Research Article

XGBoost Optimized by Adaptive Particle Swarm Optimization for Credit Scoring

Chao Qin ¹, Yunfeng Zhang ¹, Fangxun Bao ², Caiming Zhang ¹, Peide Liu ³,
and Peipei Liu¹

¹School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan 250000, China

²School of Mathematics, Shandong University, Jinan 250000, China

³School of Management Science and Engineering, Shandong University of Finance and Economics, Jinan 250000, China

Correspondence should be addressed to Yunfeng Zhang; yfzhang@sdufe.edu.cn

Received 19 October 2020; Revised 23 January 2021; Accepted 6 February 2021; Published 24 March 2021

Academic Editor: Sotiris B. Kotsiantis

Copyright © 2021 Chao Qin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Personal credit scoring is a challenging issue. In recent years, research has shown that machine learning has satisfactory performance in credit scoring. Because of the advantages of feature combination and feature selection, decision trees can match credit data which have high dimension and a complex correlation. Decision trees tend to overfitting yet. eXtreme Gradient Boosting is an advanced gradient enhanced tree that overcomes its shortcomings by integrating tree models. The structure of the model is determined by hyperparameters, which is aimed at the time-consuming and laborious problem of manual tuning, and the optimization method is employed for tuning. As particle swarm optimization describes the particle state and its motion law as continuous real numbers, the hyperparameter applicable to eXtreme Gradient Boosting can find its optimal value in the continuous search space. However, classical particle swarm optimization tends to fall into local optima. To solve this problem, this paper proposes an eXtreme Gradient Boosting credit scoring model that is based on adaptive particle swarm optimization. The swarm split, which is based on the clustering idea and two kinds of learning strategies, is employed to guide the particles to improve the diversity of the subswarms, in order to prevent the algorithm from falling into a local optimum. In the experiment, several traditional machine learning algorithms and popular ensemble learning classifiers, as well as four hyperparameter optimization methods (grid search, random search, tree-structured Parzen estimator, and particle swarm optimization), are considered for comparison. Experiments were performed with four credit datasets and seven KEEL benchmark datasets over five popular evaluation measures: accuracy, error rate (type I error and type II error), Brier score, and $F1$ score. Results demonstrate that the proposed model outperforms other models on average. Moreover, adaptive particle swarm optimization performs better than the other hyperparameter optimization strategies.

1. Introduction

Granting potential borrower loans is the core operation of lending establishments around the world. The loan business brings huge profits to a company, while it also enables the company to face a huge financial loss. Therefore, lending institutions need to comprehensively analyse the basic information and credit histories of applicants to estimate the possibility of repayment and then decide whether to approve the application.

An acceptable credit scoring method can help lending institutions distinguish good applicants from loan applications and reject the unacceptable applications. Machine

learning (ML) technology has attracted more attention, such as the most commonly used neural network (NN) [1], the support vector machine (SVM) [2], and decision tree (DT) [3]. They have been increasingly applied in credit scoring. NN is a model of information processing that uses structures similar to synaptic connections in the brain, which will be improved by iteratively adjusting weights to minimize the error of prediction. The capability of NN in treating non-linear data is beneficial to identifying intrinsic patterns in complex financial credit data. Chuang and Huang [4] proposed a hybrid credit score model based on an NN, in which the first part of the model divides applications into the accepted group and rejected group. The results show that the

model obtains a more accurate result than the other compared methods; moreover, it has been proven to reverse potential customer churn. The proposed model reinforced NN primarily to enhance the accuracy without much help in reducing misclassification. Fonseca [5] proposed a two-stage process that employs a fuzzy inference model as an input for an NN model using a credit score rating as a response to conduct the fuzzy reasoning step of the analysis. The results indicated satisfactory predictability of the model. Although these methods add different models to improve the performance of NN, NN still has limitations: it lacks the explanation capability of making lending decisions, performing time-consuming, and overfitting [6]. The SVM is a suitable technique for credit scoring as binary classification [7]. The goal of the SVM is to identify the hyperplane that can correctly separate training instances into correct classes in the space. Hens and Tiwari [8] utilize an SVM in combination with the feature selection index F score which measures the ability of features to distinguish between two categories. The results show that the method is superior to other experimental methods in terms of precision and calculation time. The results obtained by SVM are still sensitive to kernel parameter selection. Luo et al. [9] proposed a kernel-free SVM model which avoids selecting kernels and related kernel parameters. The promising results demonstrate the effectiveness of the proposed method on personal credit data. The limit of the SVM still exists: the boundary of the hyperplane is close to the minority instances when dataset is imbalanced, which results in a decrease in prediction accuracy; the SVM still does not reveal the explanatory nature of lending decision-making. The DT is favored by decision makers because of its reasonable capability of interpretation [6]. By splitting the nodes, the DT forms a tree decision structure. DT is good at capturing interactive information between credit features and shows the indication of important variables. Kao et al. [10] established a Bayesian latent variable model based on a classification regression tree (CART) for the credit data of banks. Compared with other experimental ML models, the prediction accuracy of the proposed model is the highest, and its type I error is significantly fewer. One of the reasons for the improvement is pruning, which reduces the complexity of the model and avoids overfitting. The importance and deduction rules of the variable are derived from the tree structure that supports credit decisions. Xia et al. [11] present a novel model by employing an advanced tree-based model. The model is evaluated over metrics (accuracy, the area under the curve (AUC), and Brier score) on five public credit datasets. The results demonstrate that the proposed model significantly outperforms most of benchmark models on predictive accuracy. The DT has few assumptions on data distribution which allows it to handle complex financial data. Because of DT's intuitive representation of the tree, the result is interpretable. The proposed ensemble method can dynamically assign weights to the base classifier based on the performance of the model, which overcomes the inherent weakness of the sensitivity of a single DT to noisy data or redundant features. Ensemble modelling gives classifiers the opportunity to express their ability to learn on different parts

of data and the feature space. Based on the basis of the "no free lunch" theorem [12] and as the structure and characteristics of changeable credit data are different, the prediction accuracy is greatly limited by a single classifier.

Ensemble learning combines multiple classifiers which process different hypotheses to construct a better hypothesis and obtain excellent predictions. Ensemble models have been extensively applied to credit scoring [13]. Hsieh and Hung [14] employed class-wise bagging as a data augmentation strategy and combined NN, SVM, and other single classifiers into the ensemble classifier. The proposed ensemble approach has significantly better accuracy than these single classifiers on German dataset. Xiao et al. [15] proposed ensemble classification based on supervised clustering for credit scoring. In the method, the base classifier on each subset, which is divided by supervised clustering, is combined by weighted voting. The proposed approach has a higher prediction accuracy than the base classifiers DT, logistic regression (LR), and SVM on datasets of German and Australian. Xia et al. [16] proposed an ensemble credit model that combines bagging with stacking. On datasets of German, Australian, and P2P, the ensemble model is better than single classifiers (LR, SVM, and DT) in accuracy, AUC, and Brier score. Numerous studies have shown that the models for integrating a single classifier achieve better performance [6, 17], which shows that ensemble learning has a satisfactory learning ability for credit scoring.

Furthermore, Chen and Guestrin [18] proposed an advanced gradient boosting algorithm, the eXtreme Gradient Boosting (XGBoost) tree, that has obtained good results in Kaggle data competitions. Zieba and Tomczak [19] applied XGBoost to bankruptcy prediction. For the real unbalanced data of Polish companies, the XGBoost model utilizes the advantage of being insensitive to imbalanced data phenomenon as it enables the selection of an AUC measure for evaluation and forces proper ordering of the imbalanced data. The model performs significantly better than other methods, including the NN, LR, SVM, and random forest (RF). Based on the notion that credit data and bankruptcy data belong to the financial risk domain [6], XGBoost provides a viable scheme for credit scoring. Chang [20] used XGBoost to construct a credit scoring model with cluster-based undersampling. XGBoost is based on DT and is supported by gradient boosting which improve its predictive power. With real loan data of a financial institution, the assessment model exhibits superior accuracy and AUC values to LR, the NN, and the SVM. Xia et al. [21] proposed a cost-sensitive credit risk assessment model by XGBoost to predict the risk of a loan. The prediction of the default probability of the purposed model is better than LR and RF on two *P2P* lending data from platforms of LendingClub and Renrendai. An explanatory data analysis shows that variables differ in predicting the probability of default and profitability, due to the interpretability of the XGBoost tree structure.

Credit data as financial data have inherent characteristics, such as credit data imbalance caused by a small number of default applications, a large number of credit features and complex feature relationships, a sparse

matrix problem caused by many missing values and features, and interpretability issues for feature decisions. When faced with various problems, traditional ML methods and other integrated learning methods can address some of these problems. The field of credit scoring is directly related to economic returns, which requires models to have higher capabilities to address the characteristics of credit data in more aspects. XGBoost allows custom objective functions and evaluation indicators to guide the model to deal with special situations, such as dealing with imbalances by using the AUC indicator as the objective function. By calculating the gain of samples in the left and right subtrees to decide the default direction of the missing data, XGBoost learns how to deal with missing values. XGBoost with the mechanism is appropriate for credit data with missing values and sparse characteristics. XGBoost's intuitive interpretation capabilities can explain the contribution of features to prediction and identify the commonalities or opposite relationships between two features, which are meaningful for understanding decision-making.

