

Received September 2, 2019, accepted October 1, 2019, date of publication October 7, 2019, date of current version October 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945858

Tackling Class Imbalance Problem in Software Defect Prediction Through Cluster-Based Over-Sampling With Filtering

LINA GONG^{1,2,3}, SHUJUAN JIANG^{1,2}, (Member, IEEE), AND LI JIANG^{1,2}

¹School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

²Mine Digitization Engineering Research Center of Ministry of Education, China University of Mining and Technology, Xuzhou 221116, China

³Department of Information Science and Engineering, Zaozhuang University, Zaozhuang 277160, China

Corresponding author: Shujuan Jiang (shjjiang@cumt.edu.cn)


This work was supported by the Fundamental Research Funds for the Central Universities under Grant 2019BSCX28.

ABSTRACT In practice, Software Defect Prediction (SDP) models often suffer from highly imbalanced data, which makes classifiers difficult to identify defective instances. Recently, many techniques were proposed to tackle this problem, over-sampling technique is one of the most well-known methods to address class imbalance problem. This technique balances the number of defective and non-defective instances by generating new defective instances. However, these approaches would generate non-diverse synthetic instances, and many unnecessary noise instances at the same time. Motivated by this, we propose a Cluster-based Over-sampling with noise filtering (KMFOS) approach to tackle class imbalance problem in SDP. KMFOS firstly divides defective instances into K clusters, and new defective instances are generated by interpolation between instances of each two clusters. After this, these new defective instances would diversely spread in the space of defective dataset. Then, we extend this cluster-based over-sampling through the Closest List Noise Identification (CLNI) to clean the noise instances. We do extensive experiments on 24 projects to compare KMFOS with some over-sampling approaches such as SMOTE, Borderline-SMOTE, ADASYN, random over-sampling (ROS), K-means SMOTE, SMOTE + IPF, SMOTE + ENN and SMOTE + Tomek Links using five prediction classifiers. At the same time, we also compare KMFOS with other state-of-the-art class-imbalance methods including balancebaggingclassifier, RUSboostclassifier, InstanceHardnessThreshold and cost-sensitive methods. Experimental results indicate our KMFOS can obtain better *Recall* and *bal* values than other over-sampling methods and other compared class-imbalance methods. Hence, KMFOS is an efficient approach to generate balanced data for SDP and improves the performance of predicting models.

INDEX TERMS Software defect prediction, over-sampling, class imbalance, K-means, noise filtering.

I. INTRODUCTION

Software defect prediction (SDP) technologies can detect the largest number of defective modules by machine learning methods [1], [2], [30]. These machine learning methods may achieve good prediction performance when these training datasets are balanced [3], [4], [29]. However, there are more non-defective instances than defective instances in software projects, which leads class-imbalanced problem in SDP. Machine learning algorithms trained on these class-imbalanced datasets may be

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana .

biased towards non-defective instances and misclassify defective instances [5], [28]. Thus, the class imbalance problem is considered as one of the main factors affecting the performance of SDP, and more and more researchers have been working to solve this problem in SDP [6]–[8].

The prevalent methods tackling class imbalance problem are mainly sampling, cost-sensitive and ensemble learning methods. Among them, over-sampling is a classifier independent technique, which has been studied by many researchers. Chawla *et al.* [9] proposed synthetic minority over-sampling technique (SMOTE) using ROS as the core idea, whereby new artificial minority instances are generated to strike a balance in the number of minority and majority class.

