Hindawi
Scientific Programming
Volume 2020, Article ID 8852705, 18 pages
https://doi.org/10.1155/2020/8852705

Research Article

Software Defect Prediction Based on Fuzzy Weighted Extreme Learning Machine with Relative Density Information

Shang Zheng, Jinjing Gai, Hualong Yu, Haitao Zou, and Shang Gao

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

Correspondence should be addressed to Shang Gao; gao_shang1972@163.com

Received 29 March 2020; Revised 14 July 2020; Accepted 3 November 2020; Published 19 November 2020

Academic Editor: Daniela Briola

To identify software modules that are more likely to be defective, machine learning has been used to construct software defect prediction (SDP) models. However, several previous works have found that the imbalanced nature of software defective data can decrease the model performance. In this paper, we discussed the issue of how to improve imbalanced data distribution in the context of SDP, which can benefit software defect prediction with the aim of finding better methods. Firstly, a relative density was introduced to reflect the significance of each instance within its class, which is irrelevant to the scale of data distribution in feature space; hence, it can be more robust than the absolute distance information. Secondly, a *K*-nearest-neighbors-based probability density estimation (KNN-PDE) alike strategy was utilised to calculate the relative density of each training instance. Furthermore, the fuzzy memberships of sample were designed based on relative density in order to eliminate classification error coming from noise and outlier samples. Finally, two algorithms were proposed to train software defect prediction models based on the weighted extreme learning machine. This paper compared the proposed algorithms with traditional SDP methods on the benchmark data sets. It was proved that the proposed methods have much better overall performance in terms of the measures including *G*-mean, AUC, and Balance. The proposed algorithms are more robust and adaptive for SDP data distribution types and can more accurately estimate the significance of each instance and assign the identical total fuzzy coefficients for two different classes without considering the impact of data scale.

1. Introduction

SDP (software defect prediction) [1] has become an active research topic in software engineering, which has drawn growing interests from both academia and industry. It can be formulated as a learning problem, which is used to facilitate software testing and to save testing cost. Various machine learning methods have utilised software defect training data set to build prediction models, among which Random Forest [2] and Naive Bayes [3] were proved to have relatively stable performance. However, class imbalance [4–7] is a common problem in SDP data set, which can affect the model performance. Software defects distribution in software modules roughly conforms to Pareto principle, also known as the 80–20 rule. It means that 80% of the defects are concentrated in 20% of the program modules and the numbers of non-defective modules are much larger than the numbers of defective program modules. Hence, the accuracy of predicting few classes is lower.

Previous studies [8, 9] have indicated that the model tends to fail when it is applied to data with class imbalance problem. In order to address this problem, some imbalanced techniques as ROS (random oversampling) [10], RUS (random undersampling) [11], and SMOTE (synthetic minority oversampling technique) [12] have been considered to construct SDP model. In addition, Wang and Yao [13] analysed three different types of class imbalance methods for software defect prediction. They found their proposed ensemble approach DNC (Dynamic Adaboost.NC) is better than the ROS, RUS, and SMOTE. The DNC adjusts the parameter automatically during the training process, which can improve the prediction model's performance further. However, the above prediction models may encounter the following unpredictable problems: (1) choosing suitable

coefficients for different classes, (2) abandoning the instances in the small disjunctions, and (3) estimating the wrong class boundary, further resulting in the unexpected SDP classification results.

In this paper, we present a more robust representation measure of data distribution information called relative density, which can be extracted by a K-nearest-neighbors-based probability density estimation (KNN-PDE) [14–16] alike strategy, to evaluate the significance of each training instance and to design the corresponding fuzzy membership function. In contrast to Euclidean-distance-based measure, the relative density is irrelevant to the scale of data distribution in feature space. Meanwhile, it can also reflect the proportional relation of different instances within the class. Moreover, the KNN-PDE alike strategy has another merit that there is no need to normalize the fuzzy coefficients after acquiring the relative density information of all training instances. Then the fuzzy membership function is designed based on KNN-PDE and can assign the larger weights for those high-density instances. This paper recognises the fuzzy values as the weighted values of training instances and embeds them into weighted extreme learning machine (WELM), which can solve the noise or outliers effectively. WELM is selected as the baseline classifier based on three observations: (1) compared to other classifiers, WELM always has better or at least comparable generality ability and classification performance [17], (2) it can tremendously save training time compared to other classifiers [18], and (3) it can deal with data with imbalanced distribution based on cost-sensitive learning [19]. Finally, two algorithms based on WELM are proposed: one relies on the intraclass relative density and the other depends on the interclass relative density. That means the first function assigns the larger weights for those high-density instances, while the second function designates the larger weights for the examples which are nearer to the real classification boundary. To evaluate the algorithms' effectiveness, this paper performed a comparison with the previous works on the benchmark data sets, and the experimental results indicate that the proposed algorithms can generally produce better or at least comparable performance in terms of the measures including G-mean, AUC, and Balance.

The remainder of this paper is structured as follows. Section 2 introduces some a priori knowledge related to this work including software defect prediction, extreme learning, and weighted extreme learning machine. Section 3 describes the proposed methods in detail. The experiments and analysis are given in Section 4, and Section 5 concludes the research and provides suggestion for future work.

2. Related Work

In this section, some preliminaries are presented, including software defect prediction, extreme learning machine, and weighted extreme learning machine.

2.1. Software Defect Prediction. SDP models are expected to improve software quality and reduce maintenance cost of software systems. The researchers utilised the defect prediction data sets to build comparable models for studies. So far, a great number of researches have been devoted to metrics describing code modules and learning algorithms to create SDP models. A variety of machine learning methods have been proposed and compared for SDP problems, such as neural networks [20], decision trees [21], Naive Bayes [22], and support vector machine [23]. However, the above methods ignore the effect of class imbalance on the model performance [1]; that is, the numbers of defective instances are more or less than nondefective instances. It is a great challenge for most conventional classification algorithms to work with data that have an unbalanced class distribution, because they may ignore the minority class that could be more valuable in a wide range of applications. Thus, some class imbalance learning techniques have been utilised to reduce the negative effect. The work in [24] studied which type of metrics is useful to handle class imbalance based on static code. An undersampled approach was proposed to balance training data [25] and check how little information is required to learn a defect predictor. The authors found that throwing away data does not affect the performance of selected predictors. In addition, ensemble algorithms [26] and their cost-sensitive variants were studied and shown to be effective if a proper cost ratio can be set. Issam et al. [27] implemented software defect prediction using ensemble learning on selected features-greedy forward selection. Yang et al. [9] proposed an ensemble learning approach for just-in-time defect prediction, which contains two layers to improve the performance of SDP.

Besides the above introduced works, there are other existing works about software defect prediction which will not be listed because some of them do not consider data distribution, and several works just chose the basic sampling methods but different learning methods. In Section 1, it has been proved that WELM has three advantages over the traditional learning methods. Therefore, this paper will only study how to build more robust SDP models based on data distribution and do a comparison with the sampling methods through extensive experiments and comprehensive analyses.

2.2. Extreme Learning Machine. Extreme learning machine (ELM) that was proposed by Huang et al. [28] is a specific learning algorithm for single-hidden layer feedforward neural networks (SLFN). The main characteristic of ELM which distinguishes it from those conventional learning algorithms of SLFN is the random generation of hidden nodes. Therefore, ELM does not need to iteratively adjust parameters to make them approach the optimal values; thus it has faster learning speed and better generalization ability. Previous research has indicated that ELM can produce better or at least comparable generality ability and classification performance to SVM and multiple-level perceptron (MLP) but only consumes tenths or hundredths of training time compared to SVM and MLP.

Let us consider a classification problem with N training instances to distinguish m categories, and then the ith training instance can be represented as (x_i, t_i), where x_i is an $n \times 1$ input vector and t_i is the corresponding $m \times 1$ output

vector. Suppose that there are L hidden nodes in ELM and that all weights and biases on these nodes are generated randomly. Then, for the instance x_i, its hidden layer output can be represented as a row vector $h(x^i) = [h_1(x_i), h_2(x_i), \ldots, h_L(x_i)]$. The mathematical model of ELM could be described as

$$H\beta = T, \tag{1}$$

where $H = [h(x_1), h(x_2), \ldots, h(x_N)]^T$ is the hidden layer output matrix over all training instances; β is the weight matrix of the output layer; in equation (1), only β is unknown, so the least square algorithm is applied to acquire its solution, which can be described as follows:

$$\beta = H^{\dagger}T = \begin{cases} H^T\left(\mathrm{HH}^T\right)^{-1}T, & \text{when } N \le L, \\ \left(\mathrm{HH}^T\right)^{-1}H^TT, & \text{when } N > L, \end{cases} \tag{2}$$

where $H^{\dagger}$ denotes the Moore–Penrose generalized inverse of the hidden layer output matrix H, which can guarantee that the solution is the least-norm least-squares solution for equation (1).