Although XGBoost performs well, its satisfactory performance depends on having the proper hyperparameter settings. The hyperparameters of XGBoost directly affect the structure of the model and the performance of the model, so appropriate tuning is particularly important. Generally, manual tuning relies on artificial experiences, which consumes a substantial amount of time and computing resources. Hyperparameter optimization methods have been employed to tune and overcome disadvantage of manual tuning. Hyperparameter optimization that is commonly utilized in machine learning consist of grid search (GS) and random search (RS) [22]. Within the limited resources, GS and RS, as discrete optimization methods, may overlook the optimal value as the objective function is nonconvex. Different from RS or GS, the tree-structured Parzen estimator (TPE) [23] is a Bayesian method whose proxy model is a tree structure, which builds a probabilistic model based on obtaining the evaluation results of past targets to minimize the objective function. The algorithm is limited to the condition that each hyperparameter is completely independent and is difficult to meet. Particle swarm optimization (PSO), which was originally proposed by Kennedy and Eberhart [24], is a computational intelligence technique. The optimization of hyperparameters is performed in a continuous search space. Shen et al. [25] proposed an ensemble model by combining the AdaBoost method with the NN base classifier and employed PSO to search for the optimal connection weight of the NN. The results show that this model is more effective in processing datasets of German and Australian than the other models examined. The original PSO algorithm was mainly designed for the optimization of a continuous space owing to the quantities that describe the particle state, and its motion laws are continuous real numbers [26]. Therefore, PSO rather than GS, RS, and TPE is more suitable for the hyperparameter optimization of XGBoost. PSO finds the optimal solution through iteration, and it has a fast convergence speed. This characteristic easily causes the states of the particle to fall into a local optimum, thereby causing

premature convergence. In response to this problem, we use the idea of clustering [27] to adaptively divide the particle swarm into different populations and guide the populations by applying different update strategies. This enhances the diversity of particles and helps particles jump out of a local optimum.

On the basis of the above considerations, we use the adaptive particle swarm optimization (APSO)-XGBoost model for credit scoring. APSO is more suitable for the parameter optimization of XGBoost, and it improves the model prediction accuracy.

The contributions of this paper are as follows.

First, XGBoost is optimized by an APSO algorithm to establish a credit scoring model. The hyperparameters of the model are set to more reasonable values, and more accurate prediction results are obtained.

Second, the proposed APSO method is based on adaptive subgroup division and uses two different learning strategies to update different types of particles to enhance the diversity of the particle populations.

The rest of the paper is organized as follows: Section 2 explains the related work on the methods used. Section 3 introduces the principle of the APSO-XGBoost model. Section 5 describes the experimental setup. Section 6 reports the experimental analysis results. Finally, Section 7 concludes the paper and discusses future work.

2. Related Work

Much of the credit scoring research is concentrated on purposing a new algorithm to improve the accuracy of classification. In general, the credit scoring method has two major categories: statistical method and artificial intelligence (AI) method. [6] The aim of the statistical method is to obtain an optimal linear combination of input variables to predict default risk. LR, a statistical method, is a popular model that is utilized in credit scoring as it is easy to implement [28]. LR uses the maximizing likelihood function to classify parameters as default or nondefault and solve parameters of expression by gradient descent. Note the method requires a linear relationship between input variables, and its performance will be limited when addressing with real complex nonlinear credit data.

ML has more outstanding predictive ability in addressing with complex nonlinear data [29]. Several studies have shown that ML methods have better prediction accuracy than statistical methods for financial data. As traditional ML methods, NN, SVM, and DT are commonly applied to predict personal default [30]. Oreski and Oreski [31] proposed a hybrid genetic algorithm based on NN, which achieved good prediction performances. Maldonado et al. [32] proposed a profit-driven model that can simultaneously construct classifiers and select characteristics based on the SVM, which performs well in the credit scoring business of a Chilean bank. Cai and Zhang et al. [33] obtained the satisfactory performance of default prediction by applying DT and LR to LendingClub data. The amount of training data of a single classifier is insufficient, and its hypothesis space is small, so it is easy to obtain the local

optimal value. Based on reasonable but different ideas, ensemble learning can solve the issue by combining different single classifiers. Single classifications applied to credit scoring commonly suffer from negative effects from the noise and redundant attributes, similar to the other research. The single classifications show a large deviation in processing complex data with noise and redundancy characteristics, which will cause the performance of the model to be unstable on various indicators.

To compensate for the limitation of single classifier, applying ensemble learning to credit scoring is becoming a positive trend [16]. Ensemble learning has two main methods: bagging and boosting [13]. Bagging adopts repeated random sampling to generate a random subset. The final result is obtained from the concordance of prediction model on each subset. As a famous bagging classifier, the RF trains many trees on the random subset and then adopts the mean of a result set of various trees as the final output. Since bagging trains classifiers on each bootstrap sample and combines the results by majority voting or averaging, they are less influenced by noise than single classifier. The Bagging-DT and Bagging-NN introduce a bagging strategy into a single classifier to achieve better performance [17]. Unlike the bagging constructs the base models in parallel, the boosting sequentially builds models. Boosting is an iterative algorithm that combines weak learners based on weight. AdaBoost [34] is the most influential boosting algorithm that improves the weights of weak classifiers with a low error rate and reduces their weights with a high error rate in each iteration. AdaBoost is a framework that can use a variety of base learners. The DT is the most prevalent default base classifier, and the AdaBoost-NN base classifier is a multilayer perceptron; they both obtain satisfactory results for credit scoring [35]. Because of the simple construction of the weak base classifier, AdaBoost is not easy to overfit. The classification accuracy of AdaBoost will decrease due to data imbalance.

The Gradient Boosting Decision Tree (GBDT) [36] adopts the regression tree as the weak classifier, which adds a new tree to the model in each iteration to fit the residuals. GBDT has the flexibility to handle various types of credit data; it handles exceptional or fraudulent applications better due to the robustness of the loss function. Compared with GBDT, XGBoost uses first and second derivatives, and its loss function is more accurate, which can split the tree more precisely to make the tree structure more suitable for financial data characteristics. XGBoost adds a regularization term in the cost function to control the complexity of the model and prevent the model from overfitting when it learns credit data with a small sample. In the case of missing eigenvalues, the segmentation direction of the credit sample is automatically learned. These advantages make the XGBoost achieve reasonable prediction in credit scoring [20].

To take advantage of the efficacy of XGBoost, the model needs to be optimized by a hyperparameter optimization approach. GS attempts every combination of the candidate parameters via the loop traversal and selects the best performance parameters as the final result. Instead of

testing all values in the search area, RS randomly selected sample points, which is faster than GS. Due to a relatively large number of hyperparameters, the two algorithms may be infeasible to XGBoost. Chang et al. [20] proposed XGBoost optimized by the TPE to predict credit data. The results of the experiment show that the model outperforms baseline models on average with regards to accuracy, error rate, AUC, and Brier score. The TPE requires that the hyperparameters are independent of each other [23]. In the experiment, hyperparameters that are highly correlated cannot be optimized simultaneously, and one of them needs to be fixed first, which removes the relation between the hyperparameters. The aim of PSO is to obtain the optimal value via the cooperation and information sharing among individuals in the swarm, which was originally designed to address the continuous optimization problem. The algorithm searches according to individual adaptation information, so PSO is not limited by objective function constraints, such as continuity and differentiability. Moreover, the algorithm does not require mutual independence of the hyperparameter.

Based on the advantages of XGBoost in reasonably addressing the missing values of credit data, more accurate solution of loss function, mechanisms to prevent overfitting, and the advantages of PSO in optimizing parameters in high-dimensional continuous space, we build an XGBoost model that is optimized by an improved APSO to realize high-precision credit scoring. APSO sets the objective function as the fitness of the particle and directs the particle to simultaneously optimize multiple hyperparameters in the continuous space and obtains the hyperparameter value that makes the objective function lower. This reasonable hyperparameter setting enables XGBoost to fully exert its effect, reduce the influence of credit data imbalance on the model, learn the direction in which credit data missing values of credit data are split reasonably, and improve the accurate performance of the model.

3. Materials and Methods

Hyperparameter optimization has always been an important aspect of prediction models. In this section, we introduce the related works on these techniques.

3.1. XGBoost. XGBoost is a powerful methodology for regression as well as classification. It is applied as a group of winning programs from Kaggle competitions. XGBoost, which is based on the gradient boosting framework, constantly adds new decision trees to fit a value with residual multiple iterations and improve the efficiency and performance of learners. Unlike gradient boosting, XGBoost uses a Taylor expansion to approximate the loss function, and the model has a better tradeoff bias and variance, usually using fewer decision trees to obtain a higher accuracy. Below is a description of XGBoost.

