Research Article

Probability Density Machine: A New Solution of Class Imbalance Learning

Ruihan Cheng ¹, Longfei Zhang ¹, Shiqi Wu ¹, Sen Xu ², Shang Gao ^{1,3},
and Hualong Yu ^{1,3}

¹School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China

²School of Information Technology, Yancheng Institute of Technology, Yancheng, China

³Artificial Intelligence Key Laboratory of Sichuan Province, Sichuan University of Science and Engineering, Yibin, China

Correspondence should be addressed to Hualong Yu; yuhualong@just.edu.cn

Received 20 June 2021; Revised 28 August 2021; Accepted 2 September 2021; Published 9 September 2021

Academic Editor: Cristian Mateos

Copyright © 2021 Ruihan Cheng 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.

Class imbalance learning (CIL) is an important branch of machine learning as, in general, it is difficult for classification models to learn from imbalanced data; meanwhile, skewed data distribution frequently exists in various real-world applications. In this paper, we introduce a novel solution of CIL called Probability Density Machine (PDM). First, in the context of Gaussian Naive Bayes (GNB) predictive model, we analyze the reason why imbalanced data distribution makes the performance of predictive model decline in theory and draw a conclusion regarding the impact of class imbalance that is only associated with the prior probability, but does not relate to the conditional probability of training data. Then, in such context, we show the rationality of several traditional CIL techniques. Furthermore, we indicate the drawback of combining GNB with these traditional CIL techniques. Next, profiting from the idea of K-nearest neighbors probability density estimation (KNN-PDE), we propose the PDM which is an improved GNB-based CIL algorithm. Finally, we conduct experiments on lots of class imbalance data sets, and the proposed PDM algorithm shows the promising results.

1. Introduction

Learning from imbalanced data is an important and hot topic in machine learning, as it has been widely applied to diagnose and classify diseases [1, 2], detect software defects [3, 4], analyze biology and pharmacology data [5, 6], evaluate credit risk [7], predict actionable revenue change and bankruptcy [8, 9], diagnose faults in the industrial procedure [10, 11], classify soil types [12, 13], and even predict crash injury severity [14] or analyze crime linkages [15]. Meanwhile, class imbalance learning (CIL) is also a challenging task. It is known that most existing supervised learning models are constructed on the theory of empirical risk minimization; thus, they always tend to favor the majority classes, but ignore the performance of the minority classes [16, 17].

In the recent two decades, hundreds of CIL algorithms have been proposed to address the imbalanced classification problem [18, 19]. Specifically, these CIL methods could be

roughly divided into three categories: data-level [20–27], algorithmic-level [28–35], and ensemble learning [36–42]. Data-level methods solve CIL problem by balancing the training data. Algorithmic-level methods directly modify classification models to adapt imbalanced data distribution. As for ensemble learning, it adopts either data-level or algorithmic-level approaches to combine with Bagging or Boosting paradigm for improving the accuracy and robustness of CIL.

To explore the reason why imbalanced data hurt the traditional predictive models, several previous works have indicated some reasons from different aspects [34, 35, 43, 44]. Most viewpoints consider both skewed density distribution and the empirical risk minimization learning rule causing the degradation of modeling quality. In fact, for imbalanced data, the density distributions of different classes often vary sharply; however, their probability densities are generally invariable. This means if we could estimate the probability density distribution of each class

accurately, it would become easier to solve CIL problem. Motivated by this viewpoint, we develop a novel solution called Probability Density Machine (PDM) for addressing CIL problem in this study.

Firstly, to verify the validity of the idea, we analyze the reason why imbalanced data distribution hurts the performance of predictive model in theory, in the context of Gaussian Naive Bayes (GNB) predictive model [45, 46]. It is indicated that the negative impact of class imbalance only is associated with the prior probability (density distribution), but it is not related to the conditional probability (probability density distribution) of training data, which verifies our argument. Then, in the same theoretical framework, we explain the rationality of several traditional CIL techniques, including data-level and algorithm-level methods. Next, we indicate that the conditional probability estimation in GNB is generally inaccurate as the data might be far from the Gaussian distribution. Furthermore, we introduce the K-nearest-neighbors-probability-density-estimation- (KNN-PDE-) [47] alike algorithm to improve the accuracy of estimating probability density distribution and hereby propose the PDM algorithm. Finally, the effectiveness and feasibility of the proposed PDM model are verified by lots of experiments. In comparison with several traditional CIL solutions, the PDM shows promising results.

The rest of this paper is organized as follows. Section 2 reviews related work regarding CIL techniques briefly. Section 3 describes the CIL problem and analyzes the reason why class imbalance distribution hurts the performance of predictive model in the context of GNB and further explains the rationality of several traditional CIL solutions. In Section 4, we introduce an KNN-PDE-alike algorithm and then explain how to use it to develop PDM algorithm. Section 5 presents the experimental results and provides the corresponding discussion. Finally, Section 6 concludes the paper and outlines future work.

2. Related Work

In the recent two decades, several hundreds of different CIL methods have been proposed. These approaches can be roughly divided into three groups: data-level, algorithmic-level, and ensemble learning. Next, we wish to briefly review the work related to each of these three groups, respectively.

Data-level, which is also called sampling, addresses CIL problem by rebalancing data distribution. It either increases the number of minority instances (oversampling) [20–23], or decreases the number of majority instances (undersampling) [24, 25], or uses both strategies simultaneously (hybrid sampling) [26, 27]. Random undersampling (RUS) [20] and random oversampling (ROS) [20] are the simplest sampling methods, where the former randomly removes majority instances, while the latter randomly copies minority samples. These two methods both have inherent drawbacks; i.e., RUS loses much information which is helpful for modeling, and ROS tends to make the classification models overfitting. The Synthetic Minority Oversampling Technique (SMOTE) [21] is an effective method to solve the problems brought by RUS and ROS. It generates synthetic minority instances between

two adjacent real minority examples. However, SMOTE is inclined to propagate noises, which will also degrade the quality of training data. In recent years, some improved SMOTE algorithms have been proposed, including MWMOTE [22] that combines clustering technique and SMOTE to make new instances generate around those hard-to-learn samples and avoid emerging on wrong positions; MCT [23] which alters the class distribution of training data by cloning each minority class instance according to the similarity between it and the mode of the minority class; SMOTE-RSB [26] that takes advantage of rough set theory as a cleaning tool to search and remove minority noisy samples generated by SMOTE; and SMOTE-IPF [27] which adopts ensemble-based filters technique to prevent noise propagation. To avoid the information loss caused by RUS, Yu et al., [25] also integrate ant colony optimization into the procedure of undersampling to adaptively search those most important minority examples. All in all, the data-level technique has two major merits: (1) it is easier to be implemented as only data are manipulated; (2) it is irrelevant with classification models.