According to previous work, ELM can be trained in the viewpoint of optimization. In the optimization version of ELM, we wish to synchronously minimize $\|H\beta - T\|^2$ and $\|\beta\|^2$, so the question can be described as follows:

$$\begin{aligned} &\text{minimize } Lp_{\mathrm{ELM}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\left\|\xi_i\right\|^2 \\ &\text{subject to } \quad h(x_i)\beta = t_i^T - \xi_i^T, \quad i = 1, 2, \ldots, N, \end{aligned} \tag{3}$$

where $\xi_i = [\xi_{i,1}, \xi_{i,2}, \ldots, \xi_{i,m}]$ denotes the training error vector of the m output nodes with respect to the training instance x_i and C is the penalty factor, representing the tradeoff between the minimization of training errors and maximization of generality ability. Obviously, this is a typical quadratic programming problem that can be solved by the Karush–Kuhn–Tucker (KKT) theorem [29]. The solution for equation (3) can be described as follows:

$$\beta = H^{\dagger}T = \begin{cases} H^T\left(\frac{I}{C} + \mathrm{HH}^T\right)^{-1}T, & \text{when } N \le L, \\ \left(\frac{I}{C} + \mathrm{HH}^T\right)^{-1}H^TT, & \text{when } N > L. \end{cases} \tag{4}$$

2.3. Weighted Extreme Learning Machine. Weighted extreme learning machine (WELM) that can be regarded as a cost-sensitive learning version of ELM is an effective way to handle imbalanced data [19]. Similar to CS-SVM, the main idea of WELM is to assign different penalties for different categories, where the minority class has a larger penalty factor C, while the majority class has a smaller C value. Then, WELM focuses on the training errors of the minority instances, making a classification hyperplane emerge in a more impartial position. A weighted matrix W is used to regulate the parameter C for different instances; that is, equation (3) can be rewritten as

$$\begin{aligned} &\text{minimize } Lp_{\mathrm{ELM}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}W\sum_{i=1}^{N}\left\|\xi_i\right\|^2 \\ &\text{subject to } \quad h(x_i)\beta = t_i^T - \xi_i^T, \quad i = 1, 2, \ldots, N, \end{aligned} \tag{5}$$

where W is an $N \times N$ diagonal matrix in which each value existing on the diagonal represents the corresponding regulation weight of parameter. Zong et al. [19] provided two different weighting strategies, which are described as follows:

$$\begin{aligned} &\text{WELM1: } W_{ii} = \frac{1}{\#(t_i)}, \\ &\text{WELM2: } W_{ii} = \begin{cases} \frac{0.618}{\#(t_i)}, & \text{if } \#(t_i) > \mathrm{AVG}(t_i), \\ \frac{1}{\#(t_i)}, & \text{if } \#(t_i) \le \mathrm{AVG}(t_i), \end{cases} \end{aligned} \tag{6}$$

where W_{ii}, $\#(t_i)$, $\mathrm{AUG}(t_i)$, and 0.618 denote the weight of the ith training instance, the number of instances belonging to the class t_i, the average number of instances over all classes, and the value of the golden standard, respectively. Compared with WELM2, WELM1 is more practical and popular. Then, the solution can be shown as follows:

$$\beta = \begin{cases} H^T\left(\frac{I}{C} + \mathrm{WHH}^T\right)^{-1}\mathrm{WT}, & \text{when } N \le L, \\ \left(\frac{I}{C} + \mathrm{HWH}^T\right)^{-1}H^T\mathrm{WT}, & \text{when } N > L. \end{cases} \tag{7}$$

Obviously, no matter which weight distribution method is used, few types of samples will be given more weight. Hence, the class imbalance ratio is the higher, and the weight ratio between different types of samples becomes higher. According to the work in [19], users can define W_{ii} for every sample x_i to improve the performance, so the paper considers constructing new W_{ii} based on the data distribution.

3. The Proposed Methods

Although WELM can improve class imbalance problem, it does not consider the distribution of samples in feature space. In addition, there are noise and outliers in the software defect data, which can further affect the performance. Thus, this paper draws on the experience of the works in [30, 31] and introduces the concept of fuzzy sets, which can mine the distribution of each instance in feature space and conduct the more personalized setting for the weights. In order to describe our method, this section first introduces the relative density that is applied to avoid the large calculation of probability density in high-dimensional space. Then the fuzzy membership functions are designed to replace the WELM's weight matrix based on relative density, and finally the two proposed algorithms of SDP are

described. Finally, the experiments are designed to validate the methods. The whole framework can be seen in Figure 1.

3.1. Relative Density Estimation Strategy. As is known, it will be easy to identify outliers and noise from the significant instances if we can estimate the probability of each training instance. However, on high-dimensional feature space, it is always difficult to acquire the exact measurements of the probability density. It would be time-consuming even if an approximately accurate estimation of the probability density is obtained. In order to solve this problem, we introduce an improved method in this subsection. We consider that it is unnecessary to measure the probability density exactly, but it is enough to precisely extract the proportional relation of the probability densities between any two training instances. We call the information reflecting the proportional relation as relative density.

To obtain relative density, a similar K-nearest-neighbors-based probability density estimation (KNN-PDE) is applied. As a nonparametric probability density estimation approach, KNN-PDE estimates the probability density distribution in multidimensional continuous space by measuring the K-nearest-neighbor distance of each training instance. When the number of the training instances goes to infinity, the result obtained from KNN-PDE can approximately converge to the real probability density distribution. Hence, the K-nearest-neighbor distances can be used to estimate the relative density, and Euclidian distance is selected to calculate the distance in the proposed algorithms.

Suppose that a data set contains N instances; then, for each instance x_i, we can find its Kth-nearest neighbors and record the distance between them as d_i^K. As is known, the larger d_i^K is, the lower density the instance x_i will hold. At the same time, no matter noise or outliers should appear in the region of low density, thereby we can use d_i^K as the measure to evaluate the significance of each instance. However, to provide larger value for high-density instances and lower value for low-density instances, for example, noise and outliers, d_i^K should be transformed to its reciprocal $1/d_i^K$. In this paper, the reciprocal of K-nearest-neighbors distance is defined as the instance's relative density. It is not difficult to observe that the proportional relation of the relative density between any two instances exactly equals the inverse of that of the K-nearest-neighbors distance between them, as

$$\frac{1/d_i^K}{1/d_j^K} = \frac{d_j^K}{d_i^K}. \tag{8}$$

Also, it is important to confirm the selection of the parameter K for the relative density. If the value of K is too small, it would be failed to identify the noise and outliers from those normal instances, but the distinction between those significant instances and the noise or outliers might become ambiguous and some small disjunctions would not be captured if the value of K is too large. To avoid this problem, this paper considers assigning an appropriate value for K. It is empirically set to be $\sqrt{N}$ during the experiments, where N denotes the number of the training instances.

3.2. Design of Fuzzy Membership Functions. Based on the relative density, two different fuzzy membership functions are designed. One adopts intraclass relative density information, and the other uses interclass relative density information. The details will be introduced in the following sections.

3.2.1. Fuzzy Membership Function Based on Intraclass Density Information. In this type of fuzzy membership function, $f(x_i)$ is defined with respect to $1/d_i^K$, which is the reciprocal of the distance between the instance x_i and its Kth-nearest neighbors within the same class. The instances appearing in the high-density region are seen as more information ones and they are assigned higher $f(x_i)$ values, while the examples far from the high-density region are seen as the noise or outliers and assigned them lower $f(x_i)$ values. To avoid the impact induced by data distribution scale, a normalized fuzzy membership function can be represented as follows:

$$f(x_i) = \frac{1/d_i^K}{\sum_{j=1}^{N_c}\left(1/d_j^K\right)}, \tag{9}$$

where N_c denotes the number of instances belonging to the class which x_i drops in. The merit lies in the fact that the fuzzy membership value only reflects the relative density within its own class but is irrespective of the number of instances in that class. Therefore, it will be more robust to the variance of the data distribution scale. In addition, due to the fact that each class is handled independently, it is adaptive for class imbalance problem.

3.2.2. Fuzzy Membership Function Based on Interclass Relative Density Information. In this type of fuzzy membership function, $f(x_i)$ is associated with the estimated class boundary; that is, the instance closer to the estimated class boundary will be assigned a higher membership value. To precisely estimate the class boundary, we deeply investigate the characteristics of four kinds of instances with respect to different density distributions. The instances are divided into noindent normal, boundary, noise, and outliers, respectively. Figure 2 provides a visualized description for these instances. The characteristics of these four instances could be concluded as follows:

(a) Normal: the instance appears in the high-density region within its own class but low-density region in the other class

(b) Boundary: the instance appears in the low- or medium-density region in both classes but always has a little higher density within its own class than that of the other

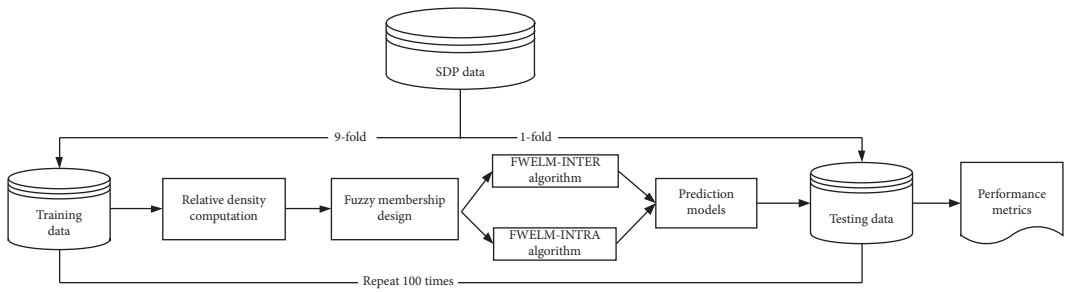

Figure 1: The framework of proposed methods.