Assume that a given sample set has n samples and m features; it can be expressed as $D = \{(x_i, y_i)\} (|D| = n, x_i \in R^m, y_i \in R)$, where x is the eigenvalue and y is the

true value. The algorithm sums the results of K trees as the final predicted value, which is expressed as

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad f_k \in F. \quad (1)$$

F is the set of decision trees as

$$F = \{f(x) = w_{q(x)}\} \quad (q: R^m \longrightarrow T, w \in R^T), \quad (2)$$

where $f(x)$ is one of the trees and $w_{q(x)}$ is the weight of the leaf nodes. T is the number of leaf nodes, and q represents the structure of each tree, which maps the sample to the corresponding leaf node. Therefore, the predicted value of XGBoost is the sum of the value of the leaf nodes of each tree. The goal of the model is to learn these k trees, so we minimize the following objective function:

$$L^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k), \quad (3)$$

where l is the loss of the difference between the estimated values $\hat{y}_i$ and the true value y_i ; common loss functions include the logarithmic loss function, square loss function, and exponential loss function. Ω regularization is used to set the penalty of the decision tree, which can prevent overfitting. Ω is expressed as follows:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \|\omega\|^2. \quad (4)$$

In the regular term, γ is a hyperparameter that controls the complexity of the model and T is the number of leaf nodes. λ is the penalty coefficient for the leaf weight ω , which is usually constant. λ and ω determine the complexity of the model and are usually given empirically. During training, a new tree is added to fit the residuals of the previous round. Therefore, when the model has t trees, it is expressed as follows:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i). \quad (5)$$

Substituting (4) into the objective function (2) yields the following function:

$$L^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t). \quad (6)$$

XGBoost carries out the Taylor expansion of the objective function, takes the first three terms, removes the high-order small infinitesimal terms, and transforms the objective function into

$$L^{(t)} \approx \sum_{i=1}^n \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t), \quad (7)$$

where g_i is the first derivative and h_i is the second derivative of loss function. The residual between the prediction score $\hat{y}_i^{(t-1)}$ and y_i does not affect the optimization of the objective function, so it is removed:

$$\tilde{L}^{(t)} = \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t). \quad (8)$$

The iteration of the tree model is transformed into the iteration of the leaf nodes, and the calculated optimal leaf node score is $G_j^2 / (H_j + \lambda)$. By substituting the optimal value into the objective function, the final objective function is obtained:

$$\text{Obj} = \frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T. \quad (9)$$

Overall, XGBoost adds regularization to the standard function as a result of the reduced model complexity. The first and second derivatives are applied to fit the residual error. This method also supports column sampling in both reducing overfitting and reducing computation. Therefore, more improvements lead to a larger number of hyperparameters than the GBDT. However, it is difficult to reasonably tune the hyperparameters. A reasonable setting requires not only the prior knowledge of researchers and their experience in parameter tuning but also a great deal of time. Hyperparameter optimization is an effective solution to this problem.

3.2. Parameter Optimization. The parameter is one of the most significant concepts in ML, and the training model essentially finds the appropriate parameters to achieve better results. The parameters are divided into model parameters and hyperparameters. The model parameters are obtained by learning the distribution of training data, without the need for human experience. A hyperparameter is defined as a higher-level concept about the model, such as its complexity or ability to learn. Having a set of satisfactory hyperparameters improves the performance of learning models, so tuning them is important. However, hyperparameter tuning is subjective and relies on empirical judgement and trial-and-error approaches [23]. The hyperparameter optimization algorithm overcomes the dependence of manual search on experience and trial and error. Common hyperparameter optimization algorithms include GS, RS, and Bayesian optimization. The booster parameters of XGBoost are commonly applied to improve the performance of the model. The parameters directly control the generation of trees at each step, and the formulation of frequently selected booster parameters is described in Table 1.

Below is a brief introduction to them.

GS: within a specified range of the hyperparameters, GS is a method that performs one step at a time to adjust the hyperparameters via training and is determined to be the best among all validation methods. However, a GS cannot widely explore a hyperparameter space as the computational complexity of GS grows exponentially when the iterations of the algorithm are increased to create more opportunities based on the number of hyperparameters. Therefore, GS is not suitable for the optimization of models with many dimensions [37].

TABLE 1: Formulation of hyperparameters of XGBoost.

Hyperparameter	Formulation
Learning rate	Reduce the weight value of each step and improve the robustness of the model.
Maximum tree depth	The maximum the depth of the tree, the greater the value, the complex the trees. It avoids overfitting.
Subsample ratio	The sampling rate of the sample.
Column subsample ratio	The proportion of features sampled when constructing each tree (typically the characteristic sampling).
Minimum child weight	Determine the sum of the minimum leaf nodes' sample weights. It reduces overfitting.
Maximum delta step	It limits the maximum length of each tree weight change. It is useful for LR when the data are very unbalanced.
Gamma	Specifies the minimum loss function reduction required when a node is partitioned. It avoids overfitting.

RS: RS samples a certain number of sets from a specified distribution by randomly sampling within a search range. The theoretical basis is that if the set of random sample points is large enough, the global optimal value or its approximation will be found.

Bayesian optimization [38] in tuning hyperparameters was intended to optimize the objective function of sample points by adding it to the objective function to update the posterior distribution. The calculation process of this algorithm is Gaussian and considers the information of the last hyperparameter and then adjusts the hyperparameter to slowly improve the joint posterior distribution. Bayesian hyperparameter optimization assumes that there is a real distribution, and the noise of the hyperparameter is mapped to a specific target function.

Bergstra et al. [23] proposed the TPE, which is a Bayesian optimization method for tuning the hyperparameters of XGBoost. The results show that the model outperforms other models according to the evaluation measures. However, Bayesian optimization is established on the basis of the distribution of the independent prior and the idealized hypothesis that properties are independent of each other. This condition is difficult to attain in practical applications; the number of properties may be large or the correlation between the properties may be high, thereby causing performance degradation.

3.3. PSO. PSO simulates a bird in a flock by designing a massless particle that has only two properties: its speed, which represents how fast it moves, and its position, which guides the direction in which it moves. Each particle determines the optimal solution in the search space of an individual and stores it as the current individual extremum. According to the current individual extremum of all the particles, to obtain the current global optimal solution, the whole particle swarm adjusts its speed and position. The process of PSO is as follows.

First, initialize the particle swarm; then, evaluate the particles, calculate the adaptive value, search for individual extrema, and find the global optimal solution. Last, modify the speed and position of the particles.

The standard PSO algorithm is as follows.

Assume that in a D -dimensional search space, there is a population of m particles represented as $\{x_1, x_2, \dots, x_m\}$, where the particles are expressed as $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})$. At a set time t , the characteristic information of x_i is the position $X_i^t = [x_{i1}^t, x_{i2}^t, \dots, x_{iD}^t]^T$, velocity $V_i^t = [v_{i1}^t, v_{i2}^t, \dots, v_{iD}^t]^T$, personal best position $p_i^t = [p_{i1}^t, p_{i2}^t, \dots, p_{iD}^t]^T$, and

global optimal position $p_g^t = [p_{g1}^t, p_{g2}^t, \dots, p_{gD}^t]^T$. Then, the speed and position information of the particle can be updated at time $t + 1$ by the following formula:

$$\begin{aligned} v_{iD}^{t+1} &= \omega v_{iD}^t + c_1 r_1^t (p_{iD}^t - x_{iD}^t) + c_2 r_2^t (p_{gD}^t - x_{iD}^t), \\ x_{iD}^{t+1} &= x_{iD}^t + v_{iD}^{t+1}. \end{aligned} \quad (10)$$

ω is the inertia weight that maintains an effective balance between global exploration and local exploration, and c_1 and c_2 are the learning factors (random numbers in a uniform distribution function) that adjust the step length of the direction of motion to the position of the particle and the direction of the global best position, respectively. To avoid a blind search by the particle, its speed and position are generally limited to $[-V_{\max}, V_{\max}]$ and $[-X_{\max}, X_{\max}]$.

4. APSO-XGBoost Credit Scoring Model

Hyperparameters control the complexity or regularization to refine the model. We need to carefully employ parameter optimization to tune the hyperparameters to obtain better prediction ability. As a powerful ML algorithm, XGBoost has numerous hyperparameters. APSO can simultaneously obtain the optimal value of multiple XGBoost hyperparameters in n -dimensional space. With the advantage of the characteristics of the position updating formula, APSO solves the continuous optimization of better searching for the optimal hyperparameter of XGBoost. Additionally, improved APSO divides the population by clustering to improve the diversity of subgroups and adopts two updating strategies for different types of particles to average the information of locally optimal particles and prevent the local convergence caused by the local aggregation of particles. APSO impacts XGBoost to adequately fit the credit data and improve its prediction accuracy. First, we described the APSO, and second, we concretely introduced the overall framework of the credit scoring model.

4.1. Adaptive PSO. In this section, we introduce PSO improved by adaptive learning strategies. In the process of searching, a swarm is adaptively divided into subgroups according to the particle distribution. In each subgroup, we use two different learning strategies to guide the search directions of two different types of particles. The search process stops when a global optimal value is found or a termination condition is met.