Unlike data-level methods, the algorithmic-level approaches focus on modifying the classification model itself to make it adapt class imbalance environment. Cost-sensitive learning and threshold moving strategy are the most popular algorithmic-level CIL methods. Cost-sensitive learning adapts the imbalanced data distribution by assigning different misclassification penalties for the instances belonging to different classes. The allocation of misclassification penalties might be either empirical [28, 30] or personalized [29, 31–33]. In addition, many supervised learning models can be made to be sensitive to cost, including neural networks [28], support vector machine [29, 31–33], and Bayesian network [30]. Threshold moving adopts a postprocessing strategy to force the classification hyperplane trained by any traditional supervised learning model to move toward the area of majority instances [34, 35]. The moving distance can be determined by either empiricism or optimization technique.

Ensemble learning is not an independent strategy to address CIL problem, but only a solution to promote the performance of other CIL methods. That is to say, ensemble learning must combine with one of the data-level or algorithmic-level methods to solve CIL problem. For examples, RUSBoost integrates RUS into boosting ensemble learning paradigm [36]; SMOTEBagging combines SMOTE with bagging ensemble learning paradigm [37]; AdaCost [38] and AdaC series algorithms [39] organize cost-sensitive learning and boosting; and EnSVM-OTHR [34] puts threshold moving strategy into bagging. Some recent studies also focus on class imbalance in high-dimension environment [40] and focus on how to improve the diversity of base models in CIL ensemble [41, 42]. Although ensemble learning always has a high time-complexity, it has presented potential to improve the quality of CIL models in various real-world applications.

3. Problem Description

3.1. Gaussian Naive Bayes (GNB) Predictive Model. As a variant of the Naive Bayes (NB) model, GNB has been widely used to solve various real-world supervised learning issues

[45, 46]. Although both NB and GNB are based on Bayes theorem, there exists an essential difference between them; i.e., NB can be only used to modeling the data with discrete features, while GNB is extended to deal with the data in continuous feature space. In order to model data with continuous features, GNB needs to destroy the feature-independent hypothesis adopted by NB. Actually, GNB assumes in data the instances belonging to each class satisfy an independent Gaussian distribution whose mean and variance can be directly obtained by calculating feature space. Then for any one given instance x_i , it is easy to calculate its conditional probability $P(x_i|y)$ in a class y , and meanwhile it is not difficult to acquire the prior probability $P(y)$ by directly counting the ratio of instances belonging to the class y in training set. According to the Bayes theorem, its posterior probability $P(y|x_i)$ can be calculated by

$$P(y|x_i) = \frac{P(x_i|y)P(y)}{P(x_i|y)P(y) + P(x_i|\sim y)P(\sim y)}, \quad (1)$$

where $P(\sim y)$ and $P(x_i|\sim y)$ represent the prior probability of the $\sim y$ class and the conditional probability of the instance x_i in the class $\sim y$, respectively. Then, GNB decides the class label of x_i by comparing $P(y|x_i)$ and $P(\sim y|x_i)$ based on the following rule:

$$f(x_i) = \begin{cases} y, & \text{if } P(x_i|y) > P(x_i|\sim y), \\ \sim y, & \text{else.} \end{cases} \quad (2)$$

3.2. Why Class Imbalance Hurts GNB? Next, in the context of GNB, we wish to analyze the reason why imbalanced data distribution will hurt the performance of traditional predictive models. Specifically, suppose we face a binary-class imbalanced classification task, and the training set can be represented as $\Phi = \{(x_i, y_i) | x_i \in \mathcal{R}^n, y_i \in \{Y_1, Y_2\}, 1 \leq i \leq s\}$, where Y_1 and Y_2 denote the class labels of the majority and minority instances, respectively. According to the class label, Φ can be further split into two subsets $\Phi_1 = \{(x_i, y_i) | x_i \in \mathcal{R}^n, y_i = Y_1, 1 \leq i \leq s_1\}$ and $\Phi_2 = \{(x_i, y_i) | x_i \in \mathcal{R}^n, y_i = Y_2, 1 \leq i \leq s_2\}$, where $s = s_1 + s_2$ and $s_1 > s_2$. Furthermore, to simplify the analysis, we restrict the dimension of the feature space in Φ as 1, no matter if Y_1 or Y_2 satisfies a Gaussian distribution. Then, we can directly acquire the prior probability and the conditional probability of each class. According to the Bayesian formula, we can respectively calculate the posterior probability belonging to each class as follows:

$$P(Y_1|X) = \frac{P(X|Y_1)P(Y_1)}{P(X|Y_1)P(Y_1) + P(X|Y_2)P(Y_2)}, \quad (3)$$

$$P(Y_2|X) = \frac{P(X|Y_2)P(Y_2)}{P(X|Y_1)P(Y_1) + P(X|Y_2)P(Y_2)}.$$

Based on the assumption above, Figure 1 shows why imbalanced data distribution can hurt the performance of predictive models.

As shown in Figure 1, the classification boundary has been obviously pushed toward the minority class, owing to

FIGURE 1: An example explaining why imbalanced data distribution hurts the performance of prediction models.

the fact that $P(Y_1)$ is larger than $P(Y_2)$. Considering the context of GNB, the classification boundary associates with the condition $P(Y_1|X) = P(Y_2|X)$ which can be equivalently expressed as $P(X|Y_1)P(Y_1) = P(X|Y_2)P(Y_2)$; hence if $P(Y_1) > P(Y_2)$, then it means only when $P(X|Y_1) < P(X|Y_2)$, we can select the corresponding X as the classification boundary. Therefore, we can say that the negative impact of class imbalance for traditional supervised learning models only associates with the difference of prior probability (density distribution) belonging to different classes, but is irrelevant to the conditional probability (probability density distribution) of each class in training data. In other words, class imbalance data possess probability density invariance. Ignoring the impact of prior probability, it is not difficult to deduce

$$\frac{P(Y_1|X)}{P(Y_2|X)} \propto \frac{P(X|Y_1)}{P(X|Y_2)}. \quad (4)$$

Any CIL solution tries its best to eliminate or at least alleviate the influence of prior probability, further making the model totally satisfy or approximately conform to the condition in (4).

3.3. The Explanation of Rationality of Several Traditional CIL Techniques. According to the theoretical analysis above, it is not difficult to explain the rationality of some traditional CIL techniques.

First, let us consider the data-level strategy. It solves the skewed density distribution problem by either increasing $P(Y_1)$, or decreasing $P(Y_2)$, or simultaneously increasing $P(Y_1)$ and decreasing $P(Y_2)$, further making $P(Y_1) = P(Y_2)$. Obviously, this strategy satisfies the condition in (4).

