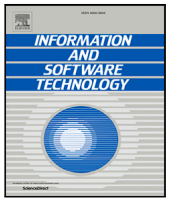
Information and Software Technology 142 (2022) 106742

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



The impact of the distance metric and measure on SMOTE-based techniques in software defect prediction

Shuo Feng [a,b], Jacky Keung [b], Peichang Zhang [c,*], Yan Xiao [d,*], Miao Zhang [b]

[a] School of Information Engineering, Zhengzhou University, Zhengzhou, China
[b] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, China
[c] College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China
[d] School of Computing, National University of Singapore, 117417, Singapore

ARTICLE INFO

Keywords:
Empirical software engineering
Defect prediction
Class imbalance
Distance metric
Synthetic minority oversampling TEchnique

ABSTRACT

Context: In software defect prediction, SMOTE-based techniques are widely adopted to alleviate the class imbalance problem. SMOTE-based techniques select instances close in the distance to synthesize minority class instances, ensuring few noise instances are generated.

Objective: However, recent studies show that selecting instances far away effectively increases the diversity and alleviates the overgeneralization brought by SMOTE-based techniques. To investigate the relationship between the distance of the selected instances and the performances of SMOTE-based techniques, we carry out this study.

Method: We first conduct experiments to empirically investigate the impact of the distance between the instances on the performances of three common SMOTE-based techniques. Based on the experimental result, we improve a recently proposed oversampling technique-SMOTUNED.

Results: The experimental results on five common classifiers across 30 imbalanced datasets from the PROMISE repository show that (1) the selection of the distance metric has little impact on the performances of SMOTE-based techniques, (2) as long as the number of synthesized noise instances is not beyond the noise-resistant ability of classifiers, the overall performances measured by AUC and *balance* of SMOTE-based techniques are not significantly affected by the distance between instances, and (3) the probability of detection (pd) and the probability of false alarm (pf) values of SMOTE-based techniques are significantly affected by the distance between the selected instances. The larger the distance between the selected instances is, the lower the pd and pf values SMOTE-based techniques obtain. The performance of the improved SMOTUNED is similar to that of the original SMOTUNED, but the improved SMOTUNED dramatically decreases the execution time of the original SMOTUNED.

Conclusion: By controlling the distance, different pd and pf values can be obtained. The diversity of SMOTE-based techniques can be improved, and the overgeneralization can be avoided.

1. Introduction

The class imbalance problem [1] present in software defect datasets significantly hinders the performance of prediction models in software defect prediction (SDP) [2]. Generally, there are more non-defective (i.e., the majority class or the negative) instances than defective (i.e., the minority class or the positive) ones. With the majority class instances dominating, the accuracy values of prediction models are mostly high, although most minority class instances are wrongly predicted as majority class instances. However, minority class instances are of more interest to software practitioners. If prediction models are biased toward the majority class instances instead of the minority class instances, they are less practical [3]. To deal with the class imbalance problem, several techniques have been proposed, which generally can be categorized into three general types, namely, (1) the data resampling technique [4–6], (2) the cost-sensitive learning technique [7–9], and (3) the ensemble learning technique [10–12]. Due to the ease of deployment and independence from machine learning algorithms, the data resampling technique is the most adopted in SDP studies [6,13,14]. Among different data resampling techniques, Synthetic Minority Oversampling TEchnique (SMOTE) [4], proposed

* Corresponding authors.
E-mail addresses: shuo.feng@hotmail.com (S. Feng), jacky.keung@cityu.edu.hk (J. Keung), pzhang@szu.edu.cn (P. Zhang), dcsxan@nus.edu.sg (Y. Xiao), miazhang9-c@my.cityu.edu.hk (M. Zhang).

https://doi.org/10.1016/j.infsof.2021.106742
Received 31 January 2021; Received in revised form 28 August 2021; Accepted 1 October 2021
Available online 16 October 2021

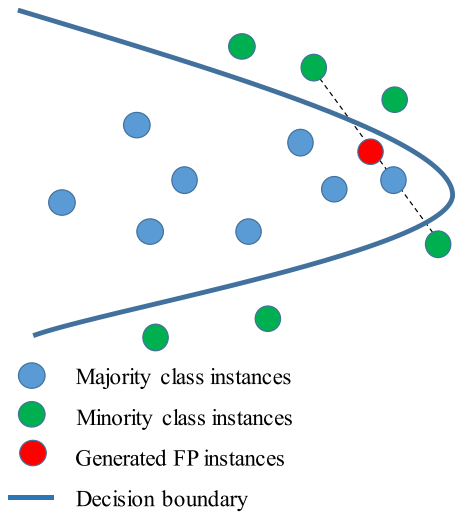


Fig. 1. The FP instances generated by SMOTE.

by Chawla et al. is the most prevalent. Based on SMOTE, several oversampling techniques were also developed, such as Adaptive Synthetic Sampling Approach [15] (ADASYN) and Borderline-SMOTE [5] (Borderline), which we refer to as SMOTE-based techniques in this study.

1.1. Motivation

SMOTE-based techniques employ the K-Nearest Neighbor (KNN) algorithm to select the minority class instances used to generate synthetic instances. The reason that SMOTE-based techniques employ the KNN algorithm is based on the common assumption [16] in the machine learning community that the similarity between instances can be measured by the distance between them, and the closer these instances are, the more similar they are. By selecting the minority class instances close in the distance, SMOTE-based techniques avoid generating too many synthetic noise instances and thus perform well. However, previous studies [17,18] have shown that selecting those instances close in the distance leads to generating near-duplicated instances and thus the overgeneralization of prediction models. To alleviate the overgeneralization and increase the diversity of the generated instances, Bennin et al. [6] proposed MAHAKIL, which adopts a different strategy from SMOTE-based techniques. MAHAKIL selects dissimilar and unrelated minority class instances to generate synthetic instances. In MAHAKIL, the distance between these selected instances is large. Based on the above assumption, MAHAKIL generates more noise instances than SMOTE-based techniques. Therefore, the performance of MAHAKIL should be inferior to those of SMOTE-based techniques. However, according to our previous study [19], the overall performance of MAHAKIL and those of SMOTE-based techniques are similar in terms of AUC and *balance*. This can be explained by Kim's study [20]. Kim's study shows that the common classifiers have a certain noise-resistant ability. Adding a certain amount of false positive (FP) instances into a dataset does not necessarily degrade the performances of the classifiers, as long as the number of FP instances is not beyond the noise-resistant ability of the classifiers. The synthetic noise instances generated by oversampling techniques all belong to FP instances, as shown in Fig. 1. The noise-resistant ability of the classifiers is beneficial to SMOTE-based techniques, because the procedure of SMOTE-based techniques generating synthetic instances is blind, which means generating FP instances is inevitable for SMOTE-based techniques.

Although the overall performances of MAHAKIL and SMOTE-based techniques are similar in terms of AUC and *balance*, there are significant differences between the performances of MAHAKIL and SMOTE-based techniques in terms of the *pd* (probability of detection) and *pf* (probability of false alarm) values. Specifically, MAHAKIL produces much lower *pf* values while SMOTE-based techniques obtain much higher *pd* values than MAHAKIL. Based on the performances of SMOTE-based techniques and MAHAKIL, we make the inference that the distance between the minority class instances used to generate synthetic instances does not impact the overall performances of SMOTE-based techniques, nor the distance metric does. Meanwhile, the *pd* and *pf* values are significantly affected by the distance. The larger the distance between the instances is, the lower the *pd* and *pf* values are. If the inference is correct, the effort spent on tuning the hyperparameter of the distance metric in SMOTE-based techniques could be saved. Besides, we could obtain different *pd* and *pf* values by controlling the distance between the minority class instances used to generate synthetic instances, which could make SMOTE-based techniques apply to different scenarios and also avoid the overgeneralization problem. For example, if high security is required, SMOTE-based techniques producing high *pd* values can be applied.

1.2. Methodology

In this study, we conduct empirical experiments to investigate the impact of the distance between instances on the performances of SMOTE-based techniques in SDP. Specifically, we investigate six distance metrics (i.e., the Manhattan, Euclidean, Chebyshev, Cosine, Hamming, and Correlation distances) in three SMOTE-based techniques (i.e., SMOTE, Borderline, and ADASYN) on five common classifiers (i.e., K-nearest neighbor (KNN), decision tree (DT), random forest (RF), naive Bayes (NB), and logistic regression (LR) classifiers) across 30 imbalanced datasets collected from the PROMISE repository [21]. Furthermore, we modify these three SMOTE-based techniques as two variants, i.e., K-farthest neighbor SMOTE-based techniques (KFN-SMOTE-based techniques) and Moderate SMOTE-based techniques (M-SMOTE-based techniques) to investigate the relationship between the distance and the performances of these techniques. The details of these variants are given in Section 4.1. We validate the performances of SMOTE-based, KFN-SMOTE-based, and M-SMOTE-based techniques by four common performance measures (i.e., the area under the ROC curve (AUC), *balance*, *pd*, and *pf*). We further analyze the performance by applying the Cliff's δ effect size [22], the Scott–Knott effect size difference test (the Scott–Knott ESD test) [23], and the Spearman correlation coefficient (SP) [24].

The experimental results show that the performances of SMOTE-based techniques with different distance metrics are similar in terms of AUC and *balance*, which indicates that the selection of the distance metric has little impact on the performances of SMOTE-based techniques. Based on this experimental result, we improve SMOTUNED [13], a recently proposed oversampling technique, by not optimizing the hyperparameter of the distance metric. The performance of the improved SMOTUNED is similar to that of the original SMOTUNED, but the execution time of the improved SMOTUNED dramatically decreases, which in turn confirms our findings. Meanwhile, the overall performances of SMOTE-based and M-SMOTE-based techniques are similar, and both are better than those of KFN-SMOTE-based techniques in terms of AUC and *balance*. In addition, SMOTE-based techniques obtain the highest *pd* and *pf* values, while KFN-SMOTE-based techniques obtain the lowest *pd* and *pf* values. We conclude that as long as the synthesized instances do not include too many noise instances beyond the noise-resistant ability of the classifiers [20], the overall performances of SMOTE-based techniques are not impacted by the distance between the minority class instances used to generate synthetic instances in terms of AUC and *balance*. However, the *pd* and *pf* values are significantly affected by the distance. The larger the distance between the selected instances is, the lower the *pd* and *pf* values of SMOTE-based techniques are. The statistical and correlation analyses confirm our findings.

1.3. Organization

The paper is organized as follows. We start by introducing the related work about this study in Section 2. Then the background of this study in Section 3 is introduced. Section 4 details the research questions, the datasets, the classifiers, the performance measures, and the experimental design to conduct our experiment. In Section 5, we show the experimental results and answer the research questions based on these results. Then we discuss the details of our work in Section 6. The threats to the validity of our work are discussed in Section 7. We round off the paper with the conclusions and the future work in Section 8.

2. Related work

According to the work conducted by Kim et al. [20], common classifiers have the certain noise-resistant ability. The overall performances of these classifiers do not decrease significantly even if a certain amount of noise is introduced. The noise-resistant ability of classifiers is beneficial to SMOTE-based techniques because the procedure of SMOTE-based techniques generating synthetic instances is blind, which means generating noise instances is inevitable for SMOTE-based techniques. However, Kim further pointed out that when the amount of noise is beyond a certain level, the overall performance of classifiers decreases dramatically.