Relevant studies have shown that the diversity of the population is the key to avoiding the premature convergence

of PSO; the core guiding principle of the algorithm is clustering [26]. According to the distribution of each particle, the fast search clustering method [39] is adopted to perform the adaptive division of the population into several subgroups. This method can automatically discover the dataset samples' class cluster centre. The basic principle is that the centre of the class cluster has two basic features: the first is that it is surrounded by points with a lower local density, and the second is that it has a greater distance from points with a higher local density. Therefore, for a population of H particles, where $S = \{x_i\}_{i=1}^H$, the two properties ρ_i and δ_i are defined for each particle. ρ_i , the distance between the local density of the particle and a higher local density of particles, is defined as follows:

$$\rho_i = \sum_{j \neq i} \exp\left(-\left(\frac{d_{ij}^2}{d_c}\right)\right), \quad (11)$$

where d_{ij} is the Euclidean distance of particles between x_i and x_j and d_c is the truncation distance. The truncation distance is $d_c = d_{R*M}$, where R represents the proportion and M indicates that the matrix d_{ij} contains $M = (1/2)N(N-1)$ values. It can be seen that d_c is the distance corresponding to the $R * M$ th value of d_{ij} . (12) gives the expression of the distance δ_i , representing the minimum distance from particle i to other particles that have a higher ρ_i :

$$\delta_i = \min_{j: \rho_j > \rho_i} (d_{ij}). \quad (12)$$

For the maximum local density ρ of the sample, $\delta_i = \max_j d_{ij}$.

According to equation (11), if the density of particle x_i is the maximum, δ_i is much larger than the distance δ of its nearest particles. Therefore, the centre of the subgroup consists of particles that have an unusually large distance δ and a relatively high density as well. In other words, the particles with larger ρ and δ values are selected as the centre of the cluster. According to the above idea from [39], the formula $\gamma_i = \rho_i * \delta_i$ is used to filter out particles that may become cluster centres. We arrange the γ_i values in descending order and then use the truncation distance to filter out the cluster centres from the order. Because the γ value of the top particle is more likely to increase exponentially than those of the other particles, it is distinguished from the γ value of the next particle. Referring to [39], R is set to be between 0.1 and 0.2. Through a parameter sensitivity analysis, we found that the value of the distribution parameter has no effect on the performance of the particle swarm algorithm. The default value in this article is 2. The cluster centre is obtained by dividing the truncation distance after placing the other particles x_j in subgroups where the denser ρ is larger than the ρ of x_j and the γ is the closest to the γ of x_j .

The particles of each subgroup are divided into ordinary particles and local optimal particle based on the result of the division of subgroups. Under the primary guidance of the optimal particles, the ordinary particles exert their local search ability, and the updated formula is given as follows:

$$\begin{aligned} v_i^d &= \omega v_i^d + c_1 \text{rand}_1^d (\text{pbest}_i^d - x_i^d) + c_2 \text{rand}_2^d (\text{cgbest}_c^d - x_i^d), \\ x_i^d &= x_i^d + v_i^d, \end{aligned} \quad (13)$$

where ω is the inertia weight, c_1 and c_2 are the learning factors, rand_1^d and rand_2^d are uniformly distributed random numbers in the interval $[0, 1]$, pbest_i^d is the best position of particles, and cgbest_c^d is the current best position of the particles in subgroup c . To enhance the exchange of information between subgroups, the local optimal particles are mainly updated by integrating the information of each subgroup. The update formula is presented as follows (see (14)), where C is the number of subgroups:

$$\begin{aligned} v_i^d &= \omega v_i^d + c_1 \text{rand}_1^d (\text{pbest}_i^d - x_i^d) \\ &+ c_2 \text{rand}_2^d \left(\frac{1}{C} \sum_{c=1}^C \text{cgbest}_c^d - x_i^d \right), \end{aligned} \quad (14)$$

$$x_i^d = x_i^d + v_i^d.$$

Ordinary particles search for local optimality, but more importantly, they are used as the medium for information exchange between subgroups to modify the direction of population search and further improve the population diversity. In the same subgroup, unlike a learning strategy that causes too many particles to be gathered locally, the learning strategy integrates the information of the locally optimal particles from different subgroups to obtain more information and help avoid local optima. In addition, learning too much information may lead to the direction of the update being too fuzzy, which may counteract the convergence of particles. Considering that the local optimal particles have the maximum probability of finding the optimal solution in the subgroup, valuable guidance for the optimal solution is provided by their information. Therefore, the cgbest_c^d of each subgroup uses the average information to guide the local optimal particle update (see (14)). The transmission of the optimized information in the subgroups can be improved by this approach, the population diversity can be further increased, and particles can be prevented from falling into local optima.

4.2. APSO-XGBoost. This paper proposes a credit scoring model based on XGBoost optimized by APSO. The model is divided into three parts: data preprocessing, feature engineering, and model training. First, the data preprocessing involves standardized datasets and marked missing values. Second, the feature engineering is based on the score of the feature importance that is obtained from the initial hyperparameter model. According to the rank of the feature importance, redundant features are removed. Last, according to the selected features and hyperparameters tuned by APSO, the model is built. The flow chart is shown (see Figure 1). The process is described in detail below.

FIGURE 1: The process of the APSO-XGBoost credit scoring model.

4.2.1. Data Preprocessing. Data preprocessing is divided into two steps: data standardization, namely, 0 – 1 scaling, and missing value processing. Although the tree-based algorithm is not affected by scaling, feature normalization can greatly improve the accuracy of classifiers, especially those based on distance or edge calculations. Therefore, the standardization of datasets in data preprocessing renders the model to be more accurate and persuasive. The training set is described as $D = \{X, Y\}$, where $X = \{x_1, x_2, \dots, x_m\}$ represents an m -dimensional eigenspace, $Y = \{0, 1\}$ represents the target value, $Y = 0$ represents poor application, and $Y = 1$ represents good application. If x is a certain feature, it is calculated by 0 – 1 scaling as follows:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}, \quad (15)$$

where x' expresses the standardized value. Credit data often have missing values. XGBoost comes with its own sparsity segmentation algorithm, which can learn the best way to deal with missing values and is more suitable for modelling than traditional methods of dealing with missing values. If there are outliers and noise in the data, standardization can indirectly avoid the influence of outliers, and centralization can deal with extreme values.

4.2.2. Feature Selection. First, the score of the relative feature importance with the initial hyperparameter is calculated,

and the redundant features are discarded by the feature selection algorithm. The importance of feature selection lies in eliminating redundant features, highlighting effective features, improving the calculation speed, and eliminating the influence of adverse features on the prediction results.

4.2.3. Training the Model. To ensure consistency between hyperparameter and the training dataset as much as possible, we perform cross-validation of the dataset. We tested several cross-validation methods. From many experimental results, we ultimately decided to use 10-fold cross-validation to divide the datasets.

First, the hyperparameters in the XGBoost model, including the maximum tree depth, subsample ratio, column subsample ratio, minimum child weight, maximum delta step, and gamma-delta, are the optimization targets, and the position of each particle is randomly initialized in the hyperparameter search space. Second, the particles are divided into adaptive populations. This step is achieved by calculating the local density of the particles and the distances to the particles whose local density is higher. According to the value determined by the position of the particle, we assign the hyperparameters of the XGBoost model and introduce the verification data into the model for prediction. Last, the loss function on the verification dataset is the fitness function of the particles. The simplified description of credit scoring is a two-category problem. If the labels of the

positive and negative samples of credit data are defined as $+1/-1$, the logistic loss function is defined as

$$L_{\text{logistic}} = \log(1 + \exp(-yp)), \quad (16)$$

where p is the predictive value and y represents the actual value. In this paper, as our model labels are 0 and 1, the logistic loss is expressed as follows:

$$L_{\text{logistic}} = -\frac{1}{N} \sum_{i=1}^N (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)). \quad (17)$$

The particles are divided into ordinary particles and optimal particles in accordance with the fitness value. Different update strategies update the information of the corresponding particles, and the algorithm checks whether the termination condition is reached; if so, we obtain the optimized value. If not, based on the positions of the particles, the model reclassifies the population again, calculates the fitness value, and updates the position information of each particle until the termination condition is reached. The optimal hyperparameters are utilized to construct the model, and the training and prediction are carried out with the data. The diagram of the model architecture is shown in Figure 2.

The algorithm steps are as follows:

- (1) Divide the datasets, train the data for the training model, and verify the data for prediction. Initialize the adaptive PSO algorithm.
Subgroups of the particle swarms are divided according to equations (11) and (12).
- (2) Take the logistic loss function as the fitness value, and calculate the fitness value of each particle according to (16). Build the XGBoost model with the corresponding hyperparameters determined by current best particle. Training and prediction of datasets, and the fitness value updated by the loss function given.
- (3) Determine the position of the global optimal particle p_{best} and the local optimal particle g_{best} according to the result of the population division and the fitness values of the particles.
- (4) According to (13) and (14), update the positions of the ordinary particles and locally optimal particles, respectively.
- (5) Judge whether to terminate. When the maximum number of iterations is n , return the optimal value of the hyperparameter; otherwise, return to 2.
- (6) Obtain the optimal hyperparameters to build the XGBoost model and calculate the indexes.

As shown in Figure 4, when the number of iterations is 200, the curve of APSO tends to converge. So, we set the maximum number of iterations $n = 200$.

The pseudocode of the APSO-XGBoost is given in Figure 3, while a short description is presented as follows:

First, the initialization process of APSO starts; the particles with position X and velocity V in the search space are initialized as the basic input. Subsequently, the overall adaptive partitioning algorithm starts: the distance ρ and

density δ of each particle are calculated, and the particles with high ρ and δ are selected as the centres; the remaining particles are assigned; and the swarm is divided into C subgroups. XGBoost has nodes set I initialized on the training data, and the hyperparameters of the model are determined by the current optimal values. In the modelling, the gain of each node is calculated, and the node with the highest score is split to generate a tree. After the modelling is completed, the type of particle is updated based on the fitness value (loss function), and then, two types of particles are updated using different strategies in each subgroup. After exiting the optimization process, XGBoost with the best hyperparameter settings was applied to the test set.