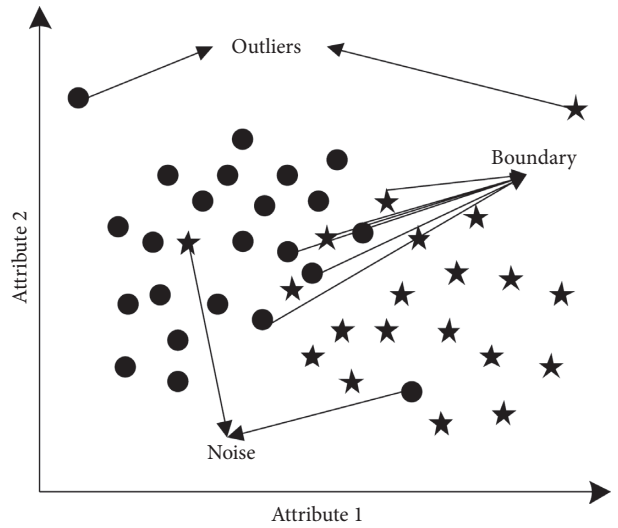

Figure 2: The example of the division for four kinds of instances distributed in feature space, where boundary, noise, and outliers have been indicated in the figure, and the remaining instances can be seen as normal ones.

(c) Noise: the instance appears in the low-density region within its own class but higher-density region in the other class

(d) Outliers: the instances appear in the low-density region in both classes

According to the characteristics listed above, we can exactly locate the boundary. First, for each instance, we can compare its intraclass relative density with interclass relative density to find the noise, which can be detected with a discriminant. If the instance x_i is from the positive class, its discriminant is shown as follows:

$$\frac{\left(1/d_i^{\lceil\sqrt{N^+}\rceil}\right)}{\sum_{j=1}^{N^+}\left(1/d_j^{\lceil\sqrt{N^+}\rceil}\right)} < \frac{\left(1/d_i'^{\lceil\sqrt{N^-}\rceil}\right)}{\sum_{j=1}^{N^-}\left(1/d_j^{\lceil\sqrt{N^-}\rceil}\right)} \times \mathrm{IR}, \tag{10}$$

where d' denotes the distance calculated only with the distance in the other class, N^+ and N^- denote, respectively, the numbers of instances in positive class and negative class, respectively, $\lceil\bullet\rceil$ provides the round-up operation, and IR is the class imbalance ratio that equals N^+/N^-. Meanwhile, if x_i comes from the negative class, then the discriminant is modified as

$$\frac{\left(1/d_i'^{\lceil\sqrt{N^+}\rceil}\right)}{\sum_{j=1}^{N^+}\left(1/d_j^{\lceil\sqrt{N^+}\rceil}\right)} > \frac{\left(1/d_i^{\lceil\sqrt{N^-}\rceil}\right)}{\sum_{j=1}^{N^-}\left(1/d_j^{\lceil\sqrt{N^-}\rceil}\right)} \times \mathrm{IR}, \tag{11}$$

for each instance satisfying the discriminant condition in equations (10) and (11), this paper can extract them as noise and then assign a very small member value λ for them.

Then, for the rest of instances, we assign their membership values with interclass relative density information. The fuzzy membership function can be represented as the following piecewise function:

$$f(x_i) = \begin{cases} \dfrac{\left(1/d_i'^{K}\right)}{\sum_{j=1}^{N_{c1}}\left(1/\left(d_j'^{K} + N_{c2} \times \lambda\right)\right)}, & \text{if } x_i \text{ is not the noise,} \\ \lambda, & \text{if } x_i \text{ is the noise,} \end{cases} \tag{12}$$

where N_{c1} and N_{c2} denote the numbers of instances belonging to nonnoise and noise with the same class of x_i, and $N_{c1} + N_{c2} = N_c$.

3.3. Two Proposed Algorithms. In this section, two proposed algorithms based on WELM are described. In order to set the personalized weights, this paper firstly considers the distribution information and obtained W_{ii}' as the new weight of each training sample based on the value of $f(x)$, which replaces the original W_{ii}. Then the new diagonal matrix W_{ii}' can be described as follows:

$$W_{ii}' = \begin{bmatrix} f(x_1) & 0 & 0 & 0 \\ 0 & f(x_2) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & f(x_N) \end{bmatrix}. \quad (13)$$

Next, this section describes the algorithm based on the intraclass relative density information called FWELM-INTRA and the algorithm based on the interclass relative density information called FWELM-INTER. Their flow paths are briefly described in Algorithms 1 and 2.

4. Experiments and Analysis

4.1. Data Sets. During this study, the experimental data sets are available from the public PROMISE repository [32], which have been commonly used in empirical studies of SDP. Detailed information about the data sets is shown in Table 1; each data set contains the number of instances, the number of defects, the number of metrics, and the percentage of defective modules. According to the defective modules ratio, each data set is imbalanced. In order to ensure the accuracy and convergence of the proposed solutions, the zero padding is used to solve the missing values in the date set and data normalization [33] is adopted before conducting the experiments.

4.2. Experimental Settings. Firstly, to validate the effectiveness and superiority of the two proposed algorithms, this paper compared them not only with many representative class imbalance learning algorithms based on ELM but also with WELM1 and WELM2. In addition, we also compared them with the ensemble method of DNC [13] that has been proved to be better than the traditional classifiers, that is, Naive Bayes and Random Forest. The simple description can be seen as follows:

(1) ELM [28]: it is the standard ELM algorithm without any operations to address class imbalance problem of SDP data sets.

(2) RUS [34]: it first adopts RUS algorithm to generate a totally balanced training set and then trains an ELM model on this training set.

(3) ROS [35]: it first adopts ROS algorithm to generate a totally balanced training set and then trains an ELM model on this training set.

(4) SMOTE [12]: it first adopts SMOTE algorithm to generate a totally balanced training set and then trains an ELM model on this training set.

(5) WELM1 and WELM2 [19]: two different weighted strategies of WELM have been adopted as the balance control for a binary-classification task. In particular, they can be regarded as a baseline algorithm that is used to indicate the effect of noise or outliers of SDP data sets.

(6) DNC [13]: it is the ensemble learning method to solve imbalance problem of SDP data sets. It can be regarded as a baseline algorithm that is used for comparison with the ensemble learning on performance.

Secondly, to measure the performance on the SDP data set, the probability of detection (PD) and the probability of false alarm (PF) are used based on [13]. For more comprehensive evaluation of predictors in the imbalanced context, *G*-mean [36] and AUC [37] are frequently used to measure how well the predictor can balance the performance between two classes. In the SDP context, *G*-mean reflects the change in PD efficiently [38]. It can be calculated by

$$G - \text{mean} = \sqrt{\text{PD}(1 - \text{PF})}. \quad (14)$$

AUC estimates the area under the ROC curve, formed by a set of (PF, PD) pairs. The ROC curve illustrates the tradeoff between detection and false alarm rates, which serves as the performance of a classifier across all possible decision thresholds. AUC provides a single number for performance comparison, varying in [0, 1]. A better classifier should produce a higher AUC. AUC is equivalent to the probability that a randomly chosen example of the positive class will have a smaller estimated probability of belonging to the negative class than a randomly chosen example of the negative class.

In the work in [13], the point (PF = 0, PD = 1) was proposed as the ideal position on the ROC curve, where all defects are recognized without mistakes; the measure Balance is introduced by calculating the Euclidean distance from the real (PF, PD) point to (0, 1) and is frequently used by software engineers in practice [39]. By definition,

$$\text{balance} = 1 - \frac{\sqrt{(0 - \text{PF})^2 + (1 - \text{PD})^2}}{\sqrt{2}}. \quad (15)$$

In summary, this paper uses *G*-mean, AUC, and Balance to guarantee that the experiments are effective. All of them are expected to be high for a good predictor. The advantage of the three measures is their insensitivity to class distributions in data [40].

Thirdly, to avoid the randomness of the experiments, this paper applied 10-fold cross validation at each time of building modes using nine of ten partitions and testing on

Input:
Software defect training set $\Theta = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where $y_i \in \{+, -\}$;
Penalty factor C, Hidden layer neurons L.
Output:
FWELM-INTRA Classifier.
(1) Divide Θ into two sets, Θ^+ only contains positive instances, and Θ^- only contains negative instances. Here, SDP aims to discover the defective modules, so this paper uses positive instances to represent software defective instance, and negative instances to represent non-defective instances;
(2) Count the number of instances in Θ^+ and Θ^-, then record them as N^+ and N^-, where $N^+ + N^- = N$;
(3) Calculate the parameter K for positive and negative class, where $K^+ = \lceil \sqrt{N^+} \rceil$, $K^- = \lceil \sqrt{N^-} \rceil$;
(4) For each instance x_i^+ in Θ^+, calculate its instances to the K^+ th nearest neighbors in Θ^+ and record it as d_i^+, as well for each instances x_j^- in Θ^-, calculate its distance to the K^-th nearest neighbors in Θ^- and record it as d_j^-;
(5) For each instance x_i in Θ, calculate its relative density by equation (8), and then calculate its fuzzy membership value $f(x_i)$ by equation (9);
(6) Obtain W_{ii}' based on the value of $f(x_i)$, and train a FWELM-INTRA classifier by equation (5) with the given parameters C and L.

