the default risk prediction method. The XGBoost model includes tree-based boosting, and at the same time, it adds a regularization term in the loss function and makes some engineering modifications according to GBDT. However, the XGBoost algorithm contains various hyperparameters, and it is very easy to affect the prediction performance if it is not optimized by hyperparameters. To prevent this situation, this chapter adopts the RS hyperparameter method to adjust these hyperparameters.

Risk control is an important part of credit management. Data-driven risk control is a type of quantitative risk control. The development of an information society, an explosion in big data and the Internet have greatly promoted the development of data-driven quantitative risk control. Judging the credit risk of borrowers based on big data is more dependent on technical approaches than traditional risk control methods. At the same time, a large amount of data has characteristics of class imbalance, scarcity and high dimensionality, which further indicates that the machine learning algorithm should be the core method used because it can address these issues.

Future research works should consider ensemble learning modeling based on sequential approaches. At the same time, exponential loss, square loss and other user-defined loss functions should be considered to check how these loss functions are performed, and a more novel hyperparametric optimization method should be applied. The above machine learning algorithm also performs tuning.

5 Conclusion

This paper presents a default risk prediction model based on a heterogeneous ensemble learning strategy and uses three different individual classifiers: XGBoost, DNN and LR. In this heterogeneous ensemble prediction model, a simple and transparent linear weighted fusion strategy is adopted to predict the credit risk of borrowers of P2P loans in China, and an

excellent prediction result is obtained. This paper verifies the validity and accuracy of the heterogeneous ensemble default model with feature engineering for high-dimensional, sparse and class-imbalanced credit datasets.

First, in data preprocessing and feature engineering, addressing missing values is usually done through imputation or simple deletion. However, this paper does not adopt these traditional methods because they may lead to information loss or prediction bias but rather this paper treats missing values as a kind of feature: Ranked and discrete features are generated on the basis of the original feature and these features are treated using a learning model-based feature selection method to ensure that model learning and feature selection are performed simultaneously.

Second, this paper starts model pretraining after the operation above is completed. We train the three individual classifier models XGBoost, DNN and LR first, and then, three traditional machine learning models SVM, DT and KNN are trained. Then, three machine learning models based on traditional ensemble learning AdaBoost, RF and GBD are trained. Because there are many parameters in the XGBoost model, hyperparameter optimization must be completed for the model during the pretraining period.

Finally, this paper uses a linear weighted fusion for the three individual classifier models XGBoost, DNN and LR and then compares the prediction result with other reference models after 36 loop iterations. The default prediction model we propose yields the best prediction performance.

Funding This work was funded by the National Natural Science Foundation of China under Grant Nos. 91846107, 71571058 and Anhui Provincial Science and Technology Major Project under Grant Nos. 16030801121 and 17030801001.

References

1. Bergstra, J., Yoshua Bengio, U.: Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **13**, 281–305 (2012). <https://doi.org/10.1162/153244303322533223>
2. Brown, I., Mues, C.: An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Syst. Appl.* **39**, 3446–3453 (2012). <https://doi.org/10.1016/j.eswa.2011.09.033>
3. Chen, T., International, C.G.-P. of the 22nd acm sigkdd: U.: XGBoost: a scalable tree boosting system. *DL. Acm.Org.* **785–794**(2016), (2016). <https://doi.org/10.1145/2939672.2939785>
4. Chen, K., Jiang, J., Zheng, F., Chen, K.: A novel data-driven approach for residential electricity consumption prediction based on ensemble learning. *Energy.* **150**, 49–60 (2018)
5. Cheng, M.Y., Hoang, N.D., Limanto, L., Wu, Y.W.: A novel hybrid intelligent approach for contractor default status prediction. *Knowledge-Based Syst.* **71**, 314–321 (2014). <https://doi.org/10.1016/j.knsys.2014.08.009>
6. Crone, S.F., Finlay, S.: Instance sampling in credit scoring: an empirical study of sample size and balancing. *Int. J. Forecast.* **28**, 224–238 (2012). <https://doi.org/10.1016/j.ijforecast.2011.07.006>
7. Emekter, R., Tu, Y., Jirasakuldech, B., Lu, M.: Evaluating credit risk and loan performance in online peer-to-peer (P2P) lending. *Appl. Econ.* **47**, 54–70 (2015). <https://doi.org/10.1080/00036846.2014.962222>
8. Feng, X., Xiao, Z., Zhong, B., Qiu, J., Dong, Y.: Dynamic ensemble classification for credit scoring using soft probability. *Appl. Soft Comput. J.* **65**, 139–151 (2018). <https://doi.org/10.1016/j.asoc.2018.01.021>
9. Genre, V., Kenny, G., Meyler, A., Timmermann, A.: Combining expert forecasts: can anything beat the simple average? *Int. J. Forecast.* **29**, 108–121 (2013). <https://doi.org/10.1016/j.ijforecast.2012.06.004>
10. Guo, Y., Zhou, W., Luo, C., Liu, C., Xiong, H.: Instance-based credit risk assessment for investment decisions in P2P lending. *Eur. J. Oper. Res.* **249**, 417–426 (2016). <https://doi.org/10.1016/j.ejor.2015.05.050>
11. Haixiang, G., Yijing, L., Yanan, L., Xiao, L., Jinling, L.: BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification. *Eng. Appl. Artif. Intell.* **49**, 176–193 (2016). <https://doi.org/10.1016/j.engappai.2015.09.011>
12. Han, L., Han, L., Zhao, H.: Orthogonal support vector machine for credit scoring. *Eng. Appl. Artif. Intell.* **26**, 848–862 (2013). <https://doi.org/10.1016/j.engappai.2012.10.005>
13. Ignatov, A.: Real-time human activity recognition from accelerometer data using convolutional neural networks. *Appl. Soft Comput. J.* **62**, 915–922 (2018). <https://doi.org/10.1016/j.asoc.2017.09.027>