5. Experimental Setup

In this section, we evaluate the performance of the APSO-XGBoost model by experiments. First, the credit dataset is introduced. Second, we introduce baseline models and set up the hyperparameters of the XGBoost group. Third, the evaluation scale is given. Finally, we compare the proposed APSO-XGBoost model with other commonly used ML models in terms of the evaluation scale.

5.1. Datasets. In this section, the performance of the model is verified by UCI credit datasets. Two credit datasets, German and Austrian, from the UCI machine learning repository (<http://archive.ics.uci.edu/ml/datasets.html>) are used. In addition to the above datasets, *P2P* credit data from two platforms (LendingClub in the US and Ren.com in China) were also used to verify the effectiveness of our model in providing decision support for *P2P* lending businesses and to verify the generalization of the model. The formulation of the data is listed in Table 2.

5.2. Feature Engineering. Feature engineering selects important characteristics and removes irrelevant features to build a model. It can greatly reduce the dimension disaster problem, improve the operational efficiency, reduce the difficulty in learning tasks, make the model simpler, and reduce the computational complexity.

By calculating the importance of the features, features that are more favorable to the model are selected. XGBoost offers three measures of feature importance: gain, cover, and frequency. The gain refers to the tree gain of the model, which accounts for the contribution of each feature; the feature is used to split the nodes and calculate the contribution to the model compared with the other characteristics, and the higher the value is, the better the feature is for predicting the importance. The cover is affected by the division of the average sample number. The frequency is for specific features in the model tree that explains the percentage of the relative number of gain characteristics in terms of the relative importance of the relevant properties. Therefore, we choose the gain as the feature importance property. XGBoost adopts the stochastic fractal search (SFS) algorithm according to the rank of the importance of the features and adds features into the dataset to form subsets

FIGURE 2: APSO-XGBoost.

Algorithm 1: XGBoost optimized by APSO

Input: Initializes particles $x_i = (x_i^1, x_i^2, \dots, x_i^D)$ with position $X_i = [x_i^1, x_i^2, \dots, x_i^D]$ and velocity $V_i = [v_i^1, v_i^2, \dots, v_i^D]$

Output: Optimal value of hyper-parameter

```

1 for  $i = 1; i \leq N; i = i + 1$  do
2   Compute the local density  $\delta_i$  and the distance  $\rho_i$ 
3   Choose particles with high  $\delta_i$  and relatively high  $\rho_i$  as centers according to  $\gamma_i = \rho_i * \delta_i$ 
4   Assign remaining particles and get  $C$  subgroups;
5   Initialize XGBoost with instance nodes set  $I$  on train data, the hyper-parameter  $\leftarrow$  current optimal value;
6   for  $k = 1; k \leq m; k = k + 1$  do
7      $gain \leftarrow 0, G = \sum_{i \in I} g_i \leftarrow 0, H = \sum_{i \in I} h_i \leftarrow 0$ 
8     for  $j$  in sorted ( $I$ , by  $x_{jk}$ ) do
9        $G_L \leftarrow G_L + g_L, H_L \leftarrow H_L + h_L$ 
10       $G_R \leftarrow G - G_L, H_R \leftarrow H - H_L$ 
11      score max (score,  $\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} + \frac{G^2}{H + \lambda}$ )
12   Update particle state ( $pBest_p, gBest$ ) refer to the loss function
13   for  $c = 1; c \leq C; c = c + 1$  do
14     if particle is local optimal then
15        $v_i^d = \omega v_i^d + c_1 rand_1^d (pbest_1^d - x_i^d) + c_2 rand_2^d (\frac{1}{C} \sum_{c=1}^C gbest_c^d - x_i^d)$ 
16        $x_i^d = x_i^d + v_i^d$ 
17     else
18        $v_i^d = \omega v_i^d + c_1 rand_1^d (pbest_1^d - x_i^d) + c_2 rand_2^d (cgbest_c^d - x_i^d)$ 
19        $x_i^d = x_i^d + v_i^d$ 

```

FIGURE 3: The pseudocode of APSO-XGBoost.

TABLE 2: Formulation of credit data.

Dataset	Samples	Features	Training set	Test set	Good/bad
German	1000	24	800	200	700/300
Australian	690	14	552	138	307/383
P2P-LC	2642	11	2114	528	1322/1320
P2P-Ren	1421	17	1137	284	1072/349

one by one. With the default hyperparameter of XGBoost, the subset that minimizes the logistical loss is selected as the subset of the features after 10-fold cross-validation.

5.3. Baseline Models. To explore the performance differences between XGBoost and other methods, the first set of experiments is arranged as follows. Ten competitive algorithms that are widely used for credit scoring, including statistical algorithm LR, traditional ML algorithms (DT, LR, NN, SVM, and RF), and ensemble learning algorithms (AdaBoost, AdaBoost-NN, Bagging-DT, Bagging-NN, and GBDT), were tested for comparison with default XGBoost. Moreover, to reveal the effectiveness of the APSO for optimizing the XGBoost model, the proposed model is compared with XGBoost models optimized by several

hyperparameter optimization methods, including XGBoost-GS, XGBoost-RS, XGBoost-TPE, and PSO-XGBoost. In order to ensure the fairness of the experiment, while taking into account the accuracy and computational complexity of the model, the number of iterations of the hyperparametric algorithm is 200. The number of computations is the number of times that the hyperparametric algorithm calls the objective function in an iteration and the GS method based on a traversal idea, so the number of computations is determined by the parameter space. The search space and number of computations of each optimization method set are described in Table 3.

The baseline models are described below.

NN: it refers to neural principles, where each neuron can be regarded as a learning unit. The NN is constructed on the basis of many neurons, which are composed of an input layer, hidden layer, and output layer. These neurons take certain characteristics as input and obtain output according to their own model. For the NN model of credit scoring, its input is the applicant's attribute vector, and the output is the default or nondefault category, +1 or -1. The weight assigned to each attribute varies according to its relative importance, and the weight is adjusted iteratively to make the predicted output closer to the actual target.

SVM: by mapping the feature vector of an instance to a point in space, the purpose of the SVM is to draw a line to best distinguish the two types of points. In credit scoring, the data are correctly divided into the default type and the nondefault type. The SVM finds the hyperplane that separates the data. To best distinguish the data, the sum of the distances from the closest points on both sides of the hyperplane is required to be as large as possible.

DT: this is commonly used in credit scoring. The DT is a process for classifying instances based on features, where each internal node represents a judgement on an attribute, each branch represents the output of a judgement result, and each leaf node represents a classification result. The classification result in credit scoring is default or nondefault. The decision-making algorithm loops all splits and selects the best-partitioned subtree based on the error rate and the cost of misclassification.

RF [40]: the principle of random forest is to generate multiple decision tree models, where each tree learns and makes predictions independently on the bootstrap sample. By adding the results of each decision tree, the result with the most votes is selected as the final prediction result.

LR: the statistical technique of logistic regression is usually used to solve binary classification problems, and it is regarded as the benchmark of credit scoring [28]. First, regression analysis is used to describe the relationship between the independent variable x and the dependent variable Y and to predict the dependent variable Y . LR adds a logistic function on the basis of regression. In credit scoring, the probability of future results is predicted by the performance of historical data from applicants. The goal of LR is to judge whether the feature vectors of customers belong to a certain category, default or nondefault.

Bagging and AdaBoost are two popular methods in ensemble learning.

Bagging [41]: this method randomly samples and substitutes a training set with m samples to generate multiple new datasets and then builds models for each bootstrap sample. These models are merged through a certain strategy, such as majority voting.

Among the boosting methods, the most important include AdaBoost and GBDT.

AdaBoost [42]: the misclassified samples in the training set are assigned greater weights. AdaBoost sequentially combines weak learners into strong learners to boost the model. Based on the error of the previous classifier, AdaBoost adjusts the weight of the sample. Through the iterations, it gives larger weights to the misclassified samples in the training set.

GBDT: this is an iterative decision tree algorithm. The conclusions of all trees are added to obtain the final answer. The core of the algorithm is to use the most rapidly descending approximation, which is the negative gradient value of the loss function, as an approximate value of the residual of the algorithm to fit a regression or classification tree.

The configurations of the parameters of the traditional ML algorithms and ensemble learning used employed in the experiments are displayed in Table 4.

5.4. Evaluation Measures. In credit scores, the average accuracy is one of the most popular evaluation indices and represents the overall performance of the model. To better explore the ability of the model to distinguish between nondefault applications and default applications, type I errors and type II errors in the confusion matrix (see Table 5) are often applied to evaluate the models to predict their performance in detail. A type I error occurs when a default loan application is being wrongly classified as a nondefault loan application. Conversely, a type II error occurs when a nondefault is misclassified as default. TP and TN in the confusion matrix represent the numbers of correctly classified good borrowers and bad borrowers, respectively. FP and FN represent the numbers of misclassified loan applications.

The formulas are defined as follows.