Algorithm 1: FWELM-INTRA.

Input:
Software defect training set $\Theta = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where $y_i \in \{+, -\}$;
Penalty factor C, Hidden layer neurons L.
Output:
FWELM-INTER Classifier.
(1) Divide Θ into two sets, Θ^+ only contains positive instances, and Θ^- only contains negative instances. Here, SDP aims to discover the defective modules, so this paper uses positive instances to represent software defective instance, and negative instances to represent non-defective instances;
(2) Count the number of instances in Θ^+ and Θ^-, then record them as N^+ and N^-, where $N^+ + N^- = N$;
(3) Calculate the class imbalance ratio as $\mathrm{IR} = N^-/N^+$;
(4) Calculate the parameter K for positive and negative class, where $K^+ = \lceil \sqrt{N^+} \rceil$, $K^- = \lceil \sqrt{N^-} \rceil$;
(5) For each instance x_i^+ in Θ^+, calculate its instances to the K^+ th nearest neighbors in Θ^+ and record it as d_i^+, as well for each instances x_j^- in Θ^-, calculate its distance to the K^- th nearest neighbors in Θ^- and record it as d_j^-;
(6) Calculate the relative density of each instance and find the noise in two different classes by equations (10) and (11);
(7) For each instance x_i in Θ, calculate its relative density by equation (8), and then calculate its fuzzy membership value $f(x_i)$ by equation (12);
(8) Obtain W_{ii}' based on the value of $f(x_i)$, and train a FWELM-INTER classifier by equation (5) with the given parameters C and L.

Algorithm 2: FWELM-INTER.

the remaining one. The above procedure is repeated 100 times (10-fold cross validation) to calculate the average result for each algorithm, and the results are provided in the form of mean ± standard deviation. The whole settings can be seen in Figure 3.

Meanwhile, for each algorithm related to ELM and WELM, a sigmoid function is used to calculate the hidden layer output matrix, and two main parameters L and C are determined by grid search, where $L \in \{10, 20, \ldots, 200\}$ and $C \in \{2^{-4}, 2^{-2}, \ldots, 2^{20}\}$.

4.3. Research Questions. In this section, we are interested in answering the following three research questions:

(i) RQ1: do the proposed algorithms perform better than the previous studies?
(ii) RQ2: how does the K value impact performance?
(iii) RQ3: how about time-complexity of the two proposed algorithms?

In RQ1, we evaluate the effectiveness of the two proposed algorithms and compare them with previous studied G-mean, AUC, and Balance. In RQ2, we investigate the K value impacting our algorithms based on the range. In RQ3, we examine the time-complexity of two algorithms when the number of training instances and the number of attributes change, respectively. In the following sections, we provide the results analysis of the aforementioned three research questions.

4.4. RQ1: Do the Proposed Algorithms Perform Better than the Previous Studies?

4.4.1. Motivation. Previous studies note that good SDP methods can support the developers to find the defects. In this RQ, we would like to investigate whether our proposed

Table 1: NASA data sets.

Data set	Number of instances	Number of defects	Number of metrics	Defective%
mc2	125	44	40	35.2
kc2	522	107	22	20.5
jm1	10885	2106	22	19.3
kc3	194	36	40	18.5
kc1	2109	326	22	15.4
pc3	1077	134	38	12.4
pc4	1458	178	38	12.2
mw1	264	27	38	10.2
cm1	498	49	22	9.8
pc1	1109	77	22	6.9

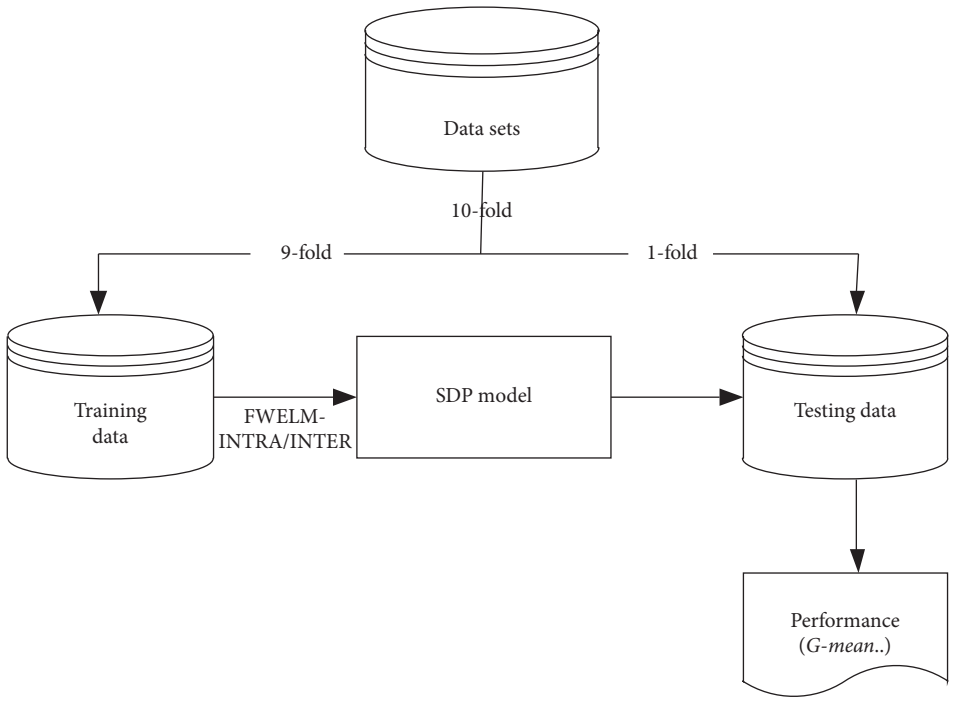

Figure 3: The whole experimental process.

algorithms can effectively perform better. The benefits of identifying software defects lie in two aspects: First, once a software defect is predicted, we can provide a timely warning to the development team and save the effort and time of the developers. Second, identifying the software defects can help to avoid defects in the future.

4.4.2. Approach. To answer this RQ, we implement our approach (the data and code for the algorithms are available at https://github.com/Dark204/Work) and compare the performance with the baselines based on the open-source projects. Then, we measure the performance and perform the statistical test.

4.4.3. Results. Tables 2–4 show the mean and standard deviation values of *G*-mean, AUC, and Balance. By observing the results, it is not difficult to draw some conclusions as follows:

(1) Class imbalance learning techniques are useful for promoting the prediction performance of SDP imbalance data, as, on all the selected SDP data sets, no matter the proposed fuzzy membership function with similar KNN-PDE information, sampling algorithms, WELM algorithms, or ensemble DNC algorithms have gained higher *G*-mean and Balance values than the standard ELM predictive model.

(2) Compared to DNC, the predictive models trained by FWELM-INTRA and FWELM-INTER have more higher *G*-mean values on all the data sets with the exception of pc4 and higher Balance values on all the data sets except kc1 and pc4. In addition, FWELM-INTRA and FWELM-INTER also have higher AUC values than DNC on six of ten data sets. Therefore, it is clear that the proposed algorithms show better performance than DNC. For some data sets that have lower performance than DNC, we analyse that the sizes of classes overlap and the absolute number of

training instances will affect the results. The results do not only imply that the two algorithms achieve a better balance between PD and PF but also prove the importance of the distribution of samples in space.

(3) The two proposed algorithms are based on WELM. So, they are also compared with WELM 1 and WELM 2. For *G*-mean, AUC, and Balance, two algorithms show outperforming efficiency, which proves that the fuzzy values based on relative information can improve WELM to train the better SDP models. Unlike traditional WELM, the proposed algorithms assign different weights that avoid the effect of noise or outliers.

(4) As illustrated in Section 2, ELM cannot deal with the class imbalance. But the combination of ELM and data sampling techniques, ELM-RUS, ELM-ROS, and ELM-SMOTE, is effective. Table 2 shows that FWELM-INTRA and FWELM-INTER are still better than the three improved ELM algorithms on *G*-mean metric. In Tables 3 and 4, the results of AUC and Balance of FWELM-INTRA and FWELM-INTER are also higher than those on 9 of 10 data sets.

(5) According to the results, WELM1 or WELM2 assigns the weights to instances; the noise or outliers can still degrade WELM performance and are also able to achieve at least comparable performance with DNC.

Besides, *G*-mean [36] is frequently used to measure how well the predictor performs. Hence, this paper mainly chose *G*-mean to make the statistics analysis in Tables 5–7. For each data set, nine predictive models will be built following the algorithms' settings described in the previous section. The Friedman test is also used to detect significant difference among a group of results, and Holm's post hoc test is adopted to examine whether the proposed algorithms are distinctive among a $1 \times N$ comparison [41, 42]. The post hoc procedure allows us to know whether a hypothesis of comparison of means could be rejected at a specified level of significance α. The adjusted p value (APV) is also calculated to denote the lowest level of significance of a hypothesis which results in a rejection. Furthermore, we consider the average rankings of the algorithms to measure how good an algorithm is with respect to its partners. The ranking could be calculated by assigning a position to each algorithm depending on its performance on each data set. The algorithm that achieves the best performance on a specific data set will be given rank 1 (value 1); then the algorithm with the second-best result is assigned rank 2, and so forth. This task is conducted over all data sets and finally an average ranking is calculated.