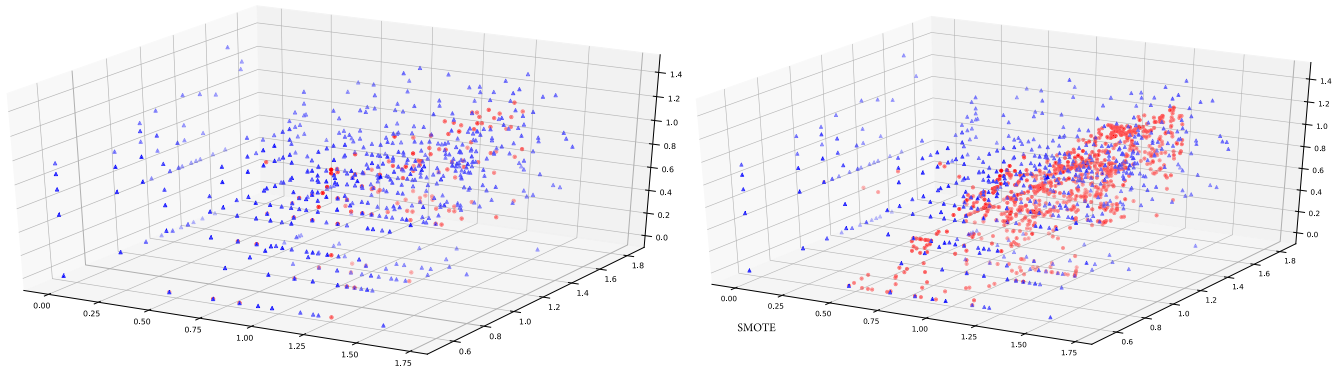


FIGURE 1. The distribution diagram of instances using SMOTE on PC4 project.

Also, Barua *et al.* [10] evaluated the positive effect of synthetic methods such as SMOTE and adaptive synthetic sampling (ADASYN) [11].

These over-sampling methods can improve classifiers to identify more instances in minority class, but at the same time they may lead to misclassify many instances in majority class. See figure 1, PC4 is the very class imbalance dataset from a real project in NASA group, and the distribution of defective instances is very sparsely. After using SMOTE, the areas including many minority instances get more new minority instances, on the contrary, sparse areas are still sparse. Moreover, the boundary of defective and non-defective class is ambiguity, which make classifiers difficult to classify.

The reasons are that: i) new synthetic instances may be non-diverse and distributed in small area, which leads classifiers easy to overfit. ii) the K-nearest neighbor instances considered would belong to majority class, which lead the boundary ambiguity problem between majority and minority classes, and generate noise instances.

For the above disadvantages, we propose a cluster-based over-sampling with filtering approach (KMFOS) that improve the recognition rate of defective instances and reduce the misclassified rate of non-defective instances for classifiers simultaneously. In general, our work consists mainly of the following contributions:

- To avoid the new defective instances non-diverse and distributed in small area, KMFOS generates new defective instances by interpolation between instances of each two clusters, and this makes these new defective instances diversely spread in the space of defective dataset.
- At the same time, in order to clean the unnecessary noise instances, KMFOS extends cluster-based over-sampling through CLNI to clean noise instances.
- To illustrate the effectiveness of our KMFOS approach, we conduct extensive experiments on 24 imbalanced datasets from NASA, AEEEM, ReLink and SOFTLAB groups in comparison to five over-sampling methods, three over-sampling with filtering methods using five classifiers, and four other class-imbalanced methods.

Experimental results indicate that our KMFOS significantly improves the *Recall* and *bal* in SDP.

The organization of this work is as follows. The related research works are discussed in section II. Section III describes our KMFOS method in detail. The experimental object, methodology and evaluation measures are presented in section IV. Section V presents the experimental results and analysis. The threats to our study are discussed in section VI. Lastly, the summary and future works are presented in section VII.

II. RELATED WORKS

Software defect prediction technology has been one of the most concerned research topics in software engineering since 1970s [31]–[33]. In recent years, with the rapid development of machine learning, various machine learning methods have been widely applied to improve the performance of SDP.

Furthermore, in software projects, defective modules are far less than non-defective modules, which is class imbalance problem. This problem seriously affects the performance of classifiers.

There have been some researches on class imbalance problem in SDP, which can be divided into sampling methods, ensemble methods and cost-sensitive learning methods. Sampling methods mainly adopt random over-sampling or under-sampling techniques to balance dataset. Ensemble learning methods combine multiple weak supervised classifiers to get a better strong supervised classifier. The commonly used ensemble learning methods are bagging, boosting and stacking. Cost-sensitive learning methods assign different misclassified cost to instances based on class label.