Next, cost-sensitive learning focuses on adjusting prior probability distribution by designating different costs for different classes. That means (3) can be transformed to

$$\begin{aligned} P(Y_1|X) &= \frac{P(X|Y_1)P(Y_1)}{P(X)} \times C_1, \\ P(Y_2|X) &= \frac{P(X|Y_2)P(Y_2)}{P(X)} \times C_2, \end{aligned} \quad (5)$$

where C_1 and C_2 denote the cost of the majority class and minority class, respectively. Generally, the cost assignment relates with the class imbalance ratio (IR), i.e.,

$$\frac{C_1}{C_2} = \frac{P(Y_2)}{P(Y_1)}. \quad (6)$$

Then it means $C_1P(Y_1) = C_2P(Y_2)$, which also satisfies the condition in (4).

Finally, we wish to investigate the rationality of threshold moving technique. It often adds a compensation threshold λ for the decision value of minority class, i.e.,

$$P(Y_2|X) = \frac{P(X|Y_2)P(Y_2)}{P(X)} + \lambda, \quad (7)$$

where λ is usually a positive value. Specifically, when $\lambda = P(X|Y_2)(P(Y_1) - P(Y_2))/P(X)$, it exactly satisfies the condition in (4).

Therefore, the rationality of various CIL techniques can be explained well within the theoretical framework of GNB, and they all can be seen as different strategies to balance the prior probabilities, further making (4) be workable.

4. Methods

4.1. KNN-PDE-Alike Conditional Probability Density Estimation Strategy. Motivated by (4), we observe a new potential CIL solution, i.e., neglecting prior probability and directly estimating the conditional probability belonging to each class to make decision. The solution avoids the tedious procedure of balancing prior probabilities and solves CIL problem in nature. The problem seems to get easier; however, it is still difficult to provide an accurate estimation for conditional probability. At least for GNB, it cannot precisely measure the conditional probability, as in many real-world applications, the data do not satisfy the Gaussian distribution. Figure 2 presents two different groups of data with the same mean and variance, but different distributions. It is clear that the GNB is able to provide an accurate estimation for the first group of data, while on the second group of data, the estimation will be extremely biased.

In order to provide accurate and robust probability density estimation, we present a novel strategy named KNN-PDE-alike algorithm. As its name indicates, KNN-PDE-alike algorithm is quite similar to KNN-PDE algorithm [47], but it has better adaptability for various complex data distributions.

In a group of data, KNN-PDE-alike algorithm firstly searches the Kth nearest neighbor for each instance x_i and records the Euclidean distance between x_i and its Kth nearest neighbor as d_i^K . Obviously, a smaller d_i^K denotes a higher probability density corresponding to x_i , and vice versa. It is a counterintuitive value principle; hence, a transformation

rule should be designed. In this study, we adopt a simple transformation rule that takes the reciprocal of distance to express the probability density. That is to say, for a sample x_i , its probability density could be represented as $o_i = 1/d_i^K$. Now, the probability density can be still represented as a function of the distance variance, which is inconsistent with the definition of probability density. Therefore, the rough probability density o_i should be further refined by the following transformation:

$$P(x_i) = \frac{o_i}{\sum}, \quad (8)$$

where $P(x_i)$ denotes the refined probability density of x_i and $\sum$ can be calculated by

$$\sum = \sum_{i=1}^s o_i. \quad (9)$$

The transformation can be seen as a normalization procedure of generating probability density. In fact, $P(x_i)$ is still not real probability density; however, it reflects the real proportional relation among probability density of different examples. Therefore, we call this kind of probability density the relative probability density.

According to the procedure description above, we provide the pseudocode of KNN-PDE-alike algorithm in Algorithm 1.

4.2. Probability Density Machine (PDM) Algorithm. It is clear that we can accurately estimate the relative probability density for each sample by virtue of KNN-PDE-alike strategy. Also, the relative probability density can be regarded as a scaling transformation of the corresponding real conditional probability density. Therefore, according to (4), we can directly make decision for the class label of a sample by comparing its relative probability densities in different classes. We call this method as Probability Density Machine (PDM).

However, we note that, in class imbalance data, different classes always have different number of samples, which means the relative probability densities in different classes calculated by (8) could not be compared directly, as they are scaled in different scales. To make them comparable, we should conduct an extra normalization step. For instance, x_i , its normalization relative probability density in a majority class j can be calculated by

$$P'(x_i) = P(x_i) \times \frac{s_j}{s_{\min}}, \quad (10)$$

where $P(x_i)$ and $P'(x_i)$ respectively denote the original and normalized relative probability density of the instance x_i in j th class, while s_j and $s_{\min}$ respectively represent the number of instances in the j th class and the class which has the least instances.

In addition, we should focus on another important parameter which might influence the accuracy of relative probability density estimation, i.e., the neighborhood parameter K . It is obviously inappropriate to designate a

FIGURE 2: Two distributions with the same mean and variance: (a) a Gaussian distribution and (b) a manifold distribution.

(i) **Input:** A data set $\Psi = \{x_i | x_i \in \mathbb{R}_n; n, 1 \leq i \leq s\}$, the parameter K
Output: A $1 \times s$ vector $\mathbf{l}$ recording the relative probability density of all instances
Procedure:
(1) For $i = 1 : s$
(2) For $j = 1 : s$
(3) If $(i \neq j)$
(4) Calculating the Euclidean distance between x_i and x_j , and recording it as d_{ij} ;
(5) End if
(6) End for
(7) Sorting all distances for the instance x_i in ascending order, then finding the K th one and recording it as d_i^K ;
(8) Transforming d_i^K to be o_i by the reciprocal rule;
(9) End for
(10) Calculating the sum of all o_i , $\sum$ by (9)
(11) For $i = 1 : s$
(12) Calculating the relative probability density $P(x_i)$ by (8);
(13) Recording $P(x_i)$ into the i th position in the vector $\mathbf{l}$;
(14) End for
(15) Output the vector $\mathbf{l}$.

ALGORITHM 1: KNN-PDE-alike probability density estimation algorithm.

common K for all classes; hence, we design an adaptive assignment rule that is automatically associated with the number of instances included in a specific class. Furthermore, we also observe that it is not suitable for designating a too large or too small value for the parameter K , as a too large K tends to destroy the difference of different density regions, while a too small K would be sensitive to noises. Hence, in this study, we empirically designate $K = \sqrt{s}$ as the default setting. The influence of the parameter K on the quality of PDM algorithm will be further discussed in Section 5.

According to the procedure description above, we provide the pseudocode of Probability Density Machine (PDM) algorithm in Algorithm 2.

From the pseudocode of PDM algorithm, we observe that it is very similar with those lazy learning algorithm, e.g., K -nearest neighbors classifier, as it needs to conduct lots of operations in testing phase. Actually, there exists a significant difference between the PDM and those lazy learning algorithms; i.e., lazy learning does not need training, but the PDM needs to conduct a complex training procedure for acquiring the normalization factor $\sum$, the number of instances s , and the parameter K in each class, respectively. Therefore, we call PDM a semilazy learning algorithm.