In Table 5, this paper calculated its rankings, and APVs first. It can be found that FWELM-INTRA algorithm has acquired the lowest average ranking value, indicating that it is the best among all algorithms. As for FWELM-INTER algorithm, it ranks second, which is only a little worse than FWELM-INTER. In addition, we observe that the APVs of the ELM, ELM-RUS, and WELM2 are lower than a standard level of significance of 0.05. That means the three algorithms are significantly different from FWELM-INTRA algorithm. However, we cannot say that FWELM-INTRA is significantly different from FWELM-INTER, ELM-ROS, ELM-Smote, WELM1, and DNC, although it has a lower ranking value. Meanwhile, we know FWELM-INTRA is less complex than FWELM-INTER. Therefore, we can recommend that FWELM-INTRA would be an efficient choice for learning from SDP data sets.

Next, this paper selects FWELM-INTRA as the baseline to execute the one-versus-one comparison. Specifically, we calculated the average percentage of performance promotion and counted win/same/loss number based on the pairwise t-test at 5% significance level throughout all data sets. The results are presented in Table 6. The results show that the proposed FWELM-INTRA algorithm has promoted the classification performance compared with the other algorithms, and the performance can be also improved more or less (2.84%–41.76%). In addition, we observed that FWELM-INTRA is better than all other algorithms on 8 data sets at least, indicating its superiority. In contrast with FWELM-INTER algorithm, its average performance only increases by 0.59%. Therefore, we can say that the two proposed algorithms are fairly similar to each other.

Furthermore, effect size is also computed, since it emphasises the practical size of difference [43]. We use Cliff's delta [44], which is a nonparametric effect size measure that quantifies the amount of difference between two groups. In our context, Cliff's delta is computed to compare FWELM-INTRA with the other approaches. That is, $|\text{delta}| < 0.147$ is "negligible," $|\text{delta}| < 0.33$ and $|\text{delta}| \geq 0.147$ are "small," $|\text{delta}| < 0.474$ and $|\text{delta}| \geq 0.33$ are "medium," and otherwise it is "large." Table 7 presents Cliff's delta values for FWELM-INTRA compared with other approaches in terms of *G*-mean values. We observe that the effect size for FWELM-INTRA compared with ELM and WELM2 is large on all data sets, and the effect size for FWELM-INTRA compared with ELM-RUS, ELM-ROS, ELM-SMOTE, WELM1, and DNC fluctuated on different data sets. Moreover, the effect size for FWELM-INTRA compared with FWELM-INTER is basically negligible on the data sets. Therefore, the readers are suggested to choose an appropriate algorithm according to the characteristics of their practical applications.

4.5. RQ2: How Does the K Value Impact Performance?

4.5.1. Motivation. In the two proposed algorithms, we need to select the correct K for the relative density. If the value of K is too small, it would be failed to identify the noise and outliers. If the value of K is too large, some distinctions would not be captured. Therefore, it is necessary to know whether there is a fixed K that can be applied to all data sets.

4.5.2. Approach. To decide the value of K in relative density calculation to the classification performance, we choose the parameter K based on the number of instances. Figure 4 plots the variance of *G*-mean with the change of the

Table 2: Means and standard deviations of G-mean on the NASA data sets.

Data set	FWELM-INTER	FWELM-INTRA	ELM	ELM-RUS	ELM-ROS	ELM-SMOTE	WELM1	WELM2	DNC
mc2	0.6348 ± 0.0630	0.6419 ± 0.0455	0.5226 ± 0.0543	0.6330 ± 0.0454	0.5722 ± 0.0670	0.5934 ± 0.0495	0.6073 ± 0.0577	0.5521 ± 0.0467	0.5571 ± 0.1775
kc2	0.7858 ± 0.0071	0.7940 ± 0.0064	0.5945 ± 0.0113	0.7751 ± 0.0076	0.7767 ± 0.0074	0.7748 ± 0.0045	0.7800 ± 0.0066	0.7412 ± 0.0105	0.7604 ± 0.0754
jm1	0.6876 ± 0.0005	0.6611 ± 0.0020	0.3202 ± 0.0025	0.6625 ± 0.0012	0.6647 ± 0.0023	0.6643 ± 0.0025	0.6762 ± 0.0012	0.5852 ± 0.0015	0.6708 ± 0.0158
kc3	0.6228 ± 0.0375	0.6514 ± 0.0568	0.2302 ± 0.0445	0.5646 ± 0.0614	0.5660 ± 0.0466	0.5757 ± 0.0521	0.5826 ± 0.0755	0.4939 ± 0.0976	0.5863 ± 0.2172
kc1	0.7255 ± 0.0038	0.7443 ± 0.0051	0.4262 ± 0.0137	0.7156 ± 0.0032	0.7160 ± 0.0033	0.7117 ± 0.0031	0.7148 ± 0.0059	0.6913 ± 0.0061	0.7245 ± 0.0391
pc3	0.7622 ± 0.0075	0.7645 ± 0.0065	0.1831 ± 0.0433	0.7431 ± 0.0115	0.7471 ± 0.0104	0.7430 ± 0.0131	0.7477 ± 0.0098	0.6969 ± 0.0136	0.7416 ± 0.0608
pc4	0.8230 ± 0.0052	0.8104 ± 0.0079	0.5921 ± 0.0199	0.8081 ± 0.0072	0.8270 ± 0.0090	0.8168 ± 0.0102	0.8246 ± 0.0098	0.7925 ± 0.0045	0.8604 ± 0.0379
mw1	0.6243 ± 0.0311	0.6654 ± 0.0521	0.0185 ± 0.0382	0.5882 ± 0.0738	0.6220 ± 0.0525	0.6121 ± 0.0723	0.6198 ± 0.0548	0.5428 ± 0.0443	0.6111 ± 0.2506
cm1	0.7212 ± 0.0273	0.7333 ± 0.0091	0.0899 ± 0.0400	0.6960 ± 0.0538	0.7311 ± 0.0307	0.7225 ± 0.0322	0.7316 ± 0.0381	0.6646 ± 0.0454	0.6841 ± 0.1246
pc1	0.7461 ± 0.0184	0.7762 ± 0.0119	0.0889 ± 0.0174	0.6695 ± 0.0167	0.6832 ± 0.0222	0.7134 ± 0.0203	0.6744 ± 0.0245	0.5770 ± 0.0223	0.7402 ± 0.1046

Table 3: Means and standard deviations of AUC on the NASA data sets.

Data set	FWELM-INTER	FWELM-INTRA	ELM	ELM-RUS	ELM-ROS	ELM-SMOTE	WELM1	WELM2	DNC
mc2	0.7261 ± 0.0278	0.7409 ± 0.0455	0.7113 ± 0.0396	0.7331 ± 0.0387	0.6968 ± 0.0442	0.7309 ± 0.0374	0.7217 ± 0.0275	0.7134 ± 0.0258	0.6522 ± 0.1577
kc2	0.8425 ± 0.0058	0.8569 ± 0.0093	0.8420 ± 0.0062	0.8389 ± 0.0056	0.8409 ± 0.0049	0.8402 ± 0.0062	0.8423 ± 0.0063	0.8433 ± 0.0062	0.8137 ± 0.0832
jm1	0.7305 ± 0.0013	0.7099 ± 0.0012	0.7264 ± 0.0006	0.7302 ± 0.0008	0.7325 ± 0.0008	0.7308 ± 0.0013	0.7327 ± 0.0003	0.7337 ± 0.0006	0.7342 ± 0.0179
kc3	0.6930 ± 0.0288	0.7053 ± 0.0405	0.6705 ± 0.0354	0.6466 ± 0.0530	0.6904 ± 0.0276	0.6688 ± 0.0411	0.6823 ± 0.0158	0.6745 ± 0.0618	0.6800 ± 0.1581
kc1	0.8042 ± 0.0041	0.7962 ± 0.0025	0.7985 ± 0.0031	0.7984 ± 0.0032	0.8004 ± 0.0029	0.7984 ± 0.0030	0.8001 ± 0.0037	0.8034 ± 0.0031	0.8055 ± 0.0360
pc3	0.8261 ± 0.0043	0.8423 ± 0.0046	0.8268 ± 0.0057	0.8165 ± 0.0065	0.8253 ± 0.0041	0.8211 ± 0.0065	0.8246 ± 0.0057	0.8262 ± 0.0050	0.8028 ± 0.0665
pc4	0.9065 ± 0.0040	0.8958 ± 0.0081	0.9125 ± 0.0043	0.8929 ± 0.0051	0.9054 ± 0.0070	0.9045 ± 0.0067	0.9104 ± 0.0060	0.9044 ± 0.0062	0.9193 ± 0.0357
mw1	0.7921 ± 0.0431	0.8010 ± 0.0447	0.7790 ± 0.0328	0.7354 ± 0.0511	0.7881 ± 0.0346	0.7910 ± 0.0268	0.7882 ± 0.0369	0.7573 ± 0.0477	0.6997 ± 0.1923
cm1	0.8065 ± 0.0170	0.8211 ± 0.0101	0.8032 ± 0.0215	0.7873 ± 0.0296	0.8025 ± 0.0129	0.7935 ± 0.0214	0.8044 ± 0.0183	0.7994 ± 0.0224	0.7545 ± 0.1134
pc1	0.8114 ± 0.0115	0.8201 ± 0.0103	0.8305 ± 0.0075	0.8258 ± 0.0121	0.8432 ± 0.0064	0.8402 ± 0.0064	0.8420 ± 0.0096	0.8449 ± 0.0050	0.8492 ± 0.0738

Table 4: Means and standard deviations of Balance on the NASA data sets.