Agrawal et al. [13] proposed an auto-tuning version of SMOTE named SMOTUNED, which automatically optimizes the hyperparameters of SMOTE. Specifically, SMOTUNED adopts the differential evolution (DE) [25] algorithm to explore the optimal values of the final defect ratio, the number of neighbors K, and the distance metric for SMOTE. SMOTUNED leads to a dramatically large improvement compared with the original SMOTE in terms of AUC. The superior performance of SMOTUNED indicates that optimizing the hyperparameters of SMOTE is an effective way to alleviate the class imbalance problem in SDP. However, due to the large searching space and the complexity of DE, the execution time of SMOTUNED significantly increases compared with the original SMOTE.

For the class imbalance problem, many prior empirical studies were conducted in SDP. For example, Bennin et al. [3] analyzed six data resampling approaches on five classifiers across 40 datasets. They found that the performances of data resampling approaches are dependent on the imbalance ratio, evaluation measure, and the selected classifiers. Tantithamthavorn et al. [26] conducted a more comprehensive empirical study, which adopted four common data resampling techniques and analyzed their performances on seven classifiers across 101 datasets in terms of ten performance measures. The experimental results show that the data resampling technique is beneficial when quality assurance teams want to increase AUC and pd values. Nevertheless, the data resampling technique also causes the concept drift [27], which leads to bias in the learned concepts. Song et al. [28] also conducted extensive experiments investigating the role of the imbalanced learning. Their experimental results show that an imbalanced learning method should be carefully chosen to ameliorate the imbalanced learning problem for SDP. Otherwise, the performance may be negatively affected by the imbalanced learning method. These empirical studies reveal some important facts about data resampling techniques. However, their work mostly focused on comparing each technique and omitted the fact that the performance of a certain data resampling technique could vary due to its different internal settings. Therefore, we chose different distance metrics and various distances to investigate the relationship between the performances of SMOTE-based techniques and these settings.

3. Background

3.1. SMOTE-based techniques

Synthetic Minority Oversampling TEchnique, proposed by Chawla et al. [4], is an oversampling technique, which generates synthetic minority class instances to tackle the class imbalance problem. SMOTE selects the minority class instances close in the distance and uses these instances to generate synthetic instances. SMOTE selects instances close in the distance to ensure that the pairwise instances used to generate synthetic instances are not too far away, reducing the possibility that the synthetic instances wrongly fall outside the region of the minority class. Therefore, SMOTE needs to tune the hyperparameters of the distance metric. The standard procedure of SMOTE is as follows: (1) A minority class instance is randomly selected. (2) Then, one of the selected instance's K nearest neighbor instances is randomly selected. (3) Finally, a new synthetic instance is randomly generated on the line between these two selected instances.

Based on SMOTE, Han et al. [5] proposed Borderline-SMOTE. Instead of treating each minority class instance equally, Borderline-SMOTE focuses more on the instances that lie on the decision boundary. It only uses those borderline instances to generate synthetic instances and thus strengthens the decision boundary. ADASYN, proposed by He et al. [15], optimizes the initial selection of the minority class instances used to generate synthetic instances. ADASYN assigns different weights to each minority class instance based on its difficulty level of classifying. The difficulty level is decided by the proportion of instances belonging to the majority class in the neighborhood. The larger the proportion, the higher the difficulty level of classifying the minority class instances, and the more weight will be allocated to those instances.

3.2. Distance metric

Distance is used to measure the similarity between instances. The Euclidean, Manhattan, and Chebyshev distances are all specific cases of the Minkowski distance. The Minkowski distance [29] is the generalized distance metric in a normed vector. Given two instances $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ from a dataset, the Minkowski distance $d(X, Y)$ between X and Y is defined as below:

$$d(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}, \tag{1}$$

where x_i, y_i is the ith feature of the instances X and Y. When p is 1 or 2, the Minkowski distance corresponds to the Manhattan distance or the Euclidean distance, respectively. When p is reaching infinity, the Chebyshev distance is obtained. The Euclidean distance is the most common distance metric applying to an extensive range of studies. It is more intuitive than other distance metrics. The Euclidean distance is a simple straight line between X and Y in the Euclidean space. The Manhattan distance is named by the way it measuring distance. Using the Manhattan distance is like walking in the block of Manhattan. The line representing the Manhattan distance between X and Y is a broken line instead of a straight line. It refers to the sum of distance along each dimension of X and Y. The Chebyshev distance is the greatest distance between X and Y along any individual dimension. Fig. 2 illustrates these three distance metrics. From Fig. 2, we can clearly see a broken line between instances A and B using the Manhattan distance. Meanwhile, the Euclidean distance and the Chebyshev distance are both simple straight lines. The Chebyshev distance only keeps the information of one dimension.

The Cosine distance is calculated based on the Cosine similarity [30]. The mathematical definition of the Cosine distance is given below:

$$\begin{aligned} d(X,Y) &= 1 - Cosine\ similarity \\ &= 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2}\sqrt{\sum_{i=1}^{n} {y_i}^2}}, \end{aligned} \tag{2}$$

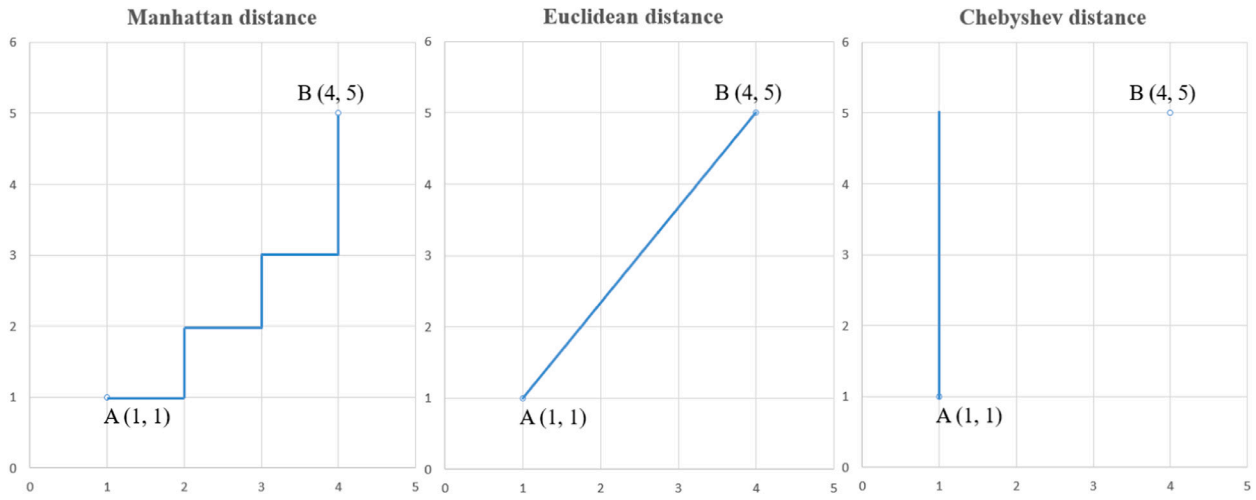


Fig. 2. The Manhattan, Euclidean, and Chebyshev distances.

where x_i, y_i is the ith feature of the instances X and Y. When the Cosine similarity equals 1, it means that X and Y are the same. When the Cosine similarity is equal to −1, X and Y are exactly the opposite, and if the Cosine similarity is equal to 0, X and Y are unrelated. The range of the Cosine distance is from 0 to 2.

The Correlation distance is similar to the Cosine distance. The Correlation distance is calculated based on the Correlation similarity. The mathematical definition of the correlation distance is as follow:

$$\begin{aligned} d(X,Y) &= 1 - Correlation\ similarity \\ &= 1 - \frac{dCov(X,Y)}{\sqrt{dVar(X)dVar(Y)}}, \end{aligned} \tag{3}$$

$dCov(,)$ is the distance covariance and $dVar(\cdot)$ represents the distance variance [31]. The range of the Correlation distance is from 0 to 1. When two instances are the same, the Correlation distance equals 0, and two instances are independent if the Correlation distance equals 1.

The Hamming distance [32] is defined as the minimum number of symbol changes needed to change one bitmap into the other given two bitstrings.

4. Experimental design

This section shows the research questions we investigate, the environment where all experiments are conducted, the details of the datasets used in the experiment, including the description of metrics, the number of instances, and the defect ratio of each software project. The description of the classifiers is also presented. The evaluation metrics and the statistical test are listed. At the end of this section, we explain the detailed setup of the experiment.

4.1. Research question

In this study, we investigate the following two research questions.

RQ1: Do different selections of the distance metrics impact the performances of SMOTE-based techniques?

To answer this question, we compare the performances of three common SMOTE-based techniques (i.e., SMOTE, ADASYN, and Borderline) with six distance metrics (i.e., the Manhattan, Euclidean, Chebyshev, Cosine, Hamming, and Correlation distances). If the performances of SMOTE-based techniques with different distance metrics are significantly different, we can conclude that the different selections of the distance metrics impact the performances of SMOTE-based techniques, which means that when SMOTE-based techniques are applied, the selection of the distance metric deserves careful consideration. On the contrary, if different selections of the distance metrics do not impact the performances of SMOTE-based techniques, we could save the effort of tuning this hyperparameter.

RQ2: Does the distance between the minority class instances used to generate synthetic instances impact the performances of SMOTE-based techniques?

To answer this question, we further modify these three SMOTE-based techniques as follows:

(1) the original SMOTE-based techniques (SMOTE, Borderline, and ADASYN) employ the K-nearest neighbor (KNN) algorithm and select the minority class instances close in the distance to generate synthetic instances.
(2) K-farthest neighbor SMOTE-based techniques (KFN-SMOTE, KFN-Borderline, and KFN-ADASYN) select one minority class instance and one of its K farthest neighbor minority class instances to generate synthetic instances.
(3) Moderate SMOTE-based techniques (M-SMOTE, M-Borderline, and M-ADASYN) randomly select two minority class instances to generate synthetic instances. In other words, the distance between the minority class instances selected by M-SMOTE-based techniques is farther than that of SMOTE-based techniques and closer than that of KFN-SMOTE-based techniques. M-SMOTE-based techniques can be treated as SMOTE-based techniques just with the K value equaling $N - 1$. Here N refers to the number of minority class instances in a dataset.

We show SMOTE, KFN-SMOTE, and M-SMOTE in Fig. 3, where we set the K value to be 3. For SMOTE, the minority class instance A can randomly select one instance from its K nearest neighbor instances B, C, and D to generate a synthetic instance. For KFN-SMOTE, the instance A can randomly select one instance from its K farthest neighbor instances E, F, and G. For M-SMOTE, the instance A can randomly select any one minority class instance from the whole dataset (i.e., instances B, C, D, E, F, and G) to generate a synthetic instance.

We compare the performances of SMOTE-based, M-SMOTE-based, and KFN-SMOTE-based techniques. SMOTE-based techniques select the instances closer in distance than M-SMOTE-based and KFN-SMOTE-based techniques to generate synthetic instances. M-SMOTE-based techniques select the instances closer in distance than KFN-SMOTE-based techniques. If the performances of SMOTE-based, M-SMOTE-based, and KFN-SMOTE-based techniques are similar, we could conclude that the distance does not impact the performances of these techniques. Otherwise, it indicates that the distance is an important factor that impacts the performances of SMOTE-based techniques. Future research on the oversampling technique should take the distance into the consideration. Thus, we can answer RQ2.