Furthermore, to verify the effectiveness of PDM algorithm, we also compare it with GNB algorithm on two imbalanced artificial data sets, where A1 contains 1000 majority instances satisfying the Gaussian distribution of

Input: An imbalanced training set $\Psi = \{(x_i, y_i) \mid x_i \in \mathbb{R}_n^n, 1 \leq i \leq s, y_i \in \{Y_1, Y_2, \dots, Y_m\}\}$, a test instance x'

Output: An predictive class label y' for x'

Training Procedure:

- (1) For $i = 1 : m$
- (2) For $j = 1 : s$
- (3) If $(y_j \in Y_i)$
- (4) Putting x_j into Φ_i
- (5) End if
- (6) End for
- (7) Recording the number of instances in Φ_i as s_i , and calculating the corresponding K_i
- (8) Calling KNN-PDE-alike algorithm to acquire and record $\sum_i$
- (9) End for
- (10) Ranking all s_i to get $s_{\min}$
- (11) Calculating each parameter K_i based on s_i .

Testing Procedure:

- (1) For $i = 1 : m$
- (2) For $j = 1 : s_j$
- (3) Calculating the Euclidean distance between x' and x_j , and recording it as d_{ij}
- (4) End for
- (5) Sorting all distances in ascending order, then finding the K_i th and recording it as d^{K_i}
- (6) Transforming d^{K_i} to be o_i by the reciprocal rule;
- (7) Calling $\sum_i$ and d^{K_i} to calculate the relative probability density $P_i(x')$ for x' by (8);
- (8) If $(s_i > s_{\min})$
- (9) Calling s_i and $s_{\min}$ to adjust the relative probability density $P_i(x')$ by (10);
- (10) End if
- (11) End for
- (12) Ranking m relative probability densities in descending order;
- (13) Output the class label $y' \in \{Y_1, Y_2, \dots, Y_m\}$ which ranks first as the prediction for the test instance x' .

ALGORITHM 2: Probability Density Machine (PDM) algorithm.

mean=0.5 and variance=0.1 and 100 minority instances satisfying the Gaussian distribution of mean=0.7 and variance=0.1, while A2 contains 5000 majority instances and 100 minority instances which satisfy the same distribution as A1. That is to say, both data sets have the same distribution, but different class imbalance ratios. Figure 3 presents the instance distribution of these two data sets, and the decision regions produced by GNB and PDM, respectively.

In Figure 3, we observe that the GNB is very sensitive to the variance of class imbalance ratio (IR), and with the increase of IR, the decision region of minority class would be significantly shrunken. By contrast, the PDM seems to be robust as its decision boundary is approximately invariable at the background of varying IR. This fact verifies our deduction; i.e., the impact of imbalanced data distribution is only related to prior probability, but not the conditional probability, again.

4.3. Time-Complexity Analysis. Finally, we focus on the time complexity of the proposed PDM algorithm. As a semilazy learning algorithm, the conduction of the PDM algorithm can be divided into two subprocedures.

Suppose the data set includes s instances which can be divided into m classes, and each instance holds n attributes; it mainly costs $O(s)$ time to divide the data subsets belonging to different classes and to determine the parameter K in each class, $O(m^2 \log m)$ time to find $s_{\min}$, $O(s^2 n)$ time to calculate

the distance among all instances, $O(s^2 \log s)$ time to sort the distances, and $O(s)$ time to conduct reciprocal operation and to calculate the normalization factor $\sum$ in each class, in training subprocedure. Considering real-world applications, the number of training instances is generally much larger than the number of classes, thereby we can say that the number of training instances s mainly dominates the time complexity in training phase. On low-dimensional data, i.e., $\log s \gg n$, its time complexity is $O(s^2 \log s)$, while on high-dimensional and small sample data, i.e., $\log s \ll n$, the time complexity becomes $O(s^2 n)$.

During testing subprocedure, for a testing instance, it will consume $O(sn)$ time to calculate distances, $O(s \log s)$ time to sort distances, $O(m)$ time to conduct reciprocal operation and to calculate the relative probability density, and $O(m^2 \log m)$ time to provide prediction. In comparison with training subprocedure, the time complexity of testing subprocedure could be negligible.

Based on the analysis above, we have to admit that the proposed PDM algorithm needs to sacrifice much more running time than that in GNB. PDM may be accurate, effective, and robust, but not efficient.

5. Experiments and Discussion

5.1. Data Set Description. We collected 30 binary-class and 10 multiclass imbalanced data sets to verify the effectiveness and feasibility of the proposed PDM algorithm. The

FIGURE 3: Instance distributions and decision regions produced by GNB and PDM on two imbalanced artificial data sets, respectively. (a) Instance distribution of A1. (b) Instance distribution of A2. (c) Decision regions produced by GNB on A1. (d) Decision regions produced by GNB on A2. (e) Decision regions produced by PDM on A1. (f) Decision regions produced by PDM on A2.

collection includes 13 data sets acquired from UCI machine learning repository [48] and 27 data sets collected from Keel data repository [49]. These data sets have different number of instances, number of attributes, number of classes, and class imbalance ratios (IR). The detailed description about these data sets is provided in Table 1.

5.2. Experimental Settings. All experiments were run on a 2.60 GHz Intel (R) Core (TM) i7 6700HQ 8 cores CPU with 16 GB RAM using MATLAB 2013a running environment.

To show the effectiveness and superiority of the proposed PDM algorithm, we compared it with the following algorithms:

- (1) GNB: it adopts GNB classifier [45, 46] to directly train the data without considering the skewed data distribution.
- (2) CS-GNB: it first trains GNB classifier [45, 46] on original data and then empirically assigns decision costs for the posterior probability in each class (see (5) and (6)).
- (3) RUS-GNB: it firstly calls random undersampling (RUS) algorithm [20] to generate a balanced training set from original data and then trains GNB classifier [45, 46] on the new balanced data.
- (4) ROS-GNB: it firstly calls random oversampling (ROS) algorithm [20] to generate a balanced training set from original data and then trains GNB classifier [45, 46] on the new balanced data.
- (5) SMOTE-GNB: it firstly calls Synthetic Minority Oversampling Technique (SMOTE) algorithm [21] to generate a balanced training set from original data and then trains GNB classifier [45, 46] on the new balanced data. In SMOTE, it adopts a default parameter $K=5$.

As we know, when we evaluate the performance of a CIL classifier, the total classification accuracy is not an excellent performance evaluation metric any longer; hence, we adopted F-measure and G-mean as performance evaluation metrics, where F-measure tests the tradeoff between precision and recall, while G-mean tests the tradeoff among the accuracy of each class.