Data set	FWELM-INTER	FWELM-INTRA	ELM	ELM-RUS	ELM-ROS	ELM-SMOTE	WELM1	WELM2	DNC
mc2	0.6324 ± 0.0496	0.6352 ± 0.0438	0.5487 ± 0.0364	0.6282 ± 0.0352	0.5840 ± 0.0496	0.6015 ± 0.0395	0.6108 ± 0.0387	0.5681 ± 0.0342	0.5583 ± 0.1471
kc2	0.7784 ± 0.0084	0.7813 ± 0.0071	0.5602 ± 0.0097	0.7674 ± 0.0071	0.7695 ± 0.0085	0.7676 ± 0.0049	0.7727 ± 0.0069	0.7240 ± 0.0010	0.7511 ± 0.0742
jm1	0.6874 ± 0.0004	0.6909 ± 0.0021	0.3884 ± 0.0011	0.6689 ± 0.0012	0.6707 ± 0.0023	0.6708 ± 0.0026	0.6824 ± 0.0013	0.5602 ± 0.0012	0.6686 ± 0.0152
kc3	0.6203 ± 0.0267	0.6427 ± 0.0542	0.3913 ± 0.0273	0.5727 ± 0.0411	0.5840 ± 0.0256	0.5891 ± 0.0366	0.5987 ± 0.0405	0.5426 ± 0.0601	0.5983 ± 0.1535
kc1	0.7178 ± 0.0034	0.7127 ± 0.0055	0.4273 ± 0.0079	0.7126 ± 0.0035	0.7132 ± 0.0031	0.7093 ± 0.0025	0.7122 ± 0.0058	0.6773 ± 0.0064	0.7217 ± 0.0380
pc3	0.7522 ± 0.0070	0.7581 ± 0.0051	0.3350 ± 0.0081	0.7355 ± 0.0121	0.7415 ± 0.0100	0.7371 ± 0.0131	0.7426 ± 0.0097	0.6842 ± 0.0146	0.7287 ± 0.0536
pc4	0.8158 ± 0.0065	0.8069 ± 0.0067	0.5503 ± 0.0164	0.7959 ± 0.0081	0.8209 ± 0.0099	0.8109 ± 0.0106	0.8182 ± 0.0089	0.7802 ± 0.0051	0.8438 ± 0.0322
mw1	0.6388 ± 0.0250	0.6572 ± 0.0396	0.3011 ± 0.0167	0.6110 ± 0.0465	0.6449 ± 0.0390	0.6355 ± 0.0509	0.6399 ± 0.0379	0.5952 ± 0.0284	0.6219 ± 0.1820
cm1	0.7077 ± 0.0209	0.7276 ± 0.0069	0.3228 ± 0.0146	0.6929 ± 0.0371	0.7229 ± 0.0197	0.7169 ± 0.0222	0.7237 ± 0.0270	0.6628 ± 0.0302	0.6732 ± 0.1141
pc1	0.7358 ± 0.0178	0.7390 ± 0.0120	0.3189 ± 0.0080	0.6577 ± 0.0160	0.6715 ± 0.0212	0.7037 ± 0.0200	0.6714 ± 0.0144	0.5688 ± 0.0112	0.7266 ± 0.1078

Table 5: Statistic results of various algorithms.

Algorithm	Average ranking	APV
FWELM-INTRA	1.60	—
FWELM-INTER	1.90	0.64
ELM	8.00	1.80×10^{-7}
ELM-RUS	4.80	3.90×10^{-2}
ELM-ROS	3.30	4.10×10^{-1}
ELM-SMOTE	4.30	1.10×10^{-1}
WELM1	2.90	4.26×10^{-1}
WELM2	7.00	1.94×10^{-5}
DNC	3.80	2.51×10^{-1}

Table 6: Further statistic results of various algorithms in comparison with FWELM-INTRA as the baseline.

Algorithm	Average percentage of promotion (%)	Win/same/loss
FWELM-INTER	0.59	8/0/2
ELM	41.76	10/0/0
ELM-RUS	3.87	8/0/2
ELM-ROS	3.37	8/0/2
ELM-SMOTE	3.15	8/0/2
WELM1	2.84	8/0/2
WELM2	15.24	10/0/0
DNC	3.06	8/0/2

Table 7: Cliff's |delta| in terms of G-mean between FWELM-INTRA and other approaches.

Data set	FWELM-INTER	ELM	ELM-RUS	ELM-ROS	ELM-SMOTE	WELM1	WELM2	DNC
mc2	0.073	0.646	0.073	0.556	0.364	0.086	0.932	0.867
kc2	0.073	0.944	0.073	0.073	0.073	0.052	0.615	0.134
jm1	0.103	0.955	0.003	0.003	0.007	0.064	0.823	0.043
kc3	0.103	1.000	0.736	0.736	0.632	0.582	1.000	0.715
kc1	0.073	0.955	0.153	0.153	0.153	0.002	0.496	0.073
pc3	0.007	1.000	0.073	0.073	0.061	0.035	0.632	0.073
pc4	0.073	0.944	0.007	0.007	0.013	0.047	0.074	0.336
mw1	0.131	1.000	0.353	0.214	0.398	0.073	0.925	0.338
cm1	0.073	1.000	0.214	0.015	0.036	0.006	0.516	0.491
pc1	0.130	1.000	0.756	0.585	0.413	0.628	1.000	0.263

parameter K for all ten data sets based on the two proposed algorithms.

4.5.3. Result. In Figure 4, the X values in axis, 1–5, denote $K = \{\lceil \sqrt{N}/4 \rceil, t\lceil \sqrt{N}/2 \rceil n, q\,h\lceil \sqrt{N} \rceil, x\lceil 2\sqrt{N} \rceil 7, C\lceil 4\sqrt{N} \rceil\}$, respectively. We observe that although there are some fluctuations, the performance still presents a rising trend with the increase of K in the initial phases on most data sets. Then, it will arrive at the peak and, subsequently, the performance will decrease. That means when the value of K is undersize or oversize, the performance of the proposed algorithms will be deteriorative. Excessively low K might assign oversize weights for some noise and outliers, while excessively high K might make the weights belonging to the instances in the same category converge. Although there might be different optimal settings for the parameter K in two different proposed algorithms, Figure 4 still shows some useful guidance; that is, data sets can choose K between $\lceil \sqrt{N}/2 \rceil$ and $\lceil 2\sqrt{N} \rceil$, and the performance of proposed algorithm could be guaranteed in safety. That also illustrates that the parameter K could be $\lceil \sqrt{N} \rceil$ conservatively and empirically in the experiments. Users are suggested and encouraged to choose an appropriate value for the parameter K by themselves in detailed situation.

4.6. RQ3: How About Time-Complexity of the Two Proposed Algorithms?

4.6.1. Motivation. Without considering the time consumption of modeling WELM classifier, the computational complexity of the two proposed algorithms is affected by the distance calculation, finding the corresponding neighbor, and calculating the fuzzy membership value. Nevertheless, we should focus on the time-complexity to find the relationships with running time.