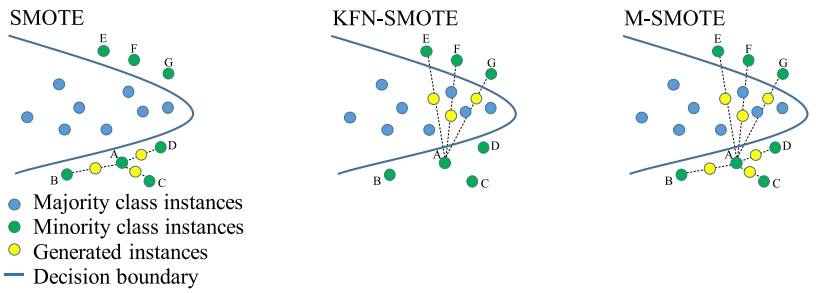


Fig. 3. SMOTE, KFN-SMOTE, and M-SMOTE.

4.2. Environment

In this study, the CPU used is an Intel Core i8-8700 K 3.7 GHz. The memory is 32 GB RAM, and the operating system is Microsoft Windows 10 Enterprise 64-bit.

4.3. Datasets

In this study, we use the datasets collected from the PROMISE repository [21]. There are 41 datasets measured by the static code metric [33]. Because we focus on the class imbalance problem, we only select the datasets whose defect ratio is less than 50%. These datasets were widely adopted by many previous studies [6,34–36] and showed good performance. They are all open-source and easily collected, making it easy for others to compare and replicate our work. We list the detailed information of these datasets, including the project names, the versions, the number of the total instances, the defective instances, and the defect ratio in Table 1. There are 20 metrics to measure the complexity of instances in these selected datasets. In addition, there is a metric indicating whether the current instance is defective or not. If the current instance is non-defective, the metric is labeled as 0. The metric is labeled as one if the instance is defective. The details of these metrics are presented in Table 2.

Table 1
Description of 30 datasets collected from the PROMISE repository.

Dataset	# Instances	# Defect	% Defect ratio
ant-1.3	125	20	16
ant-1.4	178	40	22.5
ant-1.5	293	32	10.9
ant-1.6	351	92	26.2
ant-1.7	745	166	22.3
camel-1.0	339	13	3.8
camel-1.2	608	216	35.5
camel-1.4	872	145	16.6
camel-1.6	965	188	19.5
ivy-1.4	241	16	6.6
ivy-2.0	352	40	11.4
jedit-3.2	272	90	33.1
jedit-4.0	306	75	24.5
jedit-4.1	312	79	25.3
jedit-4.2	367	48	13.1
jedit-4.3	492	11	2.2
log4j-1.0	135	34	25.2
log4j-1.1	109	37	33.9
lucene-2.0	195	91	46.7
poi-2.0	314	37	11.8
synapse-1.0	157	16	10.2
synapse-1.1	222	60	27.0
synapse-1.2	256	86	33.6
velocity-1.6	229	78	34.1
xalan-2.4	723	110	15.2
xalan-2.5	803	387	48.2
xalan-2.6	885	411	46.4
xerces-1.2	440	71	16.1
xerces-1.3	453	69	15.2
xerces-init	162	77	47.5

4.4. Classifiers

In this study, we adopt five common classifiers (i.e., K-nearest neighbor (KNN), decision tree (DT), random forest (RF), naive Bayes (NB), and logistic regression (LR)), which have been widely used in many previous studies. Their performances are pretty satisfactory. Adopting these five classifiers could make our conclusion more general and convenient for others' replication. Note that we focus on investigating the impact of the distance instead of tuning the hypermeters of these classifiers and avoiding reinventing the wheel. Therefore, we adopt the sklearn package [37] to implement the selected classifiers, and all the hyperparameters for these classifiers are set to be the default values.

4.5. Performance measures

Performances of prediction models are usually evaluated by analyzing the results computed from a confusion matrix. A typical confusion matrix is defined in . By convention, defective instances are considered as positive and non-defective instances as negative. (See Tables 3, 6, 7 and 9)

In SDP, using the overall accuracy to evaluate the performance of prediction models is inappropriate because of the class imbalance problem. Even if all instances are predicted as non-defective, the overall accuracy remains high because there are many more non-defective instances than defective ones. Therefore, we choose performance measures that are not severely affected by the class imbalance problem. In this study, we choose AUC, $balance$, pd, and pf. These performance measures were widely adopted in many studies [3,38,39]. The higher values of AUC, $balance$, and pd and the lower values of pf represent the better performance. The mathematical definitions of $balance$, pd, and pf are given as below:

$$balance = 1 - \frac{\sqrt{(0-pf)^2 + (1-pd)^2}}{\sqrt{2}} \tag{4}$$

$$pd = \frac{TP}{TP + FN} \tag{5}$$

$$pf = \frac{FP}{TN + FP} \tag{6}$$

4.6. Performance comparison

In this study, because we aim to guide the practical use of SMOTE-based techniques, We first use the effect size (i.e., Cliff's δ) to quantify the difference between each distance metric across every single dataset. We adopt the effect size because it is independent of sample size and more tied to the magnitude of the experimental results [40]. Specifically, a practical effect size between two distance metrics indicates that the effort spent on tuning the parameter of distance metrics is worthy. Otherwise, the effort can be saved. For the interpretation of the effect size, we follow Kampenes [41]. The effect size is interpreted as negligible ($0 <$ Cliff's $\delta < 0.147$), small ($0.147 <$ Cliff's $\delta < 0.33$),

Table 2
Description of the metrics.

Abbreviation	Description
WMC	Weighted methods per class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling between object classes
RFC	Response for a Class
LCOM	Lack of cohesion in methods
CA	afferent couplings
CE	efferent couplings
NPM	Number of Public Methods
LCOM3	Lack of cohesion in methods, different from LCOM
LOC	Lines of Code
DAM	Data Access Metric
MOA	Measure of Aggregation
MFA	Measure of Functional Abstraction
CAM	Cohesion Among Methods of Class
IC	Inheritance Coupling
CBM	Coupling Between Methods
AMC	Average Method Complexity
MAX(CC)	Maximum value of CC methods of the investigated class
AVG(CC)	Arithmetic mean of the CC value in the investigated class

Table 3
Confusion matrix.

	Predicted positive	Predicted negative
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

medium (0.33 < Cliff's δ < 0.474), and large (Cliff's δ > 0.474), respectively. Besides, we employ the Scott–Knott Effect Size Difference test (the Scott–Knott ESD test) [23] to compare the values of these performance measures of these techniques to investigate whether there exist statistically significant differences among them. The Scott–Knott ESD test is an algorithm that compares multiple results considering the effect size. It divides results into two different groups if the two groups have a significant difference. Then the procedure will be repeated if there is still a significant difference in each group. When no group can be further created, the Scott–Knott ESD test is terminated. The results in the different groups are significantly different. We also employ the Spearman correlation coefficient (SP) [42] to calculate the correlation between the distance and the performances of SMOTE-based techniques. SP is a special case of the Pearson correlation coefficient (PE) [43]. Unlike PE, SP uses the ranks instead of the values of the variables. We adopt SP instead of PE because SP is less sensitive to noise than PE, and the datasets adopted in SDP usually contain noisy instances.

4.7. Experimental design

Since distance is sensitive to the scale of data, we first apply the min–max normalization method to the original datasets to adjust all the features into the range from 0 to 1, which reduces the negative impact of different scales of different features. Then we adopt the 5-fold cross-validation to divide the experimental datasets using the stratification method to generate the training and testing data. Employing the 5-fold cross-validation to generate the training and testing data is quite common in SDP as well as the class imbalance problem [44,45]. Besides, SMOTUNED also employs the 5-fold cross-validation to explore the optimal values of the hyperparameters. We adopt the stratification method to split the datasets to ensure that there are always defective instances existing in all five folds and also keep the proportion of the defective instances to the non-defective instances remaining the same as the original datasets. After being split, four folds are used as the training data and the left fold as the testing data. Then we apply SMOTE-based techniques with different metrics to oversample only the training data and keep the testing data unchanged. Based on the conclusion of Ahmad Abu [46] that oversampling techniques attain the best performance at 50%, we terminate these techniques when the number of defective and non-defective instances are equal in the training data. The prediction models are trained with the oversampled training data, and the performances of the prediction models are validated by the original testing data. We iterate this procedure five times to ensure all five folds have been used once as the training and testing data. For more robust and converging results, we further run the above procedure 100 times to handle the random variation and sample bias. For each dataset, we obtain $5 \times 100 = 500$ outcomes for each distance metric in terms of each performance measure on each dataset. We take the 500 outcomes as a distribution of each distance metric. Then we apply the distribution of each distance metric to each statistical test adopted in this study. Fig. 4 presents the experimental framework in this study.

5. Experimental results and analysis

In this section, we present the heatmap of the effect sizes between SMOTE-based techniques with different distance metrics on the KNN, DT, RF, NB, and LR classifiers in terms of AUC and *balance*. We further present the Scott–Knott ESD test ranking of SMOTE-based, KFN-SMOTE-based, and M-SMOTE-based techniques on the selected classifiers across 30 datasets in terms of AUC, *balance*, *pd*, and *pf*. Then we examine the Spearman correlation coefficient (SP). We exhibit the experimental results via answering the research questions.

RQ1: Do different selections of the distance metrics impact the performances of SMOTE-based techniques?

Fig. 5 presents the heatmap of the effect sizes between the performance of SMOTE, Borderline, and ADASYN adopting different distance metrics in terms of AUC. Each 6 × 6 matrix compares the effect sizes between the six distance metrics on the combination of each dataset and each classifier. There are 150 (30 datasets × 5 classifiers) combinations of the datasets and the classifiers for SMOTE, Borderline, and ADASYN, respectively. The pink cell in Fig. 5 indicates that the effect sizes between any two distance metrics are negligible. The orange cell represents the effect size is small, and the red cell represents the effect size is larger than small. From Fig. 5(a), we can see that the selection of the distance metric has negligible impact on the performance of SMOTE across most datasets on the five classifiers. Among the 150 combinations, only two combinations have an effect size larger than small. Twenty-one combinations have a small effect size, and the effect sizes of 127 of the 150 combinations are all negligible. For Borderline, eight combinations achieves the effect sizes larger than small, and 45 combinations achieves the small effect sizes. The rest of the 150 combinations are negligible. For ADASYN, the effect sizes of three combinations are larger than small. Fifty-one combinations obtain the small effect sizes, and 96 combinations achieve the negligible effect sizes.

Fig. 6 shows the heatmap of the effect sizes between the performance of SMOTE, Borderline, and ADASYN adopting different distance metrics in terms of *balance*. It also can be seen that the selection of distance metrics has little impact on the performance of SMOTE-based techniques.