14. Iwata, K.: Extending the peak bandwidth of parameters for softmax selection in reinforcement learning. *IEEE Trans. Neural Networks Learn. Syst.* **28**, 1865–1877 (2017). <https://doi.org/10.1109/TNNLS.2016.2558295>
15. Kaneko, H., Funatsu, K.: Fast optimization of hyperparameters for support vector regression models with highly predictive ability. *Chemom. Intell. Lab. Syst.* **142**, 64–69 (2015). <https://doi.org/10.1016/j.chemolab.2015.01.001>
16. Kim, S.Y., Upneja, A.: Predicting restaurant financial distress using decision tree and AdaBoosted decision tree models. *Econ. Model.* **36**, 354–362 (2014). <https://doi.org/10.1016/j.econmod.2013.10.005>
17. Krauss, C., Do, X.A., Huck, N.: Deep neural networks, gradient-boosted trees, random forests: statistical arbitrage on the S&P 500. *Eur. J. Oper. Res.* **259**, 689–702 (2017). <https://doi.org/10.1016/j.ejor.2016.10.031>
18. Krawczyk, B., Woźniak, M., Schaefer, G.: Cost-sensitive decision tree ensembles for effective imbalanced classification. *Appl. Soft Comput. J.* **14**, 554–562 (2014). <https://doi.org/10.1016/j.asoc.2013.08.014>
19. Kuncheva, L.L., Faithfull, W.J.: PCA feature extraction for change detection in multidimensional unlabeled data. *IEEE Trans. Neural Networks Learn. Syst.* **25**, 69–80 (2014). <https://doi.org/10.1109/TNNLS.2013.2248094>
20. Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C.: Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. *Eur. J. Oper. Res.* **247**, 124–136 (2015). <https://doi.org/10.1016/j.ejor.2015.05.030>
21. Li, H., Mao, X., Wu, C., Yang, F.: Design and Analysis of a General Data Evaluation System Based on Social Networks. (2018)
22. Liu, J., Liao, X., Huang, W., Yang, J.b.: A new decision-making approach for multiple criteria sorting with an imbalanced set of assignment examples. *Eur. J. Oper. Res.* **265**, 598–620 (2018). <https://doi.org/10.1016/j.ejor.2017.07.043>
23. Liu, X., Chuai, G., Gao, W., Zhang, K.: GA-AdaBoostSVM classifier empowered wireless network diagnosis. (2018)
24. López, V., Fernández, A., García, S., Palade, V., Herrera, F.: An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. *Inf. Sci. (Ny)*. **250**, 113–141 (2013). <https://doi.org/10.1016/j.ins.2013.07.007>
25. Malekipirbazari, M., Aksakalli, V.: Risk assessment in social lending via random forests. *Expert Syst. Appl.* **42**, 4621–4631 (2015). <https://doi.org/10.1016/j.eswa.2015.02.001>
26. Nascimento, D.S.C., Coelho, A.L.V., Canuto, A.M.P.: Integrating complementary techniques for promoting diversity in classifier ensembles: a systematic study. *Neurocomputing*. **138**, 347–357 (2014). <https://doi.org/10.1016/j.neucom.2014.01.027>
27. Osanaiye, O., Cai, H., Choo, K.K.R., Dehghantanha, A., Xu, Z., Dlodlo, M.: Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP J. Wirel. Commun. Netw.* **2016**, (2016). <https://doi.org/10.1186/s13638-016-0623-3>
28. Paleologo, G., Elisseeff, A., Antonini, G.: Subbagging for credit scoring models. *Eur. J. Oper. Res.* **201**, 490–499 (2010). <https://doi.org/10.1016/j.ejor.2009.03.008>
29. Serrano-Cinca, C., Gutiérrez-Nieto, B.: The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending. *Decis. Support. Syst.* **89**, 113–122 (2016). <https://doi.org/10.1016/j.dss.2016.06.014>
30. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014). <https://doi.org/10.1214/12-AOS1000>
31. Sun, T., Jiao, L., Liu, F., Wang, S., Feng, J.: Selective multiple kernel learning for classification with ensemble strategy. *Pattern Recogn.* **46**, 3081–3090 (2013). <https://doi.org/10.1016/j.patcog.2013.04.003>
32. Sun, Z., Song, Q., Zhu, X., Sun, H., Xu, B., Zhou, Y.: A novel ensemble method for classifying imbalanced data. *Pattern Recogn.* **48**, 1623–1637 (2015). <https://doi.org/10.1016/j.patcog.2014.11.014>
33. Sun, J., Lang, J., Fujita, H., Li, H.: Imbalanced Enterprise credit evaluation with DTE-SBD: decision tree ensemble based on SMOTE and bagging with differentiated sampling rates. *Inf. Sci. (Ny)*. **425**, 76–91 (2017). <https://doi.org/10.1016/j.ins.2017.10.017>
34. Tavana, M., Abtahi, A.R., Di Caprio, D., Poortarigh, M.: An artificial neural network and Bayesian network model for liquidity risk assessment in banking. *Neurocomputing*. **275**, 2525–2554 (2018). <https://doi.org/10.1016/j.neucom.2017.11.034>
35. Tobback, E., Bellotti, T., Moeyersoms, J., Stankova, M., Martens, D.: Bankruptcy prediction for SMEs using relational data. *Decis. Support. Syst.* **102**, 69–81 (2017). <https://doi.org/10.1016/j.dss.2017.07.004>
36. Wang, G., Ma, J., Huang, L., Xu, K.: Two credit scoring models based on dual strategy ensemble trees. *Knowledge-Based Syst.* **26**, 61–68 (2012). <https://doi.org/10.1016/j.knosys.2011.06.020>