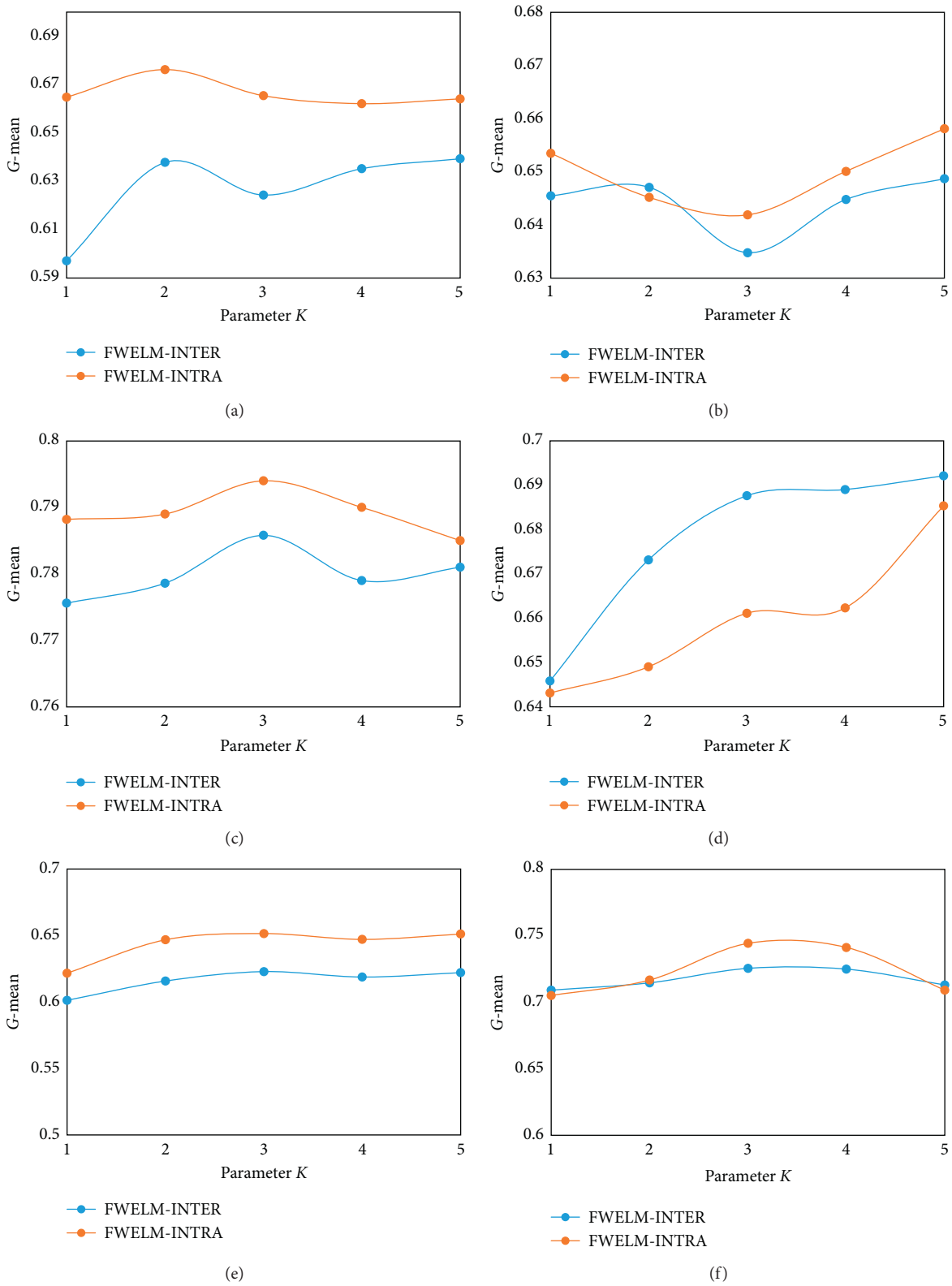

Figure 4: Continued.

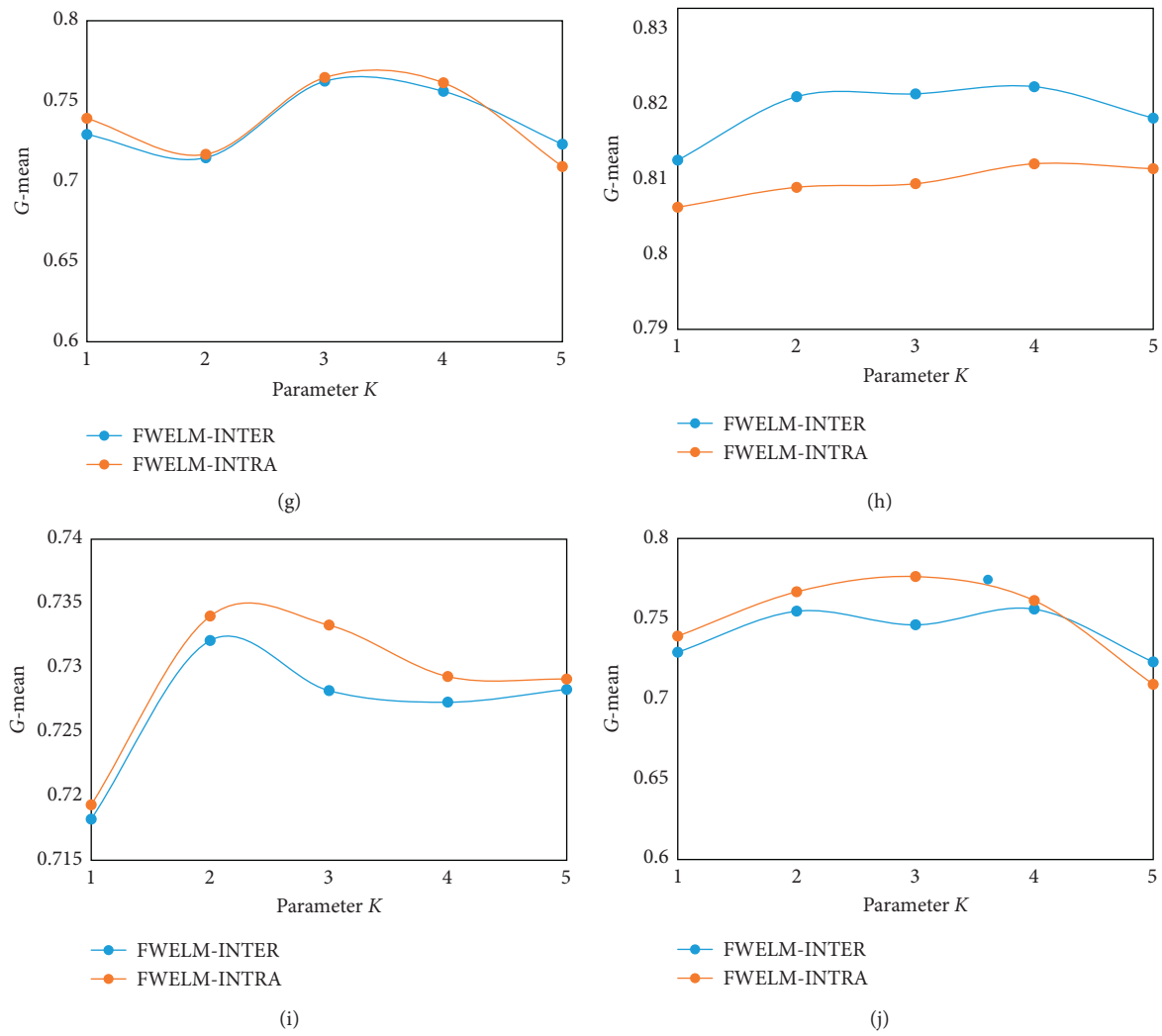

Figure 4: The variance of the performance with the change of parameter K. (a) mw1, (b) mc2, (c) kc2, (d) jm1, (e) kc3, (f) kc1, (g) pc3, (h) pc4, (i) cm1, and (j) pc1.

4.6.2. Approach. For a data set with n instances, where each instance holds a attributes, we firstly analyse that calculating distances will take $O(n^2a)$ time, sorting will take v time, and calculating and assigning fuzzy membership values will require a constant $O(n)$ time. Then, it can be found that FWELM-INTER needs to cost more training time than FWELM-INTRA, as the density calculation will be run twice.

For the two algorithms, the computational complexity depends on the calculation of the relative density and the fuzzy membership value for each training instance. Taking *jm1* as an example, this paper tracked the variance of the running time with the variance of the number of training instances n and the number of attributes a, respectively. When we track the variance of n, a is assigned a constant value of 2, while when we track the variance of a, n is fixed at 100. The range of n is 100, 500, 1000, 2000, and 10000 and $a = 1$, 5, 10, 15, and 20. Figure 5 presents the variance of running time with the variance of n and a, respectively.

4.6.3. Result. In Figure 5, the running time of the two algorithms is approximately linear with respect to the number of attributes a but exponential with respect to the number of training instances n. In addition, we found that FWELM-INTER always costs more running time than FSVM-INTRA and is consistent with the theory analysis listed above. It can be found that the computation of relative density is time-consuming, and the proposed algorithms scarify running time efficiency to acquire promotion of the learning model.

4.7. Threats to Validity. In this section, the threats to validity are described through internal aspect and external aspect.

Threads to internal validity concern the selection of data sets and methods. The research selected NASA data sets that have been commonly used in software prediction. These data sets are from software engineering and have different metrics. However, evaluating the proposed approach on a large scale of practical projects is always desirable. Meanwhile, for the NASA data sets, the parameter settings in the

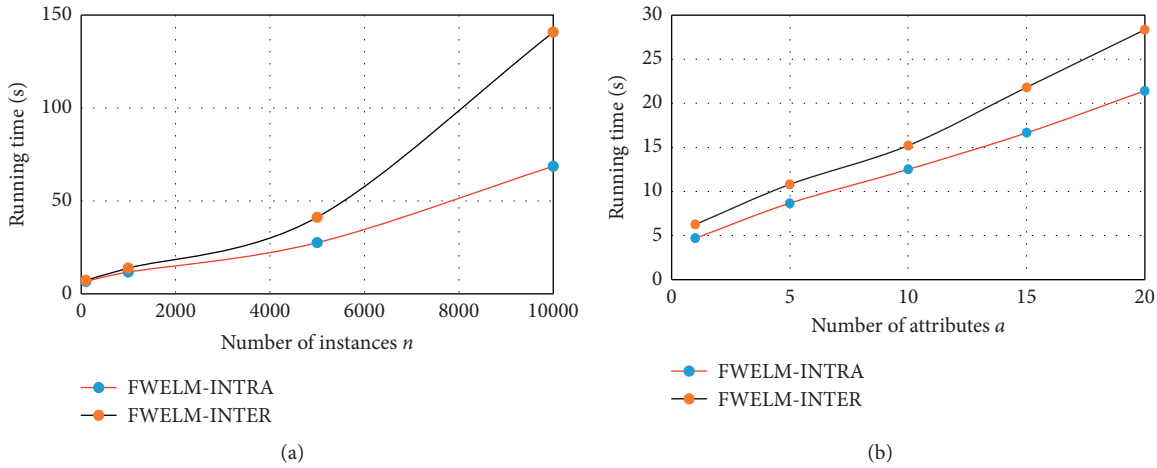

Figure 5: The variance of the running time with the change of instance and attribute. (a) Variance of instances. (b) Variance of attributes.

proposed algorithms are from the experimental results. Nevertheless, the parameters need to be selected based on the chosen data sets. Thus, the proposed method should be verified in more data sets.