To complete our experiment, we modify a recently proposed oversampling technique-SMOTUNED [13]. Specifically, the original SMOTUNED automatically optimizes three hyperparameters of SMOTE: the final defect ratio, the number of neighbors K, and the distance metric. The tuning range of these hyperparameters follows the setting of the original paper, which is shown in Table 4. From Table 4, it can be seen that there are only 20 options for K and only four options for the final defect ratio. As to the distance metric, it has a continuous range of values from 0.1 to 5. Clearly, the execution time of SMOTUNED is mostly governed by the operation exploring the optimal distance metric. By not optimizing the distance metric, the searching space of DE

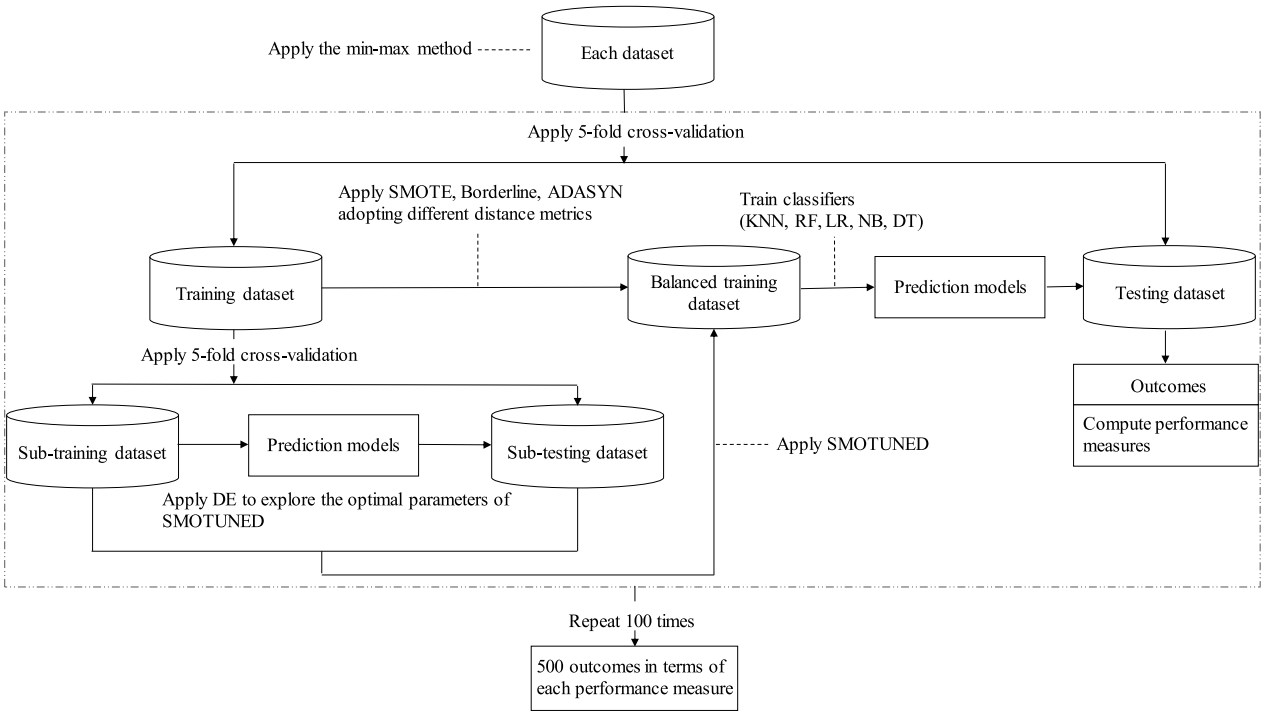


Fig. 4. Experimental framework.

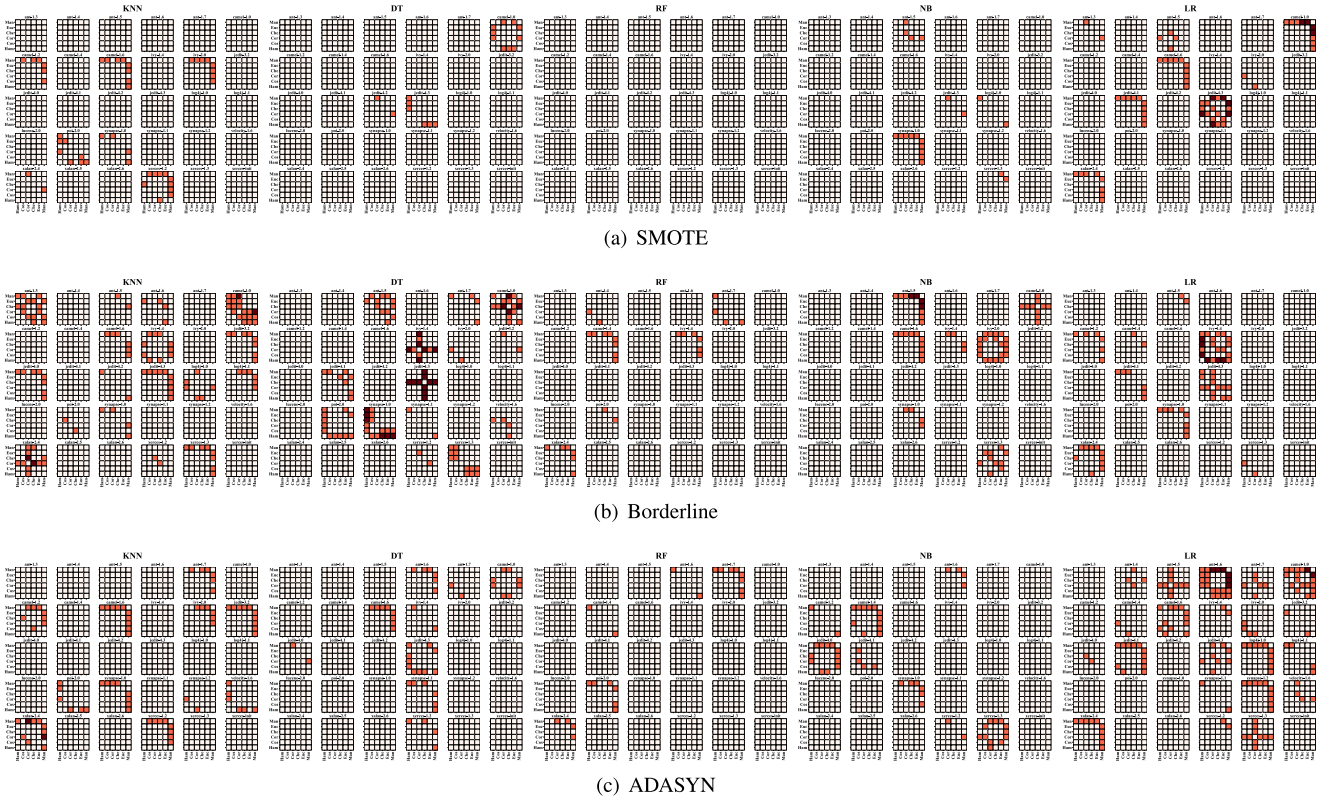
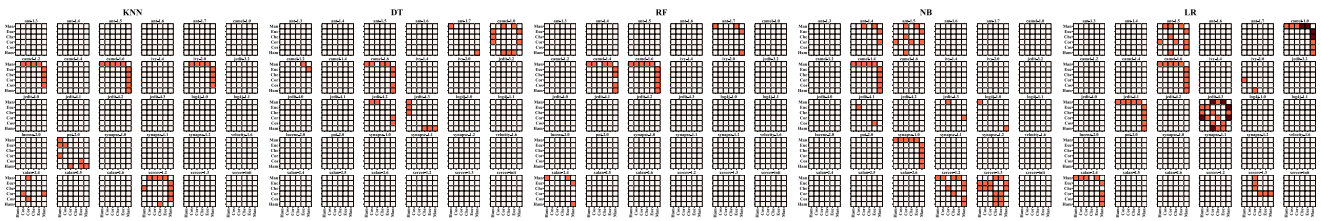


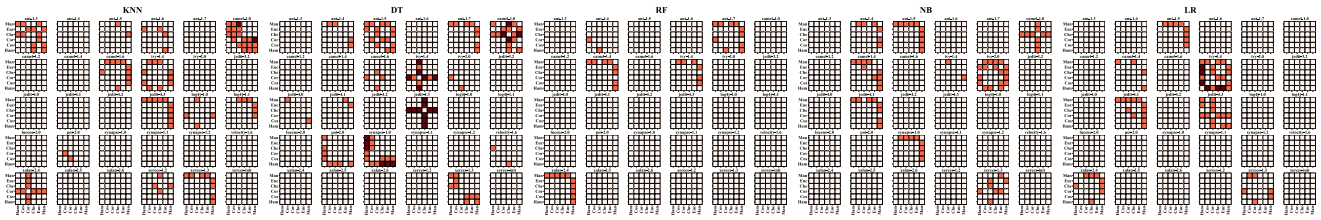
Fig. 5. The heatmap of the effect sizes between the performance of SMOTE, Borderline, and ADASYN adopting different distance metrics in terms of AUC (The pink cell represents the effect size is negligible, the orange cell represents the effect size is small, and the red cell represents the effect size is larger than small). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

could significantly decrease, which could lead to a significant decrease in the execution time of SMOTUNED.

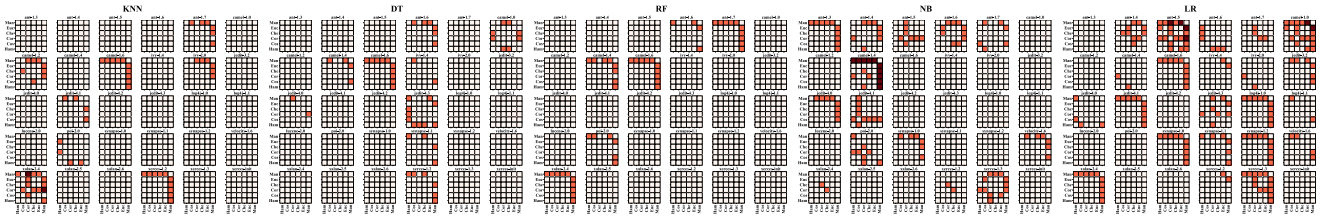
To improve SMOTUNED, we modify SMOTUNED by only optimizing the final defect ratio and K values and not optimizing the distance metric. As to the distance metric, we set it to be fixed as the Euclidean distance, which is commonly adopted as the default distance metric in the previous studies. For the other setting, we follow all the settings as the original paper does. The stop criterion *StdDev* [47] is



(a) SMOTE



(b) Borderline



(c) ADASYN

Fig. 6. The heatmap of the effect sizes between each distance metric in terms of *balance* (Pink represents the effect size is negligible, orange represents the effect size is small, and red represents the effect size is larger than small). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Tuning range of the hyperparameters of SMOTE.

Hyperparameters	Values
The number of neighbors K	[1, 2, 3, …, 19, 20]
The final defect ratio %	[50, 100, 200, 400]
The distance metric	[0.1, 5]

adopted. When the standard deviation of positions of all members is below 0.001, the optimization run is stopped, and the convergence is considered to be achieved. If there is no significant difference between the performances of the modified SMOTUNED and the original SMOTUNED, it further proves that the selection of the distance metrics has little impact on the performances of SMOTE-based techniques. To ease the demonstration, we refer to the improved SMOTUNED as I-SMOTUNED.

We record the median AUC values of every single dataset on the four classifiers in Tables 5 to Table 8. We also calculate the effect sizes between SMOTUNED and I-SMOTUNED across every single dataset. Because we repeat the experiments 100 times, we record the median execution time of 100-time 5-fold cross-validation. The column "Time" in these tables represents the median execution time of 100-time 5-fold cross-validation and is specified in seconds. From these tables, we can see that the performance of I-SMOTUNED is quite similar to that of SMOTUNED on all classifiers in terms of the median AUC values. Moreover, the effect sizes between the performances of I-SMOTUNED and SMOTUNED are negligible across most datasets on the five classifiers. Only 0, 3, 1, 1, and 2 out of 30 datasets on the KNN, DT, RF, LR, and NB classifiers achieve small effect sizes, and no datasets obtains an effect size larger than small. Besides, the execution time of SMOTUNED is much longer than that of I-SMOTUNED. Specifically, the median execution time of I-SMOTUNED is only 56.7%, 62.4%, 57.4%, 57.3%, and 46.8% of SMOTUNED on the KNN, DT, RF, NB,

Table 5
Comparison between the execution time and the performances of SMOTUNED and I-SMOTUNED on the KNN classifier in terms of AUC.