To impartially compare the performance of various algorithms, we adopted 10 times' randomly external 5-fold cross validation to calculate the average performance as final results.

5.3. Results and Discussion. Table 2 and Table 3 present the F-measure and G-mean results of six algorithms on 40 imbalanced data sets.

From the results in Tables 2 and Table 3, it is not difficult to draw some conclusions as below.

- (1) We observed an interesting result; i.e., although GNB fails to consider class imbalance issue in data distribution, it performs no worse than CS-GNB, RUS-GNB, ROS-GNB, and SMOTE-GNB algorithms on

most data sets. Specifically, GNB acquires the best F-measure and G-mean results on 7 and 5 data sets, respectively. We think this counterintuitive phenomenon is perhaps associated with two different reasons. The first and also the most important reason lies in that the instance distribution in lots of used data sets might be very similar with Gaussian distribution. In addition, on those data sets which GNB performs better, perhaps there exists a large margin among different classes, further causing a biased model that also could accurately classify most instances.

- (2) Class imbalance ratio seems to be an important influence factor on the performance of GNB, which is specifically significant in terms of G-mean metric. In particular, on abalone19 and poker8v6 data sets which hold very high class imbalance ratios, GNB only acquires 0.2402 and 0.0000 G-mean values. The results prove our theoretical deduction again.
- (3) Among those GNB variants, SMOTE-GNB shows obviously better performance than its three competitors. It is not difficult to analyze the reasons why RUS tends to lose some important information to describe the real data distribution and ROS decreases the accuracy of probability density estimation as only inserting some synthetic minority instances on those intrinsic positions, and although a cost-sensitive strategy avoids the impact of biased prior probability, it still could not provide a robust estimation for real conditional probability density. As a synthetic oversampling technique, SMOTE fills the space among intrinsic minority instances, further promoting the accuracy of conditional probability density estimation. This explains why SMOTE-GNB outperforms the other algorithms.
- (4) It is clear that our proposed PDM model is superior to the other five algorithms. Specifically, PDM has acquired the best results on 29 and 26 data sets in terms of F-measure and G-mean, respectively. Actually, PDM promotes the modeling quality from two different aspects: one is totally balancing the prior probabilities and the other is providing approximately accurate conditional probability density estimation for any types of data distribution. The results also further indicate the effectiveness and feasibility of the proposed PDM algorithm in this study.

5.4. Significance Analysis in Statistics. To present a thorough understanding of the comparison of various algorithms, we also provide their statistical results. We employ Friedman ranking test and Holm post-hoc test [50, 51] to differentiate the performance of the comparative methods on 40 data sets. The statistical analysis results obtained at the confidence level of $\alpha=0.05$ are presented in Table 4 and Table 5.

The statistical results in Table 4 and 5 show that the proposed PDM algorithm acquires the lowest average ranking values in both terms of F-measure and G-mean

TABLE 1: Data sets used in this paper.

Data set	No. of classes	No. of instances	Minority class	Majority class	IR	Source
Abalone9	2	4177	Class9	Remainder classes	5.06	UCI
Abalone19	2	4177	Class19	Remainder classes	129.53	UCI
Banknote	2	1372	Class1	Class0	1.25	UCI
Ctg15	2	2126	Class1, 5	Remainder classes	28.53	UCI
Ctg23	2	2126	Class2, 3	Remainder classes	11.08	UCI
ILPD	2	583	Nonliver patient	Liver patient	2.49	UCI
Mfeatmor0	2	2000	Class0	Remainder classes	9.00	UCI
Mfeatmor01	2	2000	Class0, 1	Remainder classes	4.00	UCI
Pageblocks2345	2	5473	Class2–5	Class1	8.77	UCI
Seeds2v13	2	210	Class2	Class1, 3	2.00	UCI
Segment1	2	2310	Class1	Class2–7	6.00	UCI
Segment12	2	2310	Class1, 2	Class3–7	2.50	UCI
Wilt	2	4839	Class1	Class2	17.54	UCI
Ecoli0v1	2	220	Class1	Class0	1.86	Keel
Ecoli1	2	336	Class1	Remainder classes	3.36	Keel
Tic-tac-toe	2	958	Negative	Positive	1.89	Keel
Glass0	2	214	Class0	Remainder classes	2.06	Keel
Glass1	2	214	Class1	Remainder classes	1.82	Keel
Haberman	2	306	Positive	Negative	2.78	Keel
Vowel0	2	988	Class0	Remainder classes	9.98	Keel
Poker8v6	2	1477	Class8	Class6	85.88	Keel
Vehicle0	2	846	Class0	Remainder classes	3.25	Keel
Vehicle2	2	846	Class2	Remainder classes	2.88	Keel
Yeast1	2	1484	Class1	Remainder classes	2.46	Keel
Heart	2	270	Negative	Positive	1.25	Keel
Appendicitis	2	106	Positive	Negative	4.05	Keel
Bands	2	365	Class1	Class0	1.70	Keel
Bupa	2	345	Class1	Class0	1.38	Keel
Sonar	2	208	Class0	Class1	1.14	Keel
Wdbc	2	569	Class1	Class0	1.68	Keel
Page-blocks	5	5472	—	—	—	Keel
Flare	6	1066	—	—	—	Keel
Movement_libras	15	360	—	—	—	Keel
Winequality-red	4	1599	—	—	—	Keel
Wine	3	178	—	—	—	Keel
Vowel	11	990	—	—	—	Keel
Letter	26	20000	—	—	—	Keel
Texture	11	5500	—	—	—	Keel
Satimage	6	6435	—	—	—	Keel
Newthyroid	3	215	—	—	—	Keel

Note: IR denotes the class imbalance ratio.

TABLE 2: F-measure of various algorithms on 40 imbalanced data sets.

Data set	GNB	CS-GNB	RUS-GNB	ROS-GNB	SMOTE-GNB	PDM
Abalone9	0.6786	0.5566	0.4955	0.5545	0.5614	0.6161
Abalone19	0.9297	0.7017	0.6686	0.7007	0.7125	0.7060
Banknote	0.8320	0.8411	0.8426	0.8407	0.8422	0.9956
Ctg15	0.7207	0.7884	0.6750	0.6972	0.7672	0.9577
Ctg23	0.9386	0.9290	0.9379	0.9339	0.9357	0.9473
ILPD	0.5541	0.4649	0.5338	0.5581	0.5636	0.6511
Mfeatmor0	0.9990	0.9985	0.9990	0.9990	0.9990	0.9990
Mfeatmor01	0.8802	0.8327	0.8019	0.8508	0.8544	0.9854
Pageblocks2345	0.8958	0.9084	0.9123	0.8966	0.9006	0.9269
Seeds2v13	0.9552	0.9507	0.9550	0.9505	0.9552	0.9738
Segment1	0.8535	0.8488	0.8042	0.8370	0.8459	0.9847
Segment12	0.7355	0.7736	0.6395	0.7094	0.7175	0.9917
Wilt	0.9045	0.6257	0.4223	0.6264	0.6702	0.9390
Ecoli0v1	0.9345	0.9724	0.9175	0.9463	0.9464	0.9796

TABLE 2: Continued.