37. Wang, Z., Jiang, C., Ding, Y., Lyu, X., Liu, Y.: A novel behavioral scoring model for estimating probability of default over time in peer-to-peer lending. *Electron. Commer. Res. Appl.* **27**, 74–82 (2018). <https://doi.org/10.1016/j.elerap.2017.12.006>
38. Wu, H., Zhang, Z., Yue, K., Zhang, B., He, J., Sun, L.: Dual-regularized matrix factorization with deep neural networks for recommender systems. *Knowledge-Based Syst.* **145**, 46–58 (2018). <https://doi.org/10.1016/j.knosys.2018.01.003>
39. Xia, Y., Liu, C., Liu, N.: Cost-sensitive boosted tree for loan evaluation in peer-to-peer lending. *Electron. Commer. Res. Appl.* **24**, 30–49 (2017). <https://doi.org/10.1016/j.elerap.2017.06.004>
40. Xia, Y., Liu, C., Li, Y., Liu, N.: A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. *Expert Syst. Appl.* **78**, 225–241 (2017). <https://doi.org/10.1016/j.eswa.2017.02.017>
41. Xia, Y., Liu, C., Da, B., Xie, F.: A novel heterogeneous ensemble credit scoring model based on bstacking approach. *Expert Syst. Appl.* **93**, 182–199 (2018). <https://doi.org/10.1016/j.eswa.2017.10.022>
42. Xiao, H., Xiao, Z., Wang, Y.: Ensemble classification based on supervised clustering for credit scoring. *Appl. Soft Comput. J.* **43**, 73–86 (2016). <https://doi.org/10.1016/j.asoc.2016.02.022>
43. Yao, C., Cai, D., Bu, J., Chen, G.: Pre-training the deep generative models with adaptive hyperparameter optimization. *Neurocomputing.* **247**, 144–155 (2017). <https://doi.org/10.1016/j.neucom.2017.03.058>
44. Yeh, C.C., Lin, F., Hsu, C.Y.: A hybrid KMV model, random forests and rough set theory approach for credit rating. *Knowledge-Based Syst.* **33**, 166–172 (2012). <https://doi.org/10.1016/j.knosys.2012.04.004>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Wei Li^{1,2} · Shuai Ding^{1,2} · Hao Wang^{1,2} · Yi Chen^{1,2} · Shanlin Yang^{1,2}

Wei Li
lw5522010@163.com

Hao Wang
wanyehfut@mail.hfut.edu.cn

Yi Chen
chenyihfut@163.com

¹ School of Management, Hefei University of Technology, Hefei 23009 Anhui, China

² Key Laboratory of Process Optimization and Intelligent Decision-Making (Ministry of Education), Hefei University of Technology, Hefei 23009 Anhui, China