Over-sampling techniques is one of sampling methods by duplicating existing minority instances or generating new minority instances to balance the imbalanced data. Random over-sampling (ROS) is to randomly sample instances from minority class, and add these sampling instances into the dataset. But the repeated sampling often leads sever overfitting. SMOTE [9] generated new synthetic instances of minority class to balance imbalanced dataset. For each instance in minority class, the euclidean distance was used to find K

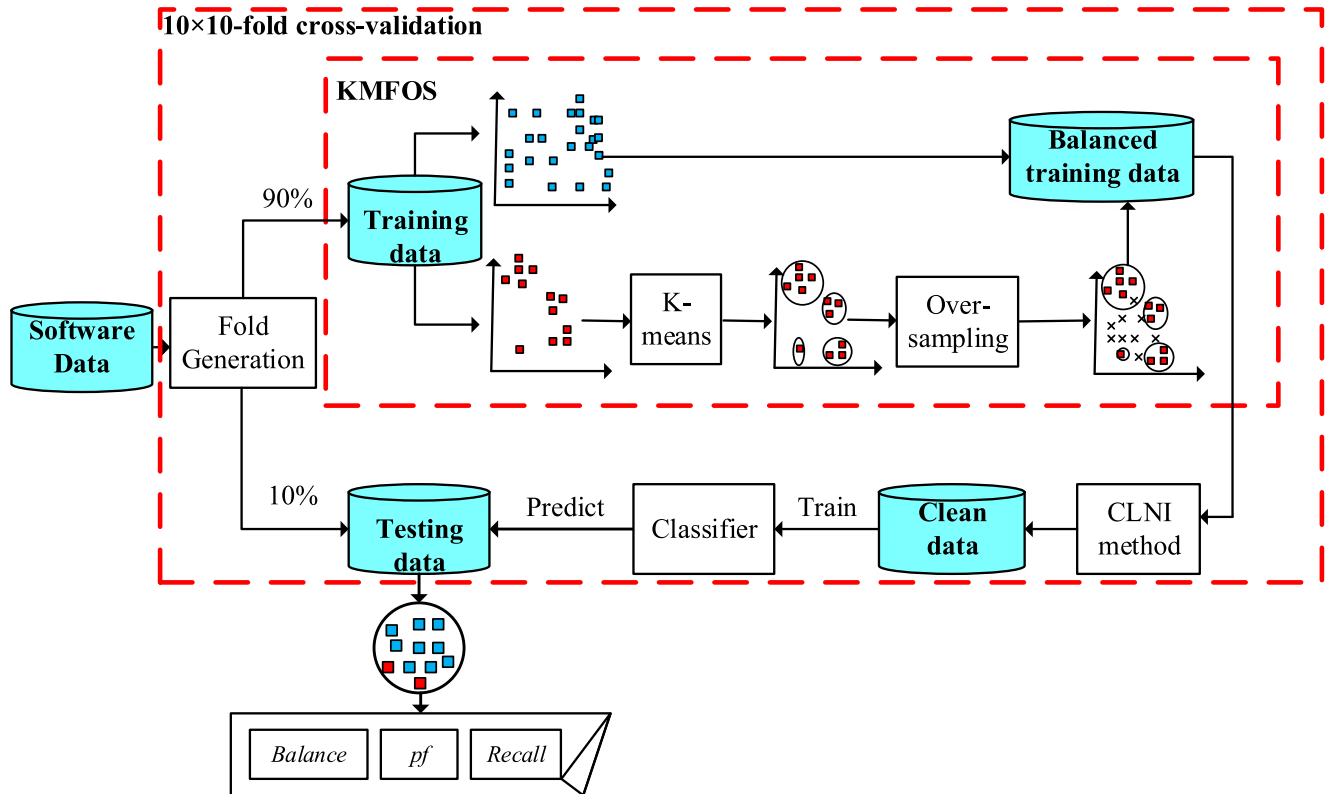


FIGURE 2. The framework of our method.

nearest neighbors, and the new synthetic instance was created along the line segment connecting any nearest neighbors.

However, SMOTE has some drawbacks related to the creation of new minority instances. One weakness was to randomly select one minority instance, which did not take the issues of within-class imbalance into account. This made minority instances non-diverse. Another weakness was that noise instances were further amplified when linearly interpolating one noise minority instance.