Data set	GNB	CS-GNB	RUS-GNB	ROS-GNB	SMOTE-GNB	PDM
Ecoli1	0.7311	0.8747	0.6053	0.6857	0.7501	0.8999
Tic-tac-toe	0.6844	0.6475	0.6478	0.6447	0.6504	0.9958
Glass0	0.6527	0.6590	0.6467	0.6584	0.6600	0.7382
Glass1	0.6117	0.5916	0.5693	0.5999	0.5951	0.7366
Haberman	0.7021	0.6900	0.6119	0.7049	0.7164	0.5608
Vowel0	0.9409	0.9565	0.8883	0.8899	0.8938	0.9878
Poker8v6	0.9833	0.7216	0.3312	0.7505	0.8187	0.9926
Vehicle0	0.6531	0.6549	0.6518	0.6470	0.6533	0.9183
Vehicle2	0.8079	0.8155	0.7333	0.7772	0.7828	0.8883
Yeast1	0.1893	0.5656	0.1943	0.1793	0.2061	0.7462
Heart	0.8492	0.8456	0.8455	0.8437	0.8455	0.7584
Appendicitis	0.8207	0.7975	0.7707	0.7981	0.8145	0.7690
Bands	0.2335	0.5919	0.2306	0.2309	0.2526	0.6787
Bupa	0.5506	0.4932	0.4675	0.5040	0.4983	0.6165
Sonar	0.6840	0.6776	0.6606	0.6925	0.6794	0.7883
Wdbc	0.9297	0.9260	0.9435	0.9271	0.9323	0.8845
Page-blocks	0.9161	0.8503	0.7871	0.8650	0.8770	0.8823
Flare	0.6459	0.6953	0.5674	0.6403	0.6455	0.4152
Movement_libras	0.6035	0.6066	0.5922	0.6129	0.6180	0.7302
Winequality-red	0.5574	0.5348	0.3317	0.4817	0.3854	0.4822
Wine	0.9807	0.9806	0.9834	0.9752	0.9834	0.9778
Vowel	0.6800	0.6704	0.6861	0.6753	0.6809	0.8604
Letter	0.6400	0.6345	0.6314	0.6392	0.6397	0.8801
Texture	0.7723	0.7565	0.7706	0.7722	0.7725	0.9660
Satimage	0.8058	0.8023	0.7780	0.8039	0.8031	0.8655
Newthyroid	0.9585	0.9584	0.9438	0.9608	0.9608	0.7676

*In each row, the best result has been highlighted in boldface.

TABLE 3: G-mean of various algorithms on 40 imbalanced data sets.

Data set	GNB	CS-GNB	RUS-GNB	ROS-GNB	SMOTE-GNB	PDM
Abalone9	0.6024	0.5966	0.5530	0.5933	0.5982	0.6397
Abalone19	0.2402	0.6058	0.6205	0.6051	0.6145	0.6432
Banknote	0.8283	0.8396	0.8412	0.8390	0.8408	0.9960
Ctg15	0.7565	0.8181	0.7189	0.7372	0.7861	0.6456
Ctg23	0.8008	0.8537	0.8797	0.8398	0.8474	0.9200
ILPD	0.6083	0.5239	0.5904	0.6115	0.6172	0.4377
Mfeatmor0	0.9945	0.9927	0.9945	0.9945	0.9945	0.9945
Mfeatmor01	0.8613	0.8478	0.8324	0.8701	0.8685	0.9713
Pageblocks2345	0.6634	0.6995	0.6156	0.6999	0.7295	0.6681
Seeds2v13	0.9579	0.9555	0.9555	0.9544	0.9581	0.9715
Segment1	0.8929	0.8881	0.8528	0.8796	0.8869	0.9460
Segment12	0.7766	0.7421	0.6911	0.7543	0.7605	0.9853
Wilt	0.7148	0.6663	0.5280	0.6655	0.6945	0.5542
Ecoli0v1	0.9057	0.9613	0.8798	0.9229	0.9232	0.9785
Ecoli1	0.7744	0.8909	0.6653	0.7384	0.7920	0.8804
Tic-tac-toe	0.5891	0.6352	0.6358	0.6300	0.6395	0.9937
Glass0	0.6844	0.6892	0.6804	0.6942	0.6924	0.5874
Glass1	0.6303	0.5980	0.5864	0.6206	0.6120	0.6243
Haberman	0.4090	0.3665	0.5063	0.4720	0.5126	0.5981
Vowel0	0.9263	0.8345	0.9008	0.9038	0.8983	0.9471
Poker8v6	0.0000	0.4711	0.2363	0.3418	0.4426	0.5300
Vehicle0	0.6894	0.6761	0.7052	0.6949	0.7024	0.9166
Vehicle2	0.7164	0.6679	0.7365	0.7472	0.7695	0.7762
Yeast1	0.2059	0.6205	0.2129	0.1855	0.2325	0.7160
Heart	0.8431	0.8399	0.8382	0.8383	0.8388	0.7567
Appendicitis	0.7653	0.7499	0.7486	0.7603	0.7721	0.7419
Bands	0.1474	0.5944	0.1321	0.1427	0.2015	0.6872
Bupa	0.5548	0.4838	0.4525	0.4989	0.4910	0.5506

TABLE 3: Continued.

Data set	GNB	CS-GNB	RUS-GNB	ROS-GNB	SMOTE-GNB	PDM
Sonar	0.6837	0.6759	0.6593	0.6903	0.6792	0.7862
Wdbc	0.9227	0.9170	0.9334	0.9203	0.9248	0.8994
Page-blocks	0.8067	0.7378	0.7664	0.8186	0.8359	0.7897
Flare	0.6563	0.7274	0.5846	0.6496	0.6604	0.5625
Movement_libras	0.7322	0.7312	0.7254	0.7392	0.7407	0.8236
Winequality-red	0.4553	0.0925	0.6074	0.6540	0.6417	0.6642
Wine	0.9869	0.9865	0.9886	0.9828	0.9888	0.9840
Vowel	0.8057	0.7986	0.8097	0.8031	0.8055	0.9161
Letter	0.7873	0.7831	0.7813	0.7867	0.7871	0.9334
Texture	0.8631	0.8526	0.8619	0.8630	0.8632	0.9809
Satimage	0.8680	0.8665	0.8525	0.8677	0.8677	0.9155
Newthyroid	0.9371	0.9350	0.9344	0.9428	0.9428	0.8821

Note: in each row, the best result has been highlighted in boldface.