Dataset	SMOTUNED		I-SMOTUNED		Cliff's δ
	AUC	Time (s)	AUC	Time (s)	
ant-1.3	0.672	40.1	0.667	24.9	Negligible
ant-1.4	0.586	144.4	0.579	88.0	Negligible
ant-1.5	0.734	108.0	0.735	58.1	Negligible
ant-1.6	0.725	129.7	0.715	82.0	Negligible
ant-1.7	0.737	777.3	0.739	282.6	Negligible
camel-1.0	0.601	416.7	0.590	225.4	Negligible
camel-1.2	0.571	802.3	0.565	178.1	Negligible
camel-1.4	0.610	1472.8	0.599	594.0	Negligible
camel-1.6	0.679	1211.2	0.672	295.2	Negligible
ivy-1.4	0.532	73.7	0.534	55.6	Negligible
ivy-2.0	0.683	271.3	0.695	118.1	Negligible
jedit-3.2	0.754	111.1	0.746	107.2	Negligible
jedit-4.0	0.751	112.8	0.745	104.1	Negligible
jedit-4.1	0.736	180.3	0.732	39.9	Negligible
jedit-4.2	0.703	274.6	0.712	62.2	Negligible
jedit-4.3	0.752	512.3	0.753	412.3	Negligible
log4j-1.0	0.735	42.4	0.732	40.1	Negligible
log4j-1.1	0.789	43.7	0.794	20.1	Negligible
lucene-2.0	0.631	31.5	0.632	22.7	Negligible
poi-2.0	0.661	322.7	0.661	205.7	Negligible
synapse-1.0	0.725	58.9	0.740	50.8	Negligible
synapse-1.1	0.693	107.7	0.690	56.4	Negligible
synapse-1.2	0.716	54.2	0.715	39.1	Negligible
velocity-1.6	0.703	156.2	0.701	110.1	Negligible
xalan-2.4	0.684	451.3	0.696	369.2	Negligible
xalan-2.5	0.648	305.9	0.646	130.2	Negligible
xalan-2.6	0.740	91.2	0.740	56.0	Negligible
xerces-1.2	0.680	413.1	0.672	307.1	Negligible
xerces-1.3	0.747	155.6	0.756	73.9	Negligible
xerces-init	0.692	16.1	0.701	9.2	Negligible
Median	0.698	150.0	0.701	85.0	

Table 6
Comparison between the execution time and the performances of SMOTUNED and I-SMOTUNED on the DT classifiers in terms of AUC.

Dataset	SMOTUNED		I-SMOTUNED		Cliff's δ
	AUC	Time (s)	AUC	Time (s)	
ant-1.3	0.627	281.6	0.642	254.4	Negligible
ant-1.4	0.595	416.0	0.617	181.2	Negligible
ant-1.5	0.610	316.6	0.610	149.3	Negligible
ant-1.6	0.693	188.0	0.693	43.4	Negligible
ant-1.7	0.652	1047.4	0.658	682.6	Negligible
camel-1.0	0.500	311.8	0.514	190.2	Negligible
camel-1.2	0.592	1180.7	0.586	196.3	Negligible
camel-1.4	0.610	967.0	0.606	818.1	Negligible
camel-1.6	0.597	1175.4	0.611	716.8	Small
ivy-1.4	0.531	204.4	0.548	187.2	Negligible
ivy-2.0	0.619	1180.4	0.624	578.8	Negligible
jedit-3.2	0.689	248.0	0.684	64.5	Negligible
jedit-4.0	0.670	141.4	0.668	78.6	Negligible
jedit-4.1	0.691	195.8	0.685	128.8	Negligible
jedit-4.2	0.664	1054.2	0.649	240.9	Negligible
jedit-4.3	0.678	218.0	0.711	183.2	Small
log4j-1.0	0.612	131.5	0.632	69.8	Negligible
log4j-1.1	0.701	82.3	0.674	94.9	Small
lucene-2.0	0.611	76.1	0.622	23.9	Negligible
poi-2.0	0.618	237.0	0.612	111.8	Negligible
synapse-1.0	0.585	68.1	0.589	67.4	Negligible
synapse-1.1	0.660	87.3	0.656	43.3	Negligible
synapse-1.2	0.664	456.2	0.676	218.4	Negligible
velocity-1.6	0.668	160.6	0.678	110.9	Negligible
xalan-2.4	0.638	1482.7	0.633	1067.8	Negligible
xalan-2.5	0.666	937.7	0.666	280.0	Negligible
xalan-2.6	0.711	193.8	0.712	135.3	Negligible
xerces-1.2	0.644	283.9	0.652	260.0	Negligible
xerces-1.3	0.672	394.5	0.674	126.3	Negligible
xerces-init	0.694	15.1	0.694	12.8	Negligible
Median	0.648	264.8	0.651	165.3	

Table 7
Comparison between the execution time and the performances of SMOTUNED and I-SMOTUNED on the RF classifiers in terms of AUC.

Dataset	SMOTUNED		I-SMOTUNED		Cliff's δ
	AUC	Time (s)	AUC	Time (s)	
ant-1.3	0.629	320.4	0.635	182.3	Negligible
ant-1.4	0.645	921.2	0.638	359.1	Negligible
ant-1.5	0.642	625.7	0.639	360.8	Negligible
ant-1.6	0.744	497.4	0.751	148.6	Negligible
ant-1.7	0.718	1691.8	0.719	544.8	Negligible
camel-1.0	0.538	949.0	0.520	423.5	Negligible
camel-1.2	0.609	1242.6	0.596	324.2	Small
camel-1.4	0.643	1592.2	0.643	850.4	Negligible
camel-1.6	0.622	1315.4	0.622	1183.6	Negligible
ivy-1.4	0.536	685.0	0.538	242.2	Negligible
ivy-2.0	0.668	309.9	0.656	308.3	Negligible
jedit-3.2	0.745	302.0	0.751	150.4	Negligible
jedit-4.0	0.717	118.9	0.714	98.9	Negligible
jedit-4.1	0.745	160.1	0.746	109.5	Negligible
jedit-4.2	0.696	250.0	0.702	223.3	Negligible
jedit-4.3	0.665	149.1	0.665	135.3	Negligible
log4j-1.0	0.675	236.5	0.669	136.9	Negligible
log4j-1.1	0.751	110.7	0.742	106.2	Negligible
lucene-2.0	0.651	50.5	0.654	42.9	Negligible
poi-2.0	0.646	793.1	0.644	268.8	Negligible
synapse-1.0	0.599	296.7	0.613	179.4	Negligible
synapse-1.1	0.704	164.3	0.704	113.7	Negligible
synapse-1.2	0.732	119.3	0.726	110.2	Negligible
velocity-1.6	0.718	386.8	0.714	114.9	Negligible
xalan-2.4	0.675	1022.0	0.672	633.1	Negligible
xalan-2.5	0.689	331.3	0.689	140.2	Negligible
xalan-2.6	0.754	169.3	0.753	129.0	Negligible
xerces-1.2	0.687	420.3	0.683	269.8	Negligible
xerces-1.3	0.707	289.9	0.700	193.1	Negligible
xerces-init	0.749	37.7	0.745	36.4	negligible
Median	0.681	315.2	0.678	180.9	

Table 8
Comparison between the execution time and the performances of SMOTUNED and I-SMOTUNED on the LR classifiers in terms of AUC.

Dataset	SMOTUNED		I-SMOTUNED		Cliff's δ
	AUC	Time (s)	AUC	Time (s)	
ant-1.3	0.742	37.2	0.751	22.7	Negligible
ant-1.4	0.631	147.0	0.639	69.1	Negligible
ant-1.5	0.730	89.0	0.732	34.7	Negligible
ant-1.6	0.759	106.3	0.759	60.4	Negligible
ant-1.7	0.736	437.0	0.733	194.2	Negligible
camel-1.0	0.668	392.0	0.673	103.8	Negligible
camel-1.2	0.577	226.7	0.579	71.1	Negligible
camel-1.4	0.660	509.4	0.660	240.8	Negligible
camel-1.6	0.628	565.0	0.630	214.4	Negligible
ivy-1.4	0.568	370.6	0.577	263.3	Negligible
ivy-2.0	0.726	239.5	0.725	94.1	Negligible
jedit-3.2	0.770	81.4	0.772	27.6	Negligible
jedit-4.0	0.683	109.5	0.669	34.3	Small
jedit-4.1	0.748	123.1	0.749	48.5	Negligible
jedit-4.2	0.747	121.5	0.746	91.7	Negligible
jedit-4.3	0.682	426.0	0.667	331.6	Negligible
log4j-1.0	0.738	44.2	0.740	23.1	Negligible
log4j-1.1	0.776	24.7	0.777	23.2	Negligible
lucene-2.0	0.680	26.9	0.679	16.3	Negligible
poi-2.0	0.677	126.1	0.671	76.8	Negligible
synapse-1.0	0.728	38.1	0.728	33.6	Negligible
synapse-1.1	0.701	56.0	0.704	40.8	Negligible
synapse-1.2	0.719	66.8	0.716	34.3	Negligible
velocity-1.6	0.691	45.5	0.694	38.3	Negligible
xalan-2.4	0.681	292.9	0.680	180.9	Negligible
xalan-2.5	0.592	104.5	0.593	71.4	Negligible
xalan-2.6	0.731	196.8	0.730	72.9	Negligible
xerces-1.2	0.524	325.5	0.505	147.1	Negligible
xerces-1.3	0.769	345.0	0.770	147.3	Negligible
xerces-init	0.635	30.7	0.633	17.7	Negligible
Median	0.696	122.3	0.699	70.1	

Table 9
Comparison between the execution time and the performances of SMOTUNED and I-SMOTUNED on the NB classifiers in terms of AUC.