The average accuracy (ACC):

$$\frac{TP + TN}{TP + FP + TN + FN} \quad (18)$$

The type I errors:

$$\frac{FP}{TN + FP} \quad (19)$$

The type II errors:

$$\frac{FN}{TP + FN} \quad (20)$$

The Brier score measures the accuracy of the predicted probability and the calibration of the prediction performance. The Brier score ranges from 0 to 1, and the interval value represents probabilistic predictions from perfect to poor. The Brier score is defined as follows:

TABLE 3: Search space set and number of computations of each hyperparameter optimization method.

Hyperparameter	APSO, PSO, TPE, RS	GS	Default
Learning rate	0.1	0.1	0.1
Number of boosts	60	60	60
Maximum tree depth	(1, 12)	1, 2, 3	0.2 * number of features
Subsample ratio	(0.9, 1)	0.9, 0.95, 1	0.9
Column subsample ratio	(0.9, 1)	0.9, 0.95, 1	0.9
Minimum child weight	(0, 4)	0, 1, 2, 3, 4	2
Maximum delta step	(0, 1)	0.4, 0.6, 0.8, 1	1
Gamma	(0, 0.01)	0, 0.01	1
Number of computations	200	2160	—

TABLE 4: Parameters of all algorithms in the experiments.

Algorithm	Parameters
DT	min_samples_leaf = 6 max_depth = 8 min_samples_split = 2 Gamma = 0.0 max_leaf_nodes = 4
LR	No parameters specified Epoch = 1000
NN	Learning_rate = 0.01 Hidden_units = 5 Kernel: RBF C = 32 Gamma = 0.1 n_estimator = 100
AdaBoost, Bagging-DT, RF	The remaining parameters are the same as DT n_estimator = 100
AdaBoost-NN, Bagging-NN	The remaining parameters are the same as NN n_estimators = 100 Subsample = 0.9
GBDT	learning_rate = 0.1 min_samples_leaf = 1 max_depth = 7

TABLE 5: Confusion matrix.

Actual value	Nondefault		Default	
Predicted value	Nondefault	TP	FN	TP + FN
	Default	FP	TN	FP + TN
		TP + FP	FN + TN	TP + FP + FN + TN

$$BS = \frac{1}{N} \sum_{i=1}^N (p_i - y_i)^2, \quad (21)$$

where N is the number of samples and p_i and y_i denote the probability prediction and the true label, respectively, of sample i .

The $F1$ -score takes into account both precision and recall of classification models. It is the harmonic average of these two indicators, and it ranges from 0 to 1:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \quad (22)$$

where precision is the proportion of positive samples in positive cases, and it is defined as

$$\text{precision} = \frac{TP}{TP + FP}. \quad (23)$$

Also, recall is the proportion of predicted positive cases in the total positive cases, and it defined as

$$\text{recall} = \frac{TP}{TP + FN}. \quad (24)$$

5.5. Comparisons among Hyperparameter Optimization Methods. To demonstrate the performance of the tuning strategies, Figure 4 shows the convergence curves of average loss function of parameter optimization methods over four credit data. The ordinate represents average minimum value of loss function, and the abscissa represents the number of iterations. It can be seen from the figure that the convergence

FIGURE 4: Plot of the average minimum value of loss function of each parameter optimization method with respect to iterations ($n = 300$).

speed of GS and RS is slower. TPE converges faster, and its result is better than GS and RS. The value of the loss function of PSO is slightly lower than that of TPE, but the convergence speed of PSO is much faster. PSO and APSO have the fastest convergence speed among the parameter optimization methods. PSO has entered the convergence state early, which leads to its unsatisfactory final error rate and failure to find the global optimal value. APSO gets the best performance among all the converging performance. APSO still continues to decline after some iterations, indicating that the optimization mechanism helps subgroups increase diversity, prevents local particles from clustering, and helps particles to find the local optimal value.

6. Results and Discussion

The ACC is one of the most mainstream and intuitive indicators. The ACC indicates the overall predictive ability of the model. In credit scoring, a small improvement in performance may help institutions reduce their losses by helping them with a large number of applications at risk of default.

In the German dataset (see Table 6), the XGBoost model obtained the best value of ACC (76.85%). Although the single classifier NN and Bagging-NN perform best in terms of type I error, the XGBoost model achieves better results with the other four indicators than them. For the type II error, SVM achieves the best value, but SVM does not maintain consideration for classification performance because it has the worst type I error rate of all models. In addition, the XGBoost model achieved satisfactory Brier score and $F1$ -score.

For the Australia dataset (see Table 7), the XGBoost credit scoring model achieves the best performance in all measurements except the type I error rate. LR has the best performance for type I error but is the worst performance for the type II error, which shows that, with this balanced dataset, LR is inclined to determine the sample as a bad application, which is not acceptable for exploring the potential applicants. XGBoost obtained the best results of Brier score and its $F1$ -score which is higher than that of second-best classifier Bagging-NN by 0.8%.

In the balanced $P2P$ -LC dataset (see Table 8), the ACC of XGBoost reaches the best (66.70%). XGBoost achieves the lowest Brier score and best $F1$ -score as XGBoost changed the training loss function. The XGBoost assigns weights to samples with a small proportion, which affects the loss of the sample's contribution during the training process, which in turn changes the first-order derivative and the second-order derivative.

For the $P2P$ -ren dataset (see Table 9), the XGBoost model is only second to GBDT regarding type I error rate; all other indicators exceed those of GBDT. LR, SVM, and AdaBoost obtain a low type II error rate but have very serious imbalance misclassifications as their maximum type I error is 90%. The type I error and type II error of XGBoost are acceptable, which demonstrates that XGBoost sufficiently learns imbalanced data.

Overall, the ensemble classifiers perform better than the single model on the four datasets on average. The ensemble classifier generates better prediction accuracy by combining individual weak classifiers with different performances. Under the premise of ensuring accuracy, XGBoost is able to take into account various indicators. In credit scoring, each indicator is meaningful for making decisions. For example, a low type I error indicates that the model has a stronger ability to discriminate against nonperforming loan applications. The model with a type II error is likely to maximize the profits of credit institutions by better exploiting potential creditworthy applicants. XGBoost achieved the highest $F1$ -score over four credit datasets, which indicates that the model is less affected by the data imbalance than other models. The satisfactory performances on the Brier score indicate that the XGBoost model obtained a more accurate probability value prediction than other baseline models.

Table 10 summarizes the results of XGBoost models using different optimization algorithms, which indicates that the XGBoost that was optimized based on APSO has achieved promising performance with the four credit datasets.

For the German dataset, APSO-XGBoost algorithms achieved the best prediction accuracy with a score of 77.48%, which is 0.8% higher than the default XGBoost model. The proposed model ranks second place in type I error, only second to XGBoost-GS, but it surpasses XGBoost-GS in terms of other indicators. The model obtains acceptable results for type II and Brier score. Moreover, the proposed model obtained the best $F1$ -score, which demonstrates that the model can adequately learn the unbalanced data.

For the Australia dataset, APSO-XGBoost obtained the highest ACC and lowest type I error rate. A low type I error indicates that the model prevents misjudging a poor credit application with a higher probability of good credit and likely to reduce an enterprise's high debt rate risk. APSO-XGBoost achieves the third-lowest Brier score below that of XGBoost-RS and XGBoost-TPE. In addition, the $F1$ -score of APSO-XGBoost is the best as the more appropriate setting obtained by APSO. This finding can be explained by the hyperparameter of XGBoost Maximum Delta Step, which resists the problem of unbalanced data to a certain

TABLE 6: Result of performances of XGBoost and baseline methods on the German dataset.

Model	ACC (%)	Type I error (%)	Type II error (%)	Brier score	F1-score
DT	72.65	59.37	13.63	0.2257	0.5763
LR	76.43	52.26	11.28	0.1627	0.6195
NN	72.54	50.13	17.75	0.2363	0.6571
SVM	76.07	63.73	6.88*	0.1631	0.5762
AdaBoost	73.61	56.57	13.46	0.1779	0.5994
AdaBoost-NN	73.75	49.95	16.09	0.2237	0.6375
Bagging-DT	75.19	58.19	10.51	0.1718	0.5943
Bagging-NN	76.01	49.67*	12.98	0.1738	0.6075
RF	75.92	58.59	9.29	0.1646	0.6122
GBDT	76.59	51.37	11.42	0.1615	0.7246
XGBoost	76.85	52.07	10.47	0.1505	0.7496

TABLE 7: Results of the measured performance of models for the Austrian dataset.

Model	ACC (%)	Type I error (%)	Type II error (%)	Brier score	F1-score
DT	84.51	17.41	13.95	0.1359	0.8543
LR	86.77	8.67*	16.88	0.1019	0.7865
NN	85.27	12.06	16.87	0.1111	0.8612
SVM	85.54	15.35	13.74	0.1012	0.3961
AdaBoost	85.64	17.40	11.93	0.1034	0.8644
AdaBoost-NN	84.59	14.22	16.36	0.1174	0.8615
Bagging-DT	86.42	13.33	13.78	0.0987	0.4926
Bagging-NN	85.62	11.83	16.42	0.1062	0.8683
RF	87.41	13.26	12.05	0.0971	0.7571
GBDT	86.14	13.43	14.19	0.0991	0.6026
XGBoost	87.58	12.97	11.79	0.0908	0.8756

extent. APSO found a more accurate value of the hyperparameter.