TABLE 4: Statistical analysis results of comparative algorithms in terms of F-measure at the confidence level of $\alpha = 0.05$.

Algorithm	Average ranking	p	Holm	Hypothesis
PDM	2.05	—	—	—
GNB	2.96	0.0365	0.0500	Rejected
CS-GNB	3.85	0.0000	0.0167	Rejected
RUS-GNB	4.95	0.0000	0.0100	Rejected
ROS-GNB	4.26	0.0000	0.0125	Rejected
SMOTE-GNB	2.93	0.0291	0.0250	Not rejected

TABLE 5: Statistical analysis results of comparative algorithms in terms of G-mean at the confidence level of $\alpha = 0.05$.

Algorithm	Average ranking	p	Holm	Hypothesis
PDM	2.48	—	—	—
GNB	3.48	0.0168	0.0250	Rejected
CS-GNB	4.13	0.0001	0.0125	Rejected
RUS-GNB	4.53	0.0000	0.0100	Rejected
ROS-GNB	3.74	0.0025	0.0167	Rejected
SMOTE-GNB	2.66	0.6540	0.0500	Not rejected

FIGURE 4: The performance variation with the variation of the parameter (K), where, in X-axis, 1, 2, 3, 4, and 5 denote $\{\sqrt{s}/4, \sqrt{s}/2, \sqrt{s}, 2\sqrt{s}, 4\sqrt{s}\}$, respectively.

metrics, indicating that it is the best one among the comparative algorithms. For both metrics, the Holm post-hoc test rejects all comparative algorithms except SMOTE-GNB as their p values are smaller than the corresponding Holm values, which verifies that the PDM significantly outperforms these comparative algorithms. Of course, we cannot say there exists a significant difference between the PDM and the SMOTE-GNB, though the former has a lower average ranking than the latter. All in all, the proposed PDM algorithm presents more superior performance and lower average ranking than all other competitors and thereby we can safely recommend it as an effective solution to address CIL problem.

5.5. Discussion about the Impact of Parameter K . Next, we wish to discuss the sensibility of parameter K . As we know, K is also sole parameter in the PDM algorithm. In Section 4, we design it as a function of the number of instances s . To make clear whether the modeling quality of PDM is sensitive to K or not, we vary the value of K in the range of $\{\sqrt{s}/4, \sqrt{s}/2, \sqrt{s}, 2\sqrt{s}, 4\sqrt{s}\}$ and then observe the variation of performance of PDM. The results on three different data sets are presented in Figure 4.

In Figure 4, we observe that the classification performance of PDM is extremely sensitive to the parameter K , as with the variation of K , the classification performance changes drastically. Therefore, in practical applications, this parameter should be carefully selected and designated.

From the results in Figure 4, we also find that although for different performance metrics each data set has different best K values, all curves still show a common varying trend; i.e., with the increase of K , the performance firstly improves until a peak value, and then it will decline sharply. The observation is in accordance with our analysis in Section 4. That is to say, either too small or too large K value goes against the accurate estimation for conditional probability density. A too small K value would be sensitive to noises, while a too large K value would tend to destroy the difference of different density regions. According to the results presented in Figure 4, we suggest to designate K as a value lying between $\sqrt{s}/2$ and $2\sqrt{s}$. In real-world applications, the best K value can be determined by an internal cross-validation, too.

6. Concluding Remarks

In this study, from a theoretical perspective, we tried to analyze the reason why class imbalance distribution hurts the performance of predictive model in context of Gaussian Naive Bayes classifier. It is deduced that the hazard of imbalanced data distribution is only associated with prior probability, but not with conditional probability density. Accordingly, we analyzed the rationality of several popular class imbalance learning techniques and indicated the necessity of accurately estimating the conditional probability density in data distribution. Then, a robust probability density estimation algorithm named KNN-PDE-alike is proposed, and based on this algorithm, a novel class imbalance learning solution named PDM is further presented. Experimental results on 40 binary-class and multiclass

imbalanced data sets have shown the effectiveness and superiority of the proposed PDM algorithm.

The proposed PDM algorithm holds several merits as follows:

- (1) It has a good theoretical basis
- (2) It is insensitive to data distribution and is able to adapt various types of data distribution
- (3) It can be directly used to classify not only binary-class imbalance data but also multiclass imbalance data

In future work, the possibility of developing more robust and fast probability density estimation strategy will be investigated. The effectiveness and superiority of PDM will be further verified by applying it in more real-world class imbalance applications, too.

Data Availability

The data sets used in this paper were collected from the openly UCI machine learning repository and Keel data repository (<http://archive.ics.uci.edu/ml/datasets.php> and <https://sci2s.ugr.es/keel/datasets.php>). The MATLAB codes of PDM algorithm can be downloaded from <https://github.com/yuhualong1982/Probability-Density-Machine>.

Conflicts of Interest

The authors declare there are no known conflicts of interest.

Acknowledgments

This work was supported by the Natural Science Foundation of Jiangsu Province of China, under Grant no. BK20191457, the Open Project of Artificial Intelligence Key Laboratory of Sichuan Province, under Grant no. 2019RYJ02, and the National Natural Science Foundation of China, under Grant nos. 62076111 and 62076215.

References

- [1] C. Huang, X. Huang, Y. Fang et al., "Sample imbalance disease classification model based on association rule feature selection," *Pattern Recognition Letters*, vol. 133, pp. 280–286, 2020.
- [2] S. Fotouhi, S. Asadi, and M. W. Kattan, "A comprehensive data level analysis for cancer diagnosis on imbalanced data," *Journal of Biomedical Informatics*, vol. 90, Article ID 103089, 2019.
- [3] R. Malhotra and S. Kamal, "An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data," *Neurocomputing*, vol. 343, pp. 120–140, 2019.
- [4] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "MAHAKIL: diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction," *IEEE Transactions on Software Engineering*, vol. 44, no. 6, pp. 534–550, 2018.
- [5] Y. Qian, S. Ye, Y. Zhang, and J. Zhang, "SUMO-Forest: a cascade forest based method for the prediction of SUMOylation sites on imbalanced data," *Gene*, vol. 741, Article ID 144536, 2020.