Threads to external validity concern the possibility to generalize experimental results and comparisons with related algorithms. The proposed approach only requires that the metrics can be known and computed in the data sets, and the data sets are available. However, the selected data sets are open source and provide the metrics in our experiment. Meanwhile, we just compare some of state-of-the-art methods. Moreover, it is necessary to ask for support from more software developers in the companies to evaluate our approach and discuss the causal between software defects and software development. Further studies using different data sets, analysing metrics in detail, and comparing more algorithms may prove fruitful.

5. Conclusions

SDP aims at finding as many defective software modules as possible without hurting the overall performance of the constructed predictor. Although a lot of SDP models have been proposed in previous works, the imbalanced distribution between classes is still not considered well. By drawing the experience of previous works, this paper improved WELM to solve imbalance problem of SDP data set. Firstly, a novel measure called relative density was proposed to estimate the significance of each instance in feature space. Then, fuzzy membership functions are designed and replace the weights of WELM. Next, the algorithms of FWELM-INTRAN and FWELM-INTER were created to train SDP models. Finally, the algorithms were evaluated on ten real-world SDP data sets, and three performance measures were considered: G-mean, AUC, and Balance.

In order to prove the effectiveness and efficiency of our methods to tackle imbalanced SDP problems, this paper performed comparison with six class imbalance learning methods related to ELM (ELM, ELM-RUS, ELM-ROS, ELM-SMOTE, WELM1, and WELM2) and also performed comparison shown to have better performance than the two top-ranked predictors (Naive Bayes and Random Forest) in the SDP literature. From the results obtained, the proposed algorithms could result in significantly better classification results in the context of SDP.

In summary, the proposed algorithms are more robust as they are adaptive for different data distribution types and can more accurately estimate the significance of each instance and assign the identical total fuzzy coefficients for two different classes without considering the impact of data scale. Of course, they have several drawbacks as well, including quite high time-complexity induced by K-nearest-neighbors calculation, and the selection of the parameter K might also affect the quality of the classification model to some extent.

In future work, it will be interesting to develop more efficient cost-sensitive learning algorithms with low time-complexity. In addition, how to transform the proposed methods to address cross-project SDP problem effectively will be investigated, too.

Data Availability

The data and code used to support this study are available from the corresponding author upon request.

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 Scientific Research Foundation for the introduction of talent of Jiangsu University of Science and Technology, China; Natural Science Foundation of the Higher Education Institutions of Jiangsu Province, China (Grant no. 18JKB520011); Primary Research and Development Plan (Social Development) of

Zhenjiang City, China (Grant no. SH2019021); Natural Science Foundation of Jiangsu Province, China (Grant no. BK20191457); and high tech ship research project of Ministry of Industry and Information Technology.

References

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2011.

[2] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Information Sciences*, vol. 179, no. 8, pp. 1040–1058, 2009.

[3] H. Zhang and X. Zhang, "Comments on "data mining static code attributes to learn defect predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 9, pp. 2–13, 2007.

[4] 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.

[5] 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, no. 1, pp. 67–78, 2015.

[6] H. Yu, C. Sun, W. Yang, X. Yang, and X. Zuo, "Al-elm: one uncertainty-based active learning algorithm using extreme learning machine," *Neurocomputing*, vol. 166, pp. 140–150, 2015.

[7] 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.

[8] K. E. Bennin, J. Keung, A. Monden, Y. Kamei, and N. Ubayashi, "Investigating the effects of balanced training and testing datasets on effort-aware fault prediction models,"in *Proceedings of the 2016 IEEE 40thannual Computer software and applications conference(COMPSAC)*, vol. 1, pp. 154–163, IEEE, New York, NY, USA, 2016.

[9] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "The misuse of the nasa metrics data program data sets for automated software defect prediction," in *Proceedings of the 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*, pp. 96–103, IET, New York, NY, USA, 2011.

[10] L. Pelayo and S. Dick, "Applying novel resampling strategies to software defect prediction," in *Proceedings of the NAFIPS 2007–2007 Annual Meeting of the North American Fuzzy Information Processing Society*, pp. 69–72, IEEE, New York, NY, USA, 2007.

[11] Y. Kamei, A. Monden, S. Matsumoto, T. Kakimoto, and K. Matsumoto, "The effects of over and under sampling on fault-prone module detection," in *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pp. 196–204, IEEE, London, UK, 2007.

[12] 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, no. 1, pp. 321–357, 2002.

[13] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 434–443, 2013.

[14] 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.

[15] Y. P. Mack and M. Rosenblatt, "Multivariate *k*-nearest neighbor density estimates," *Journal of Multivariate Analysis*, vol. 9, no. 1, pp. 1–15, 1979.

[16] K. Fukunaga and L. Hostetler, "Optimization of *k* nearest neighbor density estimates," *IEEE Transactions on Information Theory*, vol. 19, no. 3, pp. 320–326, 1973.

[17] G. B. Huang, H. M. Zhou, X. J. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.

[18] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: a review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[19] W. Zong, G. B. Huang, and Y. Q. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.

[20] M. M. T. Thwin and T.-S. Quah, "Application of neural networks for software quality prediction using object-oriented metrics," *Journal of Systems and Software*, vol. 76, no. 2, pp. 147–156, 2005.

[21] T. M. Khoshgoftaar and N. Seliya, "Tree-based software quality estimation models for fault prediction," in *Proceedings of the Eighth IEEE Symposium on Software Metrics*, pp. 203–214, IEEE, London, UK, 2002.

[22] B. Turhan and A. Bener, "Analysis of Naive Bayes' assumptions on software fault data: an empirical study," *Data & Knowledge Engineering*, vol. 68, no. 2, pp. 278–290, 2009.

[23] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Using the support vector machine as a classification method for software defect prediction with static code metrics," in *Proceedings of the International Conference on Engineering Applications of Neural Networks*, pp. 223–234, Springer, Berlin, Germany, 2009.

[24] M. M. Öztürk, "Which type of metrics are useful to deal with class imbalance in software defect prediction?" *Information and Software Technology*, vol. 92, pp. 17–29, 2017.

[25] T. Menzies, B. Turhan, A. Bener, G. Gay, B. Cukic, and Y. Jiang, "Implications of ceiling effects in defect predictors," in *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering*, pp. 47–54, ACM, London, UK, 2008.

[26] T. M. Khoshgoftaar, E. Geleyn, L. Nguyen, and L. Bullard, "Cost-sensitive boosting in software quality modeling," in *Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering, 2002*, pp. 51–60, IEEE, Berlin, Germany, 2002.

[27] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, vol. 58, pp. 388–402, 2015.

[28] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[29] H. W. Tang and X. Z. Qin, *Practical Methods of Optimization*, Dalian University of Technology Press, Dalian, 2004.

[30] R. Batuwita and V. Palade, "Fsvm-cil: fuzzy support vector machines for class imbalance learning," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 3, pp. 558–571, 2010.

[31] 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.

[32] B. Ebo, J. Keung, P. Phannachitta, A. Monden, and S. M. Mahakil, "Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction," *IEEE Transactions on Software Engineering*, vol. 1, 2017.

[33] A. K. Jain and R. C. Dubes, "Algorithms for clustering data," 1988.

[34] G. M. Weiss, "Mining with rarity," *Acm Sigkdd Explorations Newsletter*, vol. 6, no. 1, p. 7, 2004.

[35] R. Batuwita and V. Palade, "Efficient resampling methods for training support vector machines with imbalanced datasets," in *Proceedings of the International Joint Conference on Neural Networks*, Berlin, Germany, 2010.

[36] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," *Icml*, vol. 97, pp. 179–186, 1997.

[37] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[38] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine Learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[39] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, 2006.

[40] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, vol. 7, no. 9, pp. 1263–1284, 2008.

[41] J. Demišar and D. Schuurmans, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006.

[42] 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.

[43] S. Wagner, M. Di Penta, A. Bener, L. Minku, B. Turhan, and H. Zhang in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering (Promise 2014)*, vol. 1, New York, NY, USA, 2014.

[44] N. Cliff, "Dominance statistics: ordinal analyses to answer ordinal questions," *Psychological Bulletin*, vol. 114, no. 3, pp. 494–509, 1993.