In order to eliminate these drawbacks, numerous extensions of SMOTE have been proposed, and they can be divided into two aspects:

- Improving the creation of new minority instances by considering specific parts of minority instances. Such as, Borderline-SMOTE [16] generated new minority instances focusing on the boundary between minority and majority class. ADASYN [11] generated new minority instances focusing on minority instances around the majority instances.
- Combing SMOTE with noise filtering techniques. Such as, SMOTE + ENN [19] used Edited Nearest Neighbor Rule (ENN) as a post-processing step after SMOTE. SMOTE + TL [42] applied Tomek Links TL) as a post-processing step after SMOTE. SMOTE + IPF [41] applied Iterative-Partitioning Filter (IPF) [43] a post-processing step after SMOTE.

Cluster-based SMOTE emphasized on within-class regions, which used K-means to cluster the minority

instances or all instances before using SMOTE. Such as, K-means SMOTE [40] firstly applied K-means to cluster all instances, and then filtered out clusters with a lower proportion of minority instances. Finally, SMOTE was applied in each cluster. Cluster-based SMOTE methods are different from the above SMOTE which didn't consider the class label. So motivated by this, we propose Cluster-based Over-sampling with filtering approach.

III. PROPOSED METHOD

A. OVERVIEW

Our KMFOS method introduces K-Means clustering algorithm and noise filtering into over-sampling technique to balance imbalanced datasets. Not only is the imbalance between classes considered, but also the imbalance between internal defective class is considered. This can avoid the non-diverse distribution of defective instances. Moreover, using noise filtering can avoid the influence of noise instances.

KMFOS includes three steps: clustering, over-sampling and filtering. In clustering step, the defective instances are clustered into k clusters by K-Means. In the over-sampling step, the new minority instances are generated based on two minority instances from two different clusters. In the filtering step, we apply CLNI [25] algorithm to detect and clean noisy instances. The framework is list in figure 2.

B. PHASE 1: CLUSTERING

In our method, the K-Means clustering algorithm is applied to cluster defective instances into k clusters. K-Means is a unsupervised clustering algorithm. It divide instances into k clusters according to the distance between instances. The points within the cluster should be connected as closely as possible, while the distance between the clusters should be as large as possible.

It is noted that K-Means is iterative method and the hyper-parameters k (the number of clusters) should be set firstly. The purpose of K-Means is to find the best label C_t of each instance to minimize the loss function, and then the center μ_i of each cluster can be calculated directly. Finding the appropriate k value is critical to the effectiveness of K-means suppression, because it affects how many clusters can be found in the over-sampling step. The loss function is defined as followed:

$$Loss = \sum_{i=1}^k \sum_{t=1}^n (x_t - \mu_i)^2 |_{(C_t = i)}. \quad (1)$$

where, n is the number of instances, C_t is the label of each instance in j^{th} iterative. μ_i is the center of each cluster.

C. PHASE 2: OVER-SAMPLING

After the clustering step, the over-sampling step determines how the new minority instances are be generated and how many new minority instances should be generated in each cluster. The aim of this step is to balance the distribution of instances between-class and within-class, which could generate more diverse and evenly distributed defective instances.

Let us suppose that the number of non-defective instances is N^- and the number of defective instances is N^+ . So the number of new defective instances is $N = N^- - N^+$. The defective instances in clustering step are divided into k clusters, and the number of instances in i^{th} cluster is n_i .

The core of our oversampling is to generate new defective instances between any two clusters. There are $\frac{k \times (k-1)}{2}$ combinations. For one combination of i and j cluster, we randomly sample one defective instance p from i cluster, and sample one defective instance q from j cluster. A new defective instance m is determined by the interpolating p and q : $m = \alpha \times p + \beta \times q$. Where, $\alpha = \frac{n_j}{n_i+n_j}$ and $\beta = \frac{n_i}{n_i+n_j}$. The number of new defective instances from this combination is calculate as $\frac{n_i+n_j}{(k-1) \times N^+} \times N$.

From the formula of new defective instance, we can see that if $n_i > n_j$, the coefficient β of instance q is bigger than the coefficient α of instance p . So the new instances could diversely spread in the space of defective space.

D. PHASE 3: NOISE FILTERING

As we see in figure 1, after generating new defective instances, there are more noise instances (including defective and non-defective instances) that influence the performance of classifiers. In reference [44], they found that the misclassification of classifiers often occurred the instances near the

boundaries of majority and