- [6] H.-J. Dai and C.-K. Wang, "Classifying adverse drug reactions from imbalanced twitter data," *International Journal of Medical Informatics*, vol. 129, pp. 122–132, 2019.
- [7] L. Wang, Y. Chen, H. Jiang, and J. Yao, "Imbalanced credit risk evaluation based on multiple sampling multiple kernel fuzzy self-organizing map and local accuracy ensemble," *Applied Soft Computing Journal*, vol. 91, Article ID 106262, 2020.
- [8] P. D. Mahajan, A. Maurya, A. Megahed, A. Elwany, R. Strong, and J. Blomberg, "Optimizing predictive precision in imbalanced datasets for actionable revenue change prediction," *European Journal of Operational Research*, vol. 285, no. 3, pp. 1095–1113, 2020.
- [9] D. Veganzones and E. Séverin, "An investigation of bankruptcy prediction in imbalanced datasets," *Decision Support Systems*, vol. 112, pp. 111–124, 2018.
- [10] W. Qian and S. Li, "A novel class imbalance-robust network for bearing fault diagnosis utilizing raw vibration signals," *Measurement*, vol. 156, Article ID 107567, 2020.
- [11] G. Chen and Z. Ge, "SVM-tree and SVM-forest algorithms for imbalanced fault classification in industrial processes," *IFAC Journal of Systems and Control*, vol. 8, Article ID 100052, 2019.
- [12] A. Shariffar, F. Sarmadian, and B. Minasny, "Mapping imbalanced soil classes using Markov chain random fields models treated with data resampling technique," *Computers and Electronics in Agriculture*, vol. 159, pp. 110–118, 2019.
- [13] A. Shariffar, F. Sarmadian, B. P. Malone, and B. Minasny, "Addressing the issue of digital mapping of soil classes with imbalanced class observations," *Geoderma*, vol. 350, pp. 84–92, 2019.
- [14] H. Jeong, Y. Jang, P. J. Bowman, and N. Masoud, "Classification of motor vehicle crash injury severity: a hybrid approach for imbalanced data," *Accident Analysis & Prevention*, vol. 120, pp. 250–261, 2018.
- [15] Y. S. Li, H. Chi, X. Y. Shao, M. L. Qi, and B. G. Xu, "A novel random forest approach for imbalance problem in crime linkage," *Knowledge-Based Systems*, vol. 195, Article ID 105738, 2020.
- [16] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.
- [17] H. Haibo He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
- [18] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Computing Surveys*, vol. 49, no. 2, pp. 1–50, Article ID 31, 2016.
- [19] H. Guo, Y. Li, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalance data: review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.
- [20] G. Kovacs, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," *Applied Soft Computing Journal*, vol. 83, Article ID 105662, 2019.
- [21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
- [22] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE--Majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, 2014.
- [23] L. Jiang and C. Qiu, "A novel minority cloning technique for cost-sensitive learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, Article ID 1551004, 2015.
- [24] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.
- [25] H. Yu, J. Ni, and J. Zhao, "ACOSampling: an ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data," *Neurocomputing*, vol. 101, pp. 309–318, 2013.
- [26] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera, "SMOTE-RSB *: a hybrid preprocessing approach based on over-sampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory," *Knowledge and Information Systems*, vol. 33, no. 2, pp. 245–265, 2011.
- [27] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTE-IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Information Sciences*, vol. 291, pp. 184–203, 2015.
- [28] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.
- [29] H. Yu, C. Sun, X. Yang, S. Zheng, and H. Zou, "Fuzzy support vector machine with relative density information for classifying imbalanced data," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 12, pp. 2353–2367, 2019.
- [30] L. Jiang, C. Li, and S. Wang, "Cost-sensitive Bayesian network classifiers," *Pattern Recognition Letters*, vol. 45, pp. 211–216, 2014.
- [31] D. Gupta and B. Richhariya, "Entropy based fuzzy least squares twin support vector machine for class imbalance learning," *Applied Intelligence*, vol. 48, no. 11, pp. 4212–4231, 2018.
- [32] D. Gupta, B. Richhariya, and P. Borah, "A fuzzy twin support vector machine based on information entropy for class imbalance learning," *Neural Computing & Applications*, vol. 31, no. 11, pp. 7153–7164, 2019.
- [33] P. Borah and D. Gupta, "Robust twin bounded support vector machines for outliers and imbalanced data," *Applied Intelligence*, vol. 51, pp. 1–30, 2021.
- [34] H. Yu, C. Sun, X. Yang, W. Yang, J. Shen, and Y. Qi, "ODOC-ELM: optimal decision outputs compensation-based extreme learning machine for classifying imbalanced data," *Knowledge-Based Systems*, vol. 92, pp. 55–70, 2016.
- [35] H. Yu, C. Mu, C. Sun, W. Yang, X. Yang, and X. Zuo, "Support vector machine-based optimized decision threshold adjustment strategy for classifying imbalanced data," *Knowledge-Based Systems*, vol. 76, pp. 67–78, 2015.
- [36] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: a hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, 2010.
- [37] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Proceedings of the International Conference on Computational Intelligence and Data Mining*, pp. 324–331, Nashville, TN, USA, April 2009.
- [38] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: misclassification cost-sensitive boosting," in *Proceedings of the International Conference of Machine Learning*, pp. 97–105, Bled, Slovenia, June 1999.

- [39] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.
- [40] H. Yu and J. Ni, "An improved ensemble learning method for classifying high-dimensional and imbalanced biomedicine data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 4, pp. 657–666, 2014.
- [41] P. Lim, C. K. Goh, and K. C. Tan, "Evolutionary cluster-based synthetic oversampling ensemble (ECO-Ensemble) for imbalance learning," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2850–2861, 2017.
- [42] H. G. Zefrehi and H. Altincay, "Imbalance learning using heterogeneous ensembles," *Expert Systems with Applications*, vol. 142, Article ID 113005, 2020.
- [43] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013.
- [44] H. Yu, J. Ni, S. Xu, B. Qin, and H. Jv, "Estimating harmfulness of class imbalance by scatter matrix based class separability measure," *Intelligent Data Analysis*, vol. 18, no. 2, pp. 203–216, 2014.
- [45] H. Zhang, H. Zhang, S. Pirbhulal, and W. Wu, "Active balancing mechanism for imbalanced medical data in deep learning-based classification models," *ACM Trans. Multimedia Computer Communication Application*, vol. 16, p. 39, 2020.
- [46] J. C. Griffis, J. B. Allendorfer, and J. P. Szaflarski, "Voxel-based Gaussian naïve Bayes classification of ischemic stroke lesions in individual T1-weighted MRI scans," *Journal of Neuroscience Methods*, vol. 257, pp. 97–108, 2016.
- [47] Q. Wang, S. R. Kulkarni, and S. Verdú, "Divergence estimation for multidimensional densities via k -Nearest-Neighbor distances," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2392–2405, 2009.
- [48] C. Blake, E. Keogh, and C. J. Merz, "UCI repository of machine learning databases," Technical Report 213, Department of Information and Computer Science, University of California, Irvine, CA, USA, 1998.
- [49] I. Triguero, S. González, J. M. Moyano et al., "Keel 3.0: an open source software for multi-stage analysis in data mining," *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 1238–1249, 2017.
- [50] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [51] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.