For the *P2P-LC* dataset, APSO-XGBoost is the best compared to other optimized credit models for all indicators, which indicates that the hyperparameter settings are more reasonable than other model settings and promote the combination of the model tree structure and credit characteristics. The APSO-XGBoost model has obvious improvement over PSO-XGBoost in each indicator. APSO can further explore the predictive performance of the XGBoost model.

In the *P2P-ren*, APSO has also reached the best in all indicators except for the type II error rate. Although the default nonoptimized model has the lowest type II error rate, the type I error rate is too high because of the unbalanced data that have more labels with good credit. Poor hyperparameter settings hinder the model's ability to sufficiently learn to resolve the situation. APSO-XGBoost also achieves the best capability of probability prediction as it obtains the highest Brier score.

APSO-XGBoost perform better than most models on average. The results show that the APSO can promote the matching of the XGBoost structure with the characteristics of the credit data. From the performance of ACC and low type errors, PSO-XGBoost is slightly better than XGBoost-TPE. XGBoost-TPE also achieved acceptable performance with various datasets, especially German datasets, which demonstrate that it is an acceptable model for credit scoring. The APSO-XGBoost model is better than the PSO-XGBoost model, which indicates that APSO overcomes the local

optimality of the particle swarm to a certain extent and allows particles to obtain hyperparameters that render the model more accurate.

6.1. Statistical Significance Analysis. We use the Friedman nonparametric test and Nemenyi post hoc test to verify that statistical difference of models in the accuracy over four credit datasets. The null hypothesis is that there is no difference between the models. First, the Friedman test rejects the null hypothesis which means that models are significant difference. Next, the Nemenyi post hoc test is applied to evaluate the differences among individual models. When the averaged ranks differ by at least a critical difference (CD), the result indicates the difference between models. When the number of samples is 4, the number of algorithms is 16, and the critical value q_α is 3.196 from the Nemenyi critical value table with a significance level of 0.1. The critical difference is computed as 10.759 according to formula (25): where N is the number of sample and K is the number of the algorithms.

In Figure 5, the bars represent the average rank of the models over four datasets. APSO-XGBoost is the benchmark model whose average rank is 1, the horizontal line that represents the threshold of CD computed as 11.759. The average rank of XGBoost models is higher than other models, and our model has reached the top. The models above the horizontal line which includes AdaBoost, SVM, AdaBoost-NN, DT, and NN are significantly worse than the APSO-XGBoost model.

TABLE 8: Result of performances of XGBoost and baseline methods on the *P2P-LC* dataset.

Model	ACC (%)	Type I error (%)	Type II error (%)	Brier score	F1-score
DT	60.11	46.03	33.74	0.2549	0.3796
LR	64.74	41.37	29.14*	0.2247	0.6023
NN	63.65	32.22	40.49	0.2279	0.6011
SVM	60.67	41.29	37.36	0.2331	0.6083
AdaBoost	61.25	40.18	37.32	0.2336	0.6102
AdaBoost-NN	64.09	33.61	38.22	0.2251	0.6483
Bagging-DT	62.43	37.43	37.71	0.2328	0.4857
Bagging-NN	65.34	34.07	35.25	0.2198	0.6082
RF	63.2	35.72	37.88	0.2277	0.5601
GBDT	66.25	30.9	36.59	0.2166	0.4784
XGBoost	66.70	30.02	36.56	0.2125	0.6559

TABLE 9: Result of performances of XGBoost and baseline methods on the *P2P-Ren* dataset.

Model	ACC (%)	Type I error (%)	Type II error (%)	Brier score	F1-score
DT	76.01	63.27	11.21	0.1941	0.7986
LR	75.07	92.3	2.99	0.1637	0.8155
NN	75.4	69.01	10.14	0.1599	0.8023
SVM	75.17	97.18	1.27	0.1813	0.8025
AdaBoost	75.75	94.99	1.22*	0.1728	0.6523
AdaBoost-NN	75.25	64.79	11.72	0.1703	0.8072
Bagging-DT	79.42	66.87	5.51	0.1465	0.7783
Bagging-NN	75.83	76.99	6.97	0.1504	0.8145
RF	80.59	60.49	6.03	0.1355	0.7911
GBDT	83.52	42.2	8.1	0.1212	0.8168
XGBoost	83.70	59.10	6.01	0.1170	0.8265

TABLE 10: Result of the performances of XGBoost with various optimization methods on credit data.

Data	Model	ACC (%)	Type I error (%)	Type II error (%)	Brier score	F1-score (%)
<i>German</i>	—	—	—	—	—	—
	XGBoost	76.85	52.07	10.47	0.1505	74.96
	XGBoost-GS	76.83	49.79	11.76	0.1176	76.51
	XGBoost-RS	77.18	53.73	9.57	0.0957	75.61
	XGBoost-TPE	77.34	53.71	9.35	0.0935	75.74
	PSO-XGBoost	77.36	53.01	10.15	0.1170	76.21
	APSO-XGBoost	77.48	52.06	9.98	0.1070	77.96
<i>Australia</i>	—	—	—	—	—	—
	XGBoost	87.58	12.97	11.79	0.0908	87.56
	XGBoost-GS	87.81	13.92	10.80	0.0915	87.68
	XGBoost-RS	87.82	12.64	11.82	0.0893	87.99
	XGBoost-TPE	87.92	12.67	11.61	0.0890	87.01
	PSO-XGBoost	87.98	12.58	11.90	0.0875	87.43
	APSO-XGBoost	88.20	11.80	11.78	0.0860	88.12
<i>P2P-LC</i>	—	—	—	—	—	—
	XGBoost	66.70	30.02	36.56	0.2125	65.59
	XGBoost-GS	66.31	31.8	35.58	0.2143	65.60
	XGBoost-RS	67.08	29.78	36.06	0.2096	65.63
	XGBoost-TPE	66.97	29.82	36.23	0.2095	65.78
	PSO-XGBoost	66.99	29.85	35.97	0.2104	66.01
	APSO-XGBoost	67.63	29.15	35.56	0.2092	67.53
<i>P2P-ren</i>	—	—	—	—	—	—
	XGBoost	83.70	59.10	6.01	0.1170	82.65
	XGBoost-GS	83.83	41.19	8.03	0.1105	82.27
	XGBoost-RS	84.46	40.57	7.39	0.1070	83.02
	XGBoost-TPE	84.65	39.68	7.43	0.1067	87.10
	PSO-XGBoost	84.66	39.66	7.40	0.1066	87.08
	APSO-XGBoost	84.72	39.65	7.20	0.1060	88.12

FIGURE 5: Average rank of models and the threshold of Nemenyi test (significance levels $p = 0.1$).

$$CD = q_{\alpha} \sqrt{\frac{K(K+1)}{6N}}, \quad (25)$$

7. Conclusions

The core of ensemble learning is to gather a series of learners and combine them to form a strong learner. This reduces errors and achieves the goal of accurate prediction. The accuracy of financial credit prediction directly determines the revenue of a financing institution. The integrated model XGBoost is based on the GBDT and adds regularization constraints, improved loss functions, etc.; it obtains acceptable prediction results in the field of credit prediction. The structure of the model depends on the XGBoost hyperparameter setting. To obtain the optimal hyperparameters and improve the accuracy of the model, our model construction process is divided into three main steps: first, the data are preprocessed and standardized. Second, feature engineering removes redundant features. Third, we use the APSO algorithm to optimize the hyperparameters and set the optimal hyperparameters obtained by training. Last, the trained model tests two UCI datasets and two P2P datasets with four evaluation indicators. The results show that the proposed APSO-XGBoost model has the best ranking among all classifiers in terms of the ACC. The experimental results show that the performance of APSO-XGBoost improves the credit dataset compared to the performance of XGBoost optimized by RS, GS, and TPE. A comparison with the other baseline models shows that our model is superior in credit scoring; furthermore, the type I error rate of our model is almost lower than that of the XGBoost models optimized with

other hyperparameter optimizations. It is shown that the choice of hyperparameter optimization matched with the classifier is reasonable and effective, and our model has a stronger guiding significance to assist institutions in avoiding the risk of loss.

In future work, we will mainly focus on imbalanced data research.

Data are the basis of credit scoring research. A large portion of public data is class imbalance, which limits the research scope. The most related works have been oriented towards supervised algorithms or the construction of ensemble methods for increasing the classification accuracy rates. Supervised algorithms require a large amount of data. Semisupervised learning (SSL) provides an efficient solution since only a small amount of data are needed to achieve similar or even better learning ability and more robust classification. Karlos et al. [43] compared many semi-supervised schemes against famous supervised algorithms with data of Greek firms. The promising results show that SSL is effective in mitigating with the imbalance of enterprise bankruptcy data, which also shows that SSL is feasible for solving the same problem of financial credit scoring. Moreover, Fazakis et al. [44] propose a novel SSL scheme for self-labelled classification, which utilizes the efficacy of XGBoost to build a model with highly accuracy and robustness. Based on this research, we plan to apply XGBoost to estimate the labels for unlabelled data to overcome class imbalance and dynamically update the most confident prediction instances into the initial labelled training set by evaluating the classifier in each iteration.