Dataset	SMOTUNED		I-SMOTUNED		Cliff's δ
	AUC	Time (s)	AUC	Time (s)	
ant-1.3	0.722	121.7	0.711	36.4	Negligible
ant-1.4	0.526	106.7	0.517	71.8	Negligible
ant-1.5	0.671	56.7	0.666	37.5	Negligible
ant-1.6	0.729	277.5	0.726	102.3	Negligible
ant-1.7	0.732	975.7	0.735	400.2	Negligible
camel-1.0	0.734	121.0	0.740	70.6	Negligible
camel-1.2	0.556	161.9	0.557	68.5	Negligible
camel-1.4	0.639	1151.5	0.638	444.5	Negligible
camel-1.6	0.608	1537.8	0.609	518.8	Negligible
ivy-1.4	0.594	101.9	0.595	47.1	Negligible
ivy-2.0	0.679	261.2	0.669	166.3	Negligible
jedit-3.2	0.698	104.8	0.709	31.7	Negligible
jedit-4.0	0.699	181.3	0.692	157.5	Negligible
jedit-4.1	0.724	234.5	0.730	91.3	Small
jedit-4.2	0.733	221.0	0.738	184.4	Negligible
jedit-4.3	0.502	281.8	0.500	200.7	Negligible
log4j-1.0	0.727	67.5	0.719	54.8	Negligible
log4j-1.1	0.775	13.0	0.769	10.0	Negligible
lucene-2.0	0.678	131.1	0.684	69.7	Negligible
poi-2.0	0.655	268.2	0.618	51.1	Negligible
synapse-1.0	0.664	42.5	0.662	34.0	Negligible
synapse-1.1	0.656	85.2	0.654	25.2	Negligible
synapse-1.2	0.695	98.9	0.677	27.5	Negligible
velocity-1.6	0.607	146.0	0.610	72.4	Negligible
xalan-2.4	0.662	664.8	0.666	447.4	Negligible
xalan-2.5	0.562	750.5	0.561	244.9	Negligible
xalan-2.6	0.697	852.4	0.698	480.4	Negligible
xerces-1.2	0.509	109.8	0.516	98.8	Negligible
xerces-1.3	0.731	263.9	0.735	231.1	Negligible
xerces-init	0.616	83.4	0.590	71.0	Small
Median	0.675	154.0	0.668	72.1	

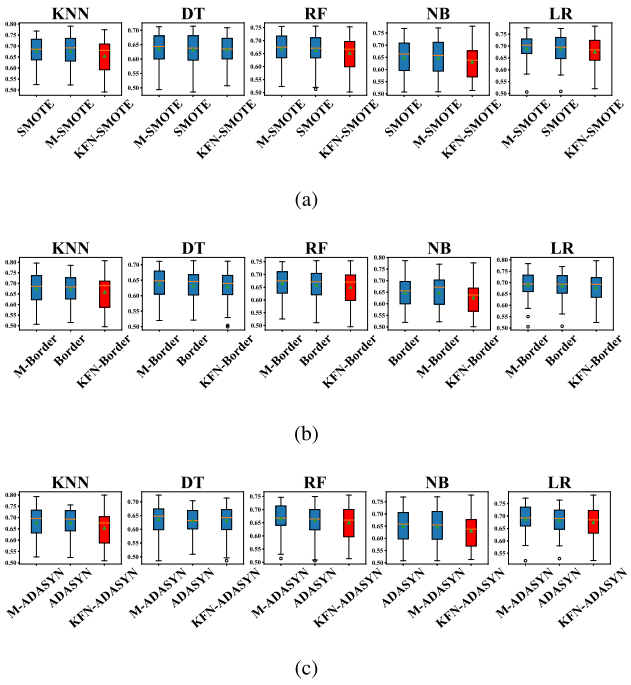


Fig. 7. The Scott–Knott ESD test ranking of the performances of SMOTE-based, M-SMOTE-based, and KFN-SMOTE-based techniques in terms of AUC on the five classifiers across 30 datasets (blue>red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

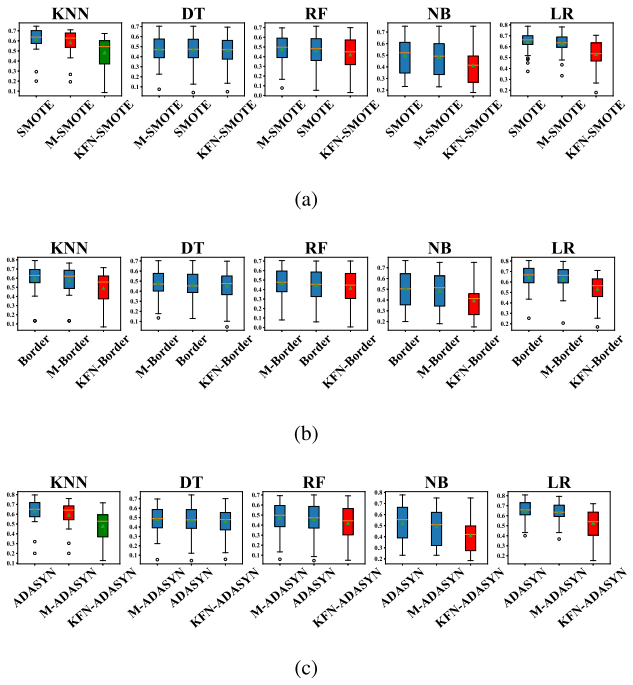


Fig. 8. The Scott–Knott ESD test ranking of the performances of SMOTE-based, M-SMOTE-based, and KFN-SMOTE-based techniques in terms of *pd* on the five classifiers across 30 datasets (blue>red>green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and LR classifiers, respectively. By not optimizing the hyperparameter of the distance metric, we can dramatically decrease the execution time of SMOTUNED while remaining its performance. It also confirms that different selections of distance metrics have little impact on the performances of SMOTE-based techniques.

> *To summarize, there is no significant difference among the performances of SMOTE-based techniques with different distance metrics based on the statistical analysis. Furthermore, the performance of the improved SMOTUNED is similar to that of the original SMOTUNED. Therefore, the answer to RQ1 is that the selection of the distance metric has little impact on the performances of SMOTE-based techniques.*

RQ2: Does the distance between the minority class instances used to generate synthetic instances impact the performances of SMOTE-based techniques?

As we have empirically proved that the selections of the distance metrics have little impact on the performances of SMOTE-based techniques, we adopt the Euclidean distance as the default distance metric in the following experiments.

To investigate the impact of the distance between the minority class instances on the performances of SMOTE-based techniques, we further modify SMOTE-based techniques and refer to them as SMOTE-based, KFN-SMOTE-based, and M-SMOTE-based techniques.

From Fig. 7(a), we can see that SMOTE and M-SMOTE perform similarly, and there is no significant difference between the performances of SMOTE and M-SMOTE in terms of AUC on all classifiers. Meanwhile, KFN-SMOTE performs poorly, and both SMOTE and M-SMOTE significantly outperform it on most classifiers. From Figs. 7(b) and 7(c), it can be seen that KFN-Borderline and KFN-ADASYN also perform poorly.

With respect to the *pd* values, we can see that SMOTE-based and M-SMOTE-based techniques both significantly outperform KFN-SMOTE-based techniques on the KNN, RF, NB, and LR classifiers from Fig. 8.

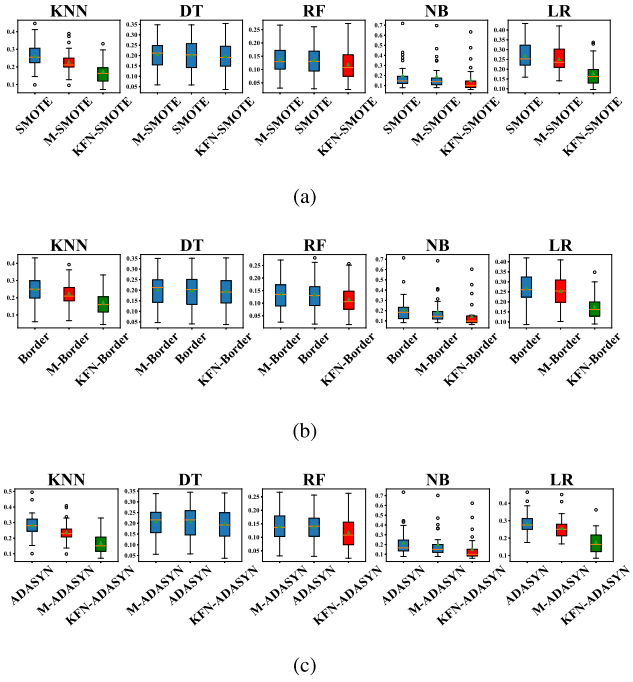
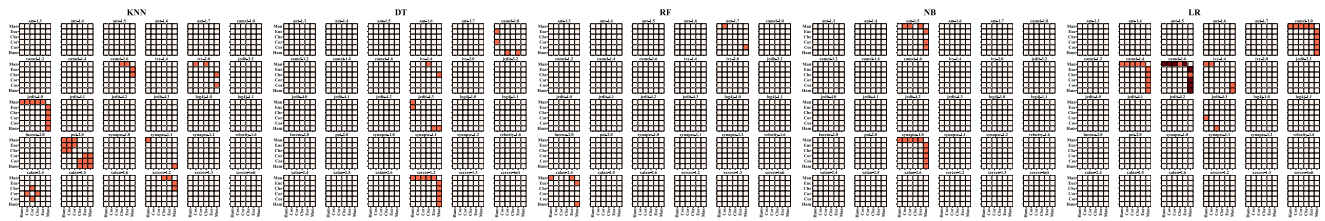
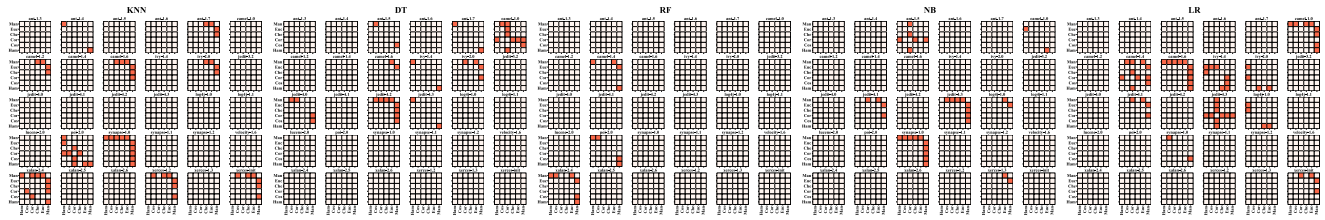


Fig. 9. The Scott–Knott ESD test ranking of the performances of SMOTE-based, M-SMOTE-based, and KFN-SMOTE-based techniques in terms of *pf* on the five classifiers across 30 datasets (blue>red>green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

On the DT classifier, although SMOTE-based and M-SMOTE-based techniques do not significantly outperform KFN-SMOTE-based techniques, the performance of KFN-SMOTE-based techniques is inferior in terms of *pd*.

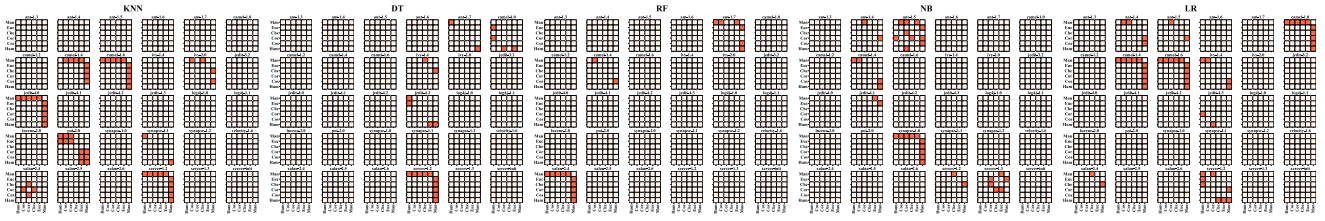


(a) balanced ratio of 0.4

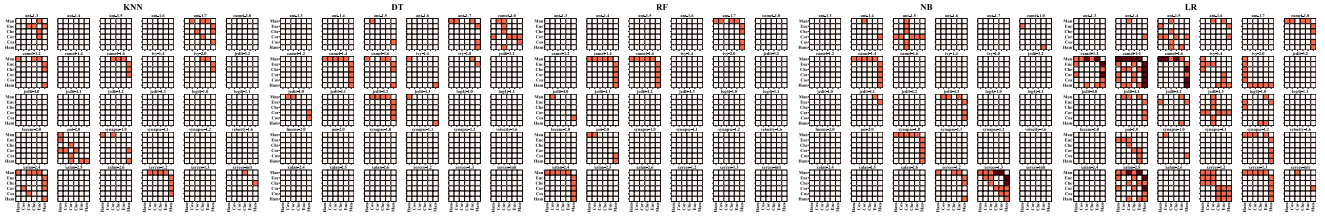


(b) balanced ratio of 0.6

Fig. 10. The heatmap of the effect sizes between the performance of SMOTE adopting different distance metrics in terms of AUC under different balanced ratios (The pink cell represents the effect size is negligible, the orange cell represents the effect size is small, and the red cell represents the effect size is larger than small). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) balanced ratio of 0.4



(b) balanced ratio of 0.6

Fig. 11. The heatmap of the effect sizes between the performance of SMOTE adopting different distance metrics in terms of *balance* under different balanced ratios (The pink cell represents the effect size is negligible, the orange cell represents the effect size is small, and the red cell represents the effect size is larger than small) . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Because of the positive relationship between pd and pf, We have also observed a similar trend in the pf values. From Fig. 9, it can be seen that the pf values of M-SMOTE-based and SMOTE-based techniques are significantly higher than those of KFN-SMOTE-based techniques.