Cost-sensitive learning (CSL) is another widely used solution to imbalance data classification; it is an ML method

with the minimum cost of misclassification. CSL implemented by instance-weighting or instance-relabelling is considered data-level solutions [6]. Akila [45] proposed a transaction window bagging model that uses a parallel bagging approach and incorporated with an incremental learning model, cost-sensitive base learner, and a weighted voting-based combiner to effectively address the data imbalance in credit data. Moreover, applying the idea of cost-sensitive boosting, we plan to introduce cost items into XGBoost. Incorporate cost as a performance metric with a regular indicator to tradeoff the decision-making which helps institutions determine whether grant a loan to an applicant or not.

Data Availability

The data used to support the findings of this study are available at <https://www.lendingclub.com/info/demand-and-credit-profile.action> and <https://www.renrendai.com/>.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant nos. 61972227, 61873117, and U1609218), in part by the Natural Science Foundation of Shandong Province (Grant nos. ZR201808160102 and ZR2019MF051), in part by the Primary Research and Development Plan of Shandong Province (Grant nos. GG201710090122, 2017GGX10109, and 2018GGX101013), and in part by the Fostering Project of Dominant Discipline and Talent Team of Shandong Province Higher Education Institutions.

Supplementary Materials

The Supplementary Materials provide the extended experiment to further prove the effectiveness of the proposed model APSO-XGBoost. The file includes datasets used, experimental setting, experimental results with corresponding analysis, and statistical tests. (*Supplementary Materials*)

References

- [1] B. D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, Cambridge, UK, 1996.
- [2] S.-L. Pang, "Study on credit scoring model and forecasting based on probabilistic neural network," *Xitong Gongcheng Lilun Yu Shijian/System Engineering Theory and Practice*, vol. 25, no. 5, pp. 43–48, 2005.
- [3] V. Vapnik, "Statistical learning theory," *Encyclopedia of the Sciences of Learning*, vol. 41, no. 4, p. 3185, 1998.
- [4] C.-L. Chuang and S.-T. Huang, "A hybrid neural network approach for credit scoring," *Expert Systems*, vol. 28, no. 2, pp. 185–196, 2011.
- [5] A. Diego Paganoti Fonseca, "A two-stage fuzzy neural approach for credit risk assessment in a Brazilian credit card company," *Applied Soft Computing*, vol. 92, 2020.
- [6] N. Chen, B. Ribeiro, and A. Chen, "Financial credit risk assessment: a recent review," *Artificial Intelligence Review*, vol. 45, no. 1, 2016.
- [7] C.-L. Huang, M.-C. Chen, and C.-J. Wang, "Credit scoring with a data mining approach based on support vector machines," *Expert Systems with Applications*, vol. 33, no. 4, pp. 847–856, 2007.
- [8] A. B. Hens and M. K. Tiwari, "Computational time reduction for credit scoring: an integrated approach based on support vector machine and stratified sampling method," *Expert Systems with Applications*, vol. 39, no. 8, pp. 6774–6781, 2012.
- [9] J. Luo, Y. Xin, and Y. Tian, "Unsupervised quadratic surface support vector machine with application to credit risk assessment," *European Journal of Operational Research*, vol. 280, no. 3, pp. 1008–1017, 2019.
- [10] L.-J. Kao, C.-C. Chiu, and F.-Y. Chiu, "A bayesian latent variable model with classification and regression tree approach for behavior and credit scoring," *Knowledge-Based Systems*, vol. 36, pp. 245–252, 2012.
- [11] Y. Xia, J. Zhao, L. He, Y. Li, and M. Niu, "A novel tree-based dynamic heterogeneous ensemble method for credit scoring," *Expert Systems with Applications*, vol. 159, Article ID 113615, 2020.
- [12] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.
- [13] G. Wang, J. Hao, J. Ma, and H. Jiang, "A comparative assessment of ensemble learning for credit scoring," *Expert Systems with Applications*, vol. 38, no. 1, pp. 223–230, 2011.
- [14] N.-C. Hsieh and L.-P. Hung, "A data driven ensemble classifier for credit scoring analysis," *Expert Systems with Applications*, vol. 37, no. 1, pp. 534–545, 2010.
- [15] H. Xiao, Z. Xiao, and Y. Wang, *Ensemble Classification Based on Supervised Clustering for Credit Scoring*, Elsevier Science Publishers B. V, Amsterdam, Netherlands, 2016.
- [16] Y. Xia, C. Liu, B. Da, and F. Xie, "A novel heterogeneous ensemble credit scoring model based on bstacking approach," *Expert Systems with Applications*, vol. 93mar, pp. 182–199, . edition, 2018.
- [17] M. Ala'Raj and M. F. Abbod, "Classifiers consensus system approach for credit scoring," *Knowledge-Based Systems*, vol. 104, no. jul, pp. 89–105, 2016.
- [18] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," 2016, <https://arxiv.org/abs/1603.02754>.
- [19] M. Zieba, S. K. Tomczak, and J. M. Tomczak, "Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction," *Expert Systems with Applications*, vol. 58, pp. 93–101, 2016.
- [20] Y. C. Chang, K. H. Chang, and G. J. Wu, "Application of eXtreme gradient boosting trees in the construction of credit risk assessment models for financial institutions," *Applied Soft Computing*, vol. 73, pp. 914–920, 2018.
- [21] Y. Xia, C. Liu, and N. Liu, "Cost-sensitive boosted tree for loan evaluation in peer-to-peer lending," *Electronic Commerce Research and Applications*, vol. 24, pp. 30–49, 2017.
- [22] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.
- [23] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in Proceedings*

- of *Advances in Neural Information Processing Systems*, pp. 2546–2554, MIT press, Cambridge, MA, USA, 2011.
- [24] R. C. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proceedings of the MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, October 1995.
 - [25] F. Shen, X. Zhao, Z. Li, K. Li, and Z. Meng, “A novel ensemble classification model based on neural networks and a classifier optimisation technique for imbalanced credit risk evaluation,” *Physica A: Statistical Mechanics and its Applications*, vol. 526, Article ID 121073, 2019.
 - [26] Q. Yang, W. N. Chen, J. D. Deng et al., “A level-based learning swarm optimizer for large-scale optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 99, pp. 578–594, 2018.
 - [27] A. Rodriguez and A. Laio, “Clustering by fast search and find of density peaks,” *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.
 - [28] S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas, “Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research,” *European Journal of Operational Research*, vol. 247, no. 1, pp. 124–136, 2015.
 - [29] D. S. C. Nascimento, A. L. V. Coelho, and A. M. P. Canuto, “Integrating complementary techniques for promoting diversity in classifier ensembles: a systematic study,” *Neurocomputing*, vol. 138, pp. 347–357, 2014.
 - [30] A. X. Dastile, “Statistical and machine learning models in credit scoring: a systematic literature survey,” *Applied Soft Computing*, vol. 91, Article ID 106263.
 - [31] S. Oreski and G. Oreski, “Genetic algorithm-based heuristic for feature selection in credit risk assessment,” *Expert Systems with Applications*, vol. 41, no. 4, pp. 2052–2064, 2020.
 - [32] S. Maldonado, C. Bravo, J. López, and J. Pérez, “Integrated framework for profit-based feature selection and SVM classification in credit scoring,” *Decision Support Systems*, vol. 104, pp. 113–121, 2017.
 - [33] S. Cai and J. Zhang, “Exploration of credit risk of P2P platform based on data mining technology,” *Journal of Computational and Applied Mathematics*, vol. 372, Article ID 112718, 2020.
 - [34] C. Wang, R. Xu, J. Qiu et al., “AdaBoost-inspired multi-operator ensemble strategy for multi-objective evolutionary algorithms,” *Neurocomputing*, vol. 384, pp. 243–255, 2019.
 - [35] V. García, A. I. Marqués, J. S. Sánchez, and Salvador, “Exploring the synergetic effects of sample types on the performance of ensembles for credit risk and corporate bankruptcy prediction,” *Information Fusion*, vol. 47, pp. 88–101, 2019.
 - [36] J. H. Friedman, “Machine,” *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
 - [37] S. Chen, J. Montgomery, and A. Bolufé-Röhler, “Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution,” *Applied Intelligence*, vol. 42, no. 3, pp. 514–526, 2015.
 - [38] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in Neural Information Processing Systems*, vol. 4, 2012.
 - [39] W. HuZ. Li et al., “A simpler and more effective particle swarm optimization algorithm,” *Journal of Software*, vol. 18, no. 4, pp. 861–868, 2007.
 - [40] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
 - [41] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
 - [42] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
 - [43] S. Karlos, S. Kotsiantis, N. Fazakis, and K. Sgarbas, “Effectiveness of semi-supervised learning in bankruptcy prediction,” in *Proceedings of the 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Chalkidiki, Greece, July 2016.
 - [44] N. Fazakis, G. Kostopoulos, S. Karlos, S. Kotsiantis, and K. Sgarbas, “Self-trained eXtreme gradient boosting trees,” in *Proceedings of the 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, Patras, Greece, July 2019.
 - [45] S. Akila and R. Srinivasulu, “Parallel and incremental credit card fraud detection model to handle concept drift and data imbalance,” *Neural Computing and Applications*, vol. 31, pp. 1–12, 2018.