To further demonstrate the relationship between the performances of these techniques and the distance, we present the distance between the instances used to generate synthetic instances and calculate the Spearman correlation coefficient (SP) [42]. According to [48], we denote SP as ρ and interpret SP as zero ($|\rho| = 0$), weak ($0.1 < |\rho| < 0.4$), moderate ($0.4 < |\rho| < 0.6$), strong ($0.6 < |\rho| < 0.9$), and perfect ($|\rho| = 1$).

We first present the median distance between the instances used to generate synthetic instances in each technique. From Table 10, it can be seen that the distance of KFN-SMOTE-based techniques is much greater than that of M-SMOTE-based techniques, and that of SMOTE-based techniques is the least. Then we present the Spearman correlation coefficient between the distance and the performances of SMOTE, M-SMOTE, and KFN-SMOTE across every single dataset. From Table 11, we can see that the AUC values of SMOTE are less correlated with the distance. The values of $|\rho|$ are all less than 0.4 on all classifiers in terms of AUC. Meanwhile, with respect to the pd and pf values, the values of $|\rho|$ are larger than 0.4 on the KNN, RF, NB, and LR classifiers, which shows a practical negative correlation between the distance and the pd and pf values on these classifiers. The larger the distance is, the lower pd and pf values SMOTE obtains. On the DT classifier, the pd and pf values are less correlated with the distance.

Table 10
The median distance between the instances used to generate synthetic instances in SMOTE-based, M-SMOTE-based, and KFN-SMOTE based techniques.

	SMOTE	M-SMOTE	KFN-SMOTE
Distance	0.599	1.201	1.933
	Borderline	M-Borderline	KFN-Borderline
Distance	0.577	1.178	1.921
	ADASYN	M-ADASYN	KFN-ADASYN
Distance	0.539	1.062	1.647

Table 11
The Spearman correlation coefficient between the distance and the performance of SMOTE.

KNN	AUC	*balance*	*pd*	*pf*
SP	−0.333	−0.567	−0.862	−0.983
DT	AUC	*balance*	*pd*	*pf*
SP	0.150	0.117	0.083	−0.167
RF	AUC	*balance*	*pd*	*pf*
SP	−0.383	−0.467	−0.496	−0.483
NB	AUC	*balance*	*pd*	*pf*
SP	−0.317	−0.450	−0.804	−0.817
LR	AUC	*balance*	*pd*	*pf*
SP	−0.350	−0.533	−0.771	−0.883

To summarize, the answer to RQ2 is that the distance between the instances used to generate synthetic instances indeed impacts the performances of SMOTE-based techniques. First, if the number of the noise instances is not beyond the noise-resistant ability of the classifiers, the overall performances of SMOTE-based techniques are not significantly impacted in terms of AUC and balance. Otherwise, the performance significantly degrades, which is consistent with Kim's conclusion [20]. Second, the pd and pf values are significantly impacted. The |ρ| values of the SP show that the larger the distance is, the lower the pd and pf values are. Our experimental results show that controlling the distance between the instances used to generate synthetic instances could alleviate the overgeneralization problem. Besides, we can adjust the pd and pf values of SMOTE-based techniques by controlling the distance between the instances used to generate synthetic instances to meet different requirements.

6. Discussion

6.1. Is the conclusion consistent under different balanced ratios?

In the above experiments, the default balanced ratio of SMOTE-based techniques is set to be 0.5. To make our conclusion more generalized, we choose different balanced ratios to re-conduct the experiment. If the experimental results are consistent, our conclusion could be more generalized. Specifically, we set the balanced ratio as 0.4 and 0.6 for SMOTE, respectively. From Fig. 10, we can see that there is only one combination of the dataset and the classifier on which the effect sizes between different distance metrics achieve larger than small, and 19 out of 150 combinations achieve small effect sizes under the balanced ratio of 0.4 in terms of AUC. The situation is similar under the balanced ratio of 0.6 as well as in terms of *balance* in 11. Therefore, our conclusion that the selection of the distance metric has little impact on the performances of SMOTE-based techniques is consistent under different balanced ratios.

Fig. 12 presents the ESD rankings of SMOTE, M-SMOTE, and KFN-SMOTE in terms of AUC under different balanced ratios. It can be seen that there is still no significant difference between the performances of SMOTE and M-SMOTE. Furthermore, it is notable that under the balanced ratio of 0.6, KFN-SMOTE performs better than SMOTE and M-SMOTE on the LR classifier. The poor performances of SMOTE and M-SMOTE are probably because of the overgeneralization. On the contrary, KFN-SMOTE produces more diverse synthetic instances and thus leads to better performance.

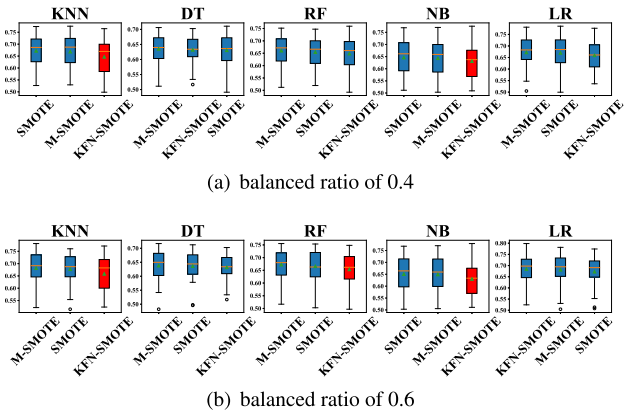


Fig. 12. The Scott–Knott ESD test ranking of the performance of SMOTE, M-SMOTE, and KFN-SMOTE in terms of AUC on the five classifiers across 30 datasets under different balanced ratio.

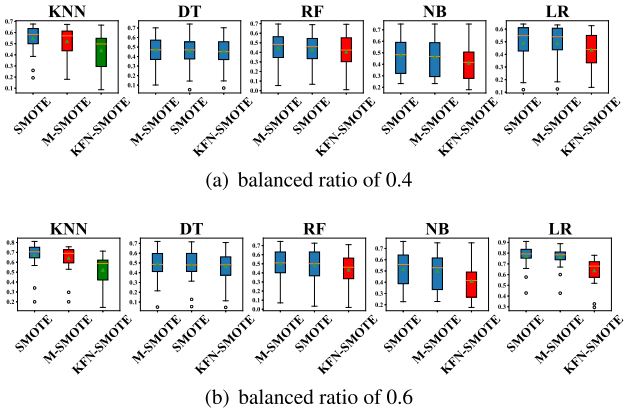


Fig. 13. The Scott–Knott ESD test ranking of the performance of SMOTE, M-SMOTE, and KFN-SMOTE in terms of *pd* on the five classifiers across 30 datasets under different balanced ratio (blue>red>green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

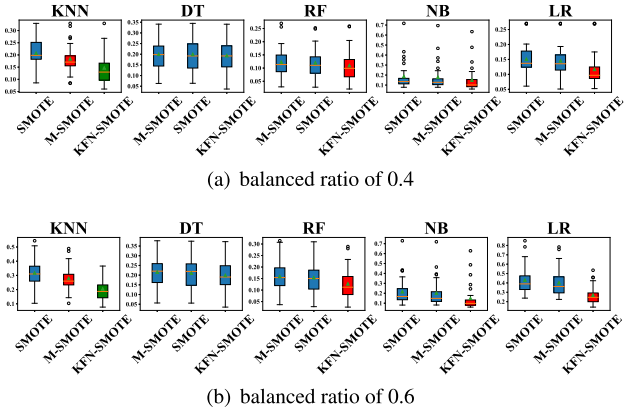


Fig. 14. The Scott–Knott ESD test ranking of the performance of SMOTE, M-SMOTE, and KFN-SMOTE in terms of *pf* on the five classifiers across 30 datasets under different balanced ratio (blue>red>green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

From Figs. 13 and 14, it can be seen that the performances of SMOTE, M-SMOTE, and KFN-SMOTE are still consistent with our conclusion that the larger the distance between the instances is, the lower the pd and pf values are.

6.2. Why does the selection of the distance metrics have little impact on the performances of SMOTE-based techniques?

Based on the above experimental results, we find that the selection of the distance metrics has little impact on the performances of SMOTE-based techniques. We analyze possible reasons for this result in the following.

As mentioned in Section 3.1, SMOTE first randomly selects a minority class instance. Then different distance metrics can be applied to obtain the K nearest neighbor instances of this selected instance. Notably, different distance metrics may obtain different K neighbor instances. However, because SMOTE is a blind oversampling technique that treats every instance equally, the quality of the K neighbor instances obtained by different distance metrics cannot be decided. Then, one instance is randomly selected from the K neighbor instances, and a synthetic instance is generated by randomly selecting an interpolation between the two instances. The randomness further makes the distance metrics less related to the performance of SMOTE. From another perspective, M-SMOTE is SMOTE. For a certain dataset, the K value of M-SMOTE is equal to the number of minority class instances minus one. When M-SMOTE is applied, two minority class instances are randomly selected. During this process, the distance metric is not involved. Based on the experimental results, the overall performances of SMOTE and M-SMOTE are similar. Therefore, the selection of the distance metrics has little impact on the performances of SMOTE-based techniques.

7. Threats to validity

As an empirical study, although we conduct our experiments on 30 datasets, we still cannot assert that the conclusion could generalize to other datasets. In addition, these datasets are measured only by the static code metric. Different conclusions may be drawn by using other types of metrics. However, these datasets and the static code metric were adopted by many previous studies, and their performances were quite satisfactory. Besides, a detailed description of our experiments is presented, which means it is easy for others to replicate our experiments using other datasets measured by different metrics. Six common distance metrics are adopted in this study. However, the effect of many other distance metrics deserves further investigation. This is left for future study. Five common classifiers are selected to build prediction models. There may exist other classifiers more sensitive to the selection of the distance metric. We intend to extend our experiments to more classifiers in the future. Some biases could be introduced into our results during the experiments. To minimize the bias, we adopt the 5-fold cross-validation and iterate the experiments 100 times. The 5-fold cross-validation is widely adopted [44,45] in the area of the class imbalance problem as well as SDP to produce training and testing data. There are other methods to produce training and testing data, such as using the older version of a project as training data and the most recent version as the testing data. When such methods are applied, the difference between the distributions of the two versions should be taken into consideration [49–51]. However, we focus on studying the class imbalance problem. The training and testing data produced by the 5-fold cross-validation belong to the same distribution, which does not need to handle the different distributions between the two versions. Therefore, the 5-fold cross-validation is adopted in this study. For the performance measures, one threshold-independent performance measure (i.e., AUC) and three threshold-dependent performance measures (i.e., $balance$, pd, and pf) are adopted to evaluate the performances of SMOTE-based techniques. These performance measures are common in SDP. However, there are many other performance measures, such as G-mean and precision. If other performance measures are adopted, different conclusions may be drawn. We plan to adopt more performance measures to generalize our conclusion in future work.

8. Conclusion and future work

Our previous study found that the overall performances of several oversampling techniques are similar, but the pd and pf values of these techniques vary significantly. The biggest difference among these techniques is the distance between the instances used to generate synthetic instances. In this study, we empirically evaluate the impact of the distance on the performances of SMOTE-based techniques with six common distance metrics (i.e., the Chebyshev, Manhattan, Euclidean, Cosine, Hamming, and Correlation distances) using five common classifiers (i.e., the KNN, DT, RF, NB, and LR classifiers) on 30 imbalanced datasets. We also evaluate the performances of SMOTE-based, M-SMOTE-based, and KFN-SMOTE-based techniques. The experimental results show that the performances of SMOTE-based techniques are not significantly impacted by the selection of the distance metric. Based on the results, we improve a recently proposed oversampling technique-SMOTUNED. The improved SMOTUNED obtains a similar performance to the original SMOTUNED, but the execution time of the improved SMOTUNED decreases dramatically. In addition, we also find that as long as the number of the synthesized noise instances is not beyond the noise-resistant ability of classifiers, the overall performances of SMOTE-based techniques are not significantly impacted by the distance between the minority class instances used to generate synthetic instances. Meanwhile, the pd and pf values of SMOTE-based techniques are significantly impacted by the distance. The practical values of the Spearman correlation coefficient show that the larger the distance is, the lower the pd and pf values are. Based on the experimental results, we could obtain different pd and pf values by selecting the instances at different distances to meet different requirements.

In our future work, we plan to improve the overall performances of the existing oversampling techniques by applying noise reduction techniques. Besides, we intend to extend our work to more datasets with different metrics using more classifiers. Then more performance measures will be adopted to generalize our conclusion.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 11208017) and the research funds of City University of Hong Kong (No. 7005028 and 7005217), and the Research Support Fund by Intel (No. 9220097), and funding supports from other industry partners (No. 9678149 , 9440227, 9229029, 9440180, and 9220103).

References

[1] S. Feng, J. Keung, X. Yu, Y. Xiao, M. Zhang, Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction, Inf. Softw. Technol. (2021) 106662.
[2] X. Chen, Y. Mu, K. Liu, Z. Cui, C. Ni, Revisiting heterogeneous defect prediction methods: How far are we? Inf. Softw. Technol. 130 (2021) 106441.
[3] K.E. Bennin, J.W. Keung, A. Monden, On the relative value of data resampling approaches for software defect prediction, Empir. Softw. Eng. 24 (2) (2019) 602–636.
[4] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[5] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: International Conference on Intelligent Computing, Springer, 2005, pp. 878–887.
[6] K.E. Bennin, J. Keung, P. Phannachitta, A. Monden, S. Mensah, Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction, IEEE Trans. Softw. Eng. 44 (6) (2017) 534–550.

[7] S.H. Khan, M. Hayat, M. Bennamoun, F.A. Sohel, R. Togneri, Cost-sensitive learning of deep feature representations from imbalanced data, IEEE Trans. Neural Netw. Learn. Syst. 29 (8) (2017) 3573–3587.

[8] D. Ryu, J.-I. Jang, J. Baik, A transfer cost-sensitive boosting approach for cross-project defect prediction, Softw. Qual. J. 25 (1) (2017) 235–272.

[9] A. Iranmehr, H. Masnadi-Shirazi, N. Vasconcelos, Cost-sensitive support vector machines, Neurocomputing 343 (2019) 50–64.

[10] X.-L. Zhang, D. Wang, A deep ensemble learning method for monaural speech separation, IEEE/ACM Transactions on Audio, Speech, and Language Processing 24 (5) (2016) 967–977.

[11] S. Liu, Y. Wang, J. Zhang, C. Chen, Y. Xiang, Addressing the class imbalance problem in twitter spam detection using ensemble learning, Comput. Secur. 69 (2017) 35–49.

[12] Z. Chen, T. Lin, X. Xia, H. Xu, S. Ding, A synthetic neighborhood generation based ensemble learning for the imbalanced data classification, Appl. Intell. 48 (8) (2018) 2441–2457.

[13] A. Agrawal, T. Menzies, Is better data better than better data miners?: on the benefits of tuning smote for defect prediction, in: Proceedings of the 40th International Conference on Software Engineering, ACM, 2018, pp. 1050–1061.

[14] M.M. Henein, D.M. Shawky, S.K. Abd-El-Hafiz, Clustering-based under-sampling for software defect prediction, in: ICSOFT, 2018, pp. 219–227.

[15] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1322–1328.

[16] D.M.J. Tax, One-class classification: Concept learning in the absence of counter-examples, 2002.

[17] G.Y. Wong, F.H. Leung, S.-H. Ling, A novel evolutionary preprocessing method based on over-sampling and under-sampling for imbalanced datasets, in: Iecon 2013-39th Annual Conference of the Ieee Industrial Electronics Society, IEEE, 2013, pp. 2354–2359.

[18] B. Turhan, T. Menzies, A.B. Bener, J. Di Stefano, On the relative value of cross-company and within-company data for defect prediction, Empir. Softw. Eng. 14 (5) (2009) 540–578.

[19] S. Feng, J. Keung, X. Yu, Y. Xiao, K.E. Bennin, M.A. Kabir, M. Zhang, COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction, Inf. Softw. Technol. (2020) 106432.

[20] S. Kim, H. Zhang, R. Wu, L. Gong, Dealing with noise in defect prediction, in: 2011 33rd International Conference on Software Engineering (ICSE), IEEE, 2011, pp. 481–490.

[21] J.S. Shirabad, T.J. Menzies, The PROMISE repository of software engineering databases, 24, School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

[22] F. Pérez, R. Lapeña, A.C. Marcén, C. Cetina, Topic modeling for feature location in software models: Studying both code generation and interpreted models, Inf. Softw. Technol. 140 (2021) 106676.

[23] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, K. Matsumoto, An empirical comparison of model validation techniques for defect prediction models, IEEE Trans. Softw. Eng. 43 (1) (2016) 1–18.

[24] D. Rankovic, N. Rankovic, M. Ivanovic, L. Lazic, Convergence rate of artificial neural networks for estimation in software development projects, Inf. Softw. Technol. (2021) 106627.

[25] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.

[26] C. Tantithamthavorn, A.E. Hassan, K. Matsumoto, The impact of class rebalancing techniques on the performance and interpretation of defect prediction models, IEEE Trans. Softw. Eng. PP (99) (2018).

[27] A. Tsymbal, The problem of concept drift: definitions and related work, 106, (2) Computer Science Department, Trinity College Dublin, 2004, p. 58.

[28] Q. Song, Y. Guo, M. Shepperd, A comprehensive investigation of the role of imbalanced learning for software defect prediction, IEEE Trans. Softw. Eng. PP (99) (2017) 1.

[29] X. Chen, D. Zhang, Y. Zhao, Z. Cui, C. Ni, Software defect number prediction: Unsupervised vs supervised methods, Inf. Softw. Technol. 106 (2019) 161–181.

[30] A. Yadav, S.K. Singh, J.S. Suri, Ranking of software developers based on expertise score for bug triaging, Inf. Softw. Technol. 112 (2019) 1–17.

[31] G.J. Székely, M.L. Rizzo, N.K. Bakirov, et al., Measuring and testing dependence by correlation of distances, Ann. Statist. 35 (6) (2007) 2769–2794.

[32] C. Coviello, S. Romano, G. Scanniello, A. Marchetto, A. Corazza, G. Antoniol, Adequate vs. inadequate test suite reduction approaches, Inf. Softw. Technol. 119 (2020) 106224.

[33] M. Jureczko, L. Madeyski, Towards identifying software project clusters with regard to defect prediction, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, 2010, pp. 1–10.

[34] S. Feng, J. Keung, X. Yu, Y. Xiao, K.E. Bennin, M.A. Kabir, M. Zhang, COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction, Inf. Softw. Technol. 129 (2021) 106432.

[35] H. Chen, X.-Y. Jing, Z. Li, D. Wu, Y. Peng, Z. Huang, An empirical study on heterogeneous defect prediction approaches, IEEE Trans. Softw. Eng. (2020).

[36] N. Li, M. Shepperd, Y. Guo, A systematic review of unsupervised learning techniques for software defect prediction, Inf. Softw. Technol. 122 (2020) 106287.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[38] M. Kondo, C.-P. Bezemer, Y. Kamei, A.E. Hassan, O. Mizuno, The impact of feature reduction techniques on defect prediction models, Empir. Softw. Eng. 24 (4) (2019) 1925–1963.

[39] V.K. Eluri, T.A. Mazzuchi, S. Sarkani, Predicting long-time contributors for GitHub projects using machine learning, Inf. Softw. Technol. 138 (2021) 106616.

[40] T. Schäfer, M.A. Schwarz, The meaningfulness of effect sizes in psychological research: Differences between sub-disciplines and the impact of potential biases, Front. Psychol. 10 (2019) 813.

[41] V.B. Kampenes, T. Dybå, J.E. Hannay, D.I. Sjøberg, A systematic review of effect size in software engineering experiments, Inf. Softw. Technol. 49 (11–12) (2007) 1073–1086.

[42] C. Spearman, The proof and measurement of association between two things, Appleton-Century-Crofts, 1961.

[43] P.A. Jaskowiak, R.J. Campello, I.G. Costa, Proximity measures for clustering gene expression microarray data: a validation methodology and a comparative analysis, IEEE/ACM Trans. Comput. Biol. Bioinform. 10 (4) (2013) 845–857.

[44] H. Wang, T.M. Khoshgoftaar, A. Napolitano, A comparative study of ensemble feature selection techniques for software defect prediction, in: 2010 Ninth International Conference on Machine Learning and Applications, IEEE, 2010, pp. 135–140.

[45] T. Zhu, Y. Lin, Y. Liu, Improving interpolation-based oversampling for imbalanced data learning, Knowl.-Based Syst. 187 (2020) 104826.

[46] A. Shanab, T. Khoshgoftaar, R. Wald, A. Napolitano, Impact of noise and data sampling on stability of feature ranking techniques for biological datasets, in: 2012 IEEE 13th International Conference on Information Reuse Integration (IRI), 2012, pp. 415–422, http://dx.doi.org/10.1109/IRI.2012.6303039.

[47] K. Zielinski, R. Laur, Stopping criteria for differential evolution in constrained single-objective optimization, in: Advances in Differential Evolution, Springer, 2008, pp. 111–138.

[48] C.P. Dancey, J. Reidy, Statistics Without Maths for Psychology, Pearson education, 2007.

[49] Z. Xu, S. Li, X. Luo, J. Liu, T. Zhang, Y. Tang, J. Xu, P. Yuan, J. Keung, TSTSS: A two-stage training subset selection framework for cross version defect prediction, J. Syst. Softw. 154 (2019) 59–78.

[50] Z. Xu, S. Li, Y. Tang, X. Luo, T. Zhang, J. Liu, J. Xu, Cross version defect prediction with representative data via sparse subset selection, in: 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC), IEEE, 2018, pp. 132–13211.

[51] Z. Xu, T. Zhang, J. Keung, M. Yan, X. Luo, X. Zhang, L. Xu, Y. Tang, Feature selection and embedding based cross project framework for identifying crashing fault residence, Inf. Softw. Technol. 131 (2021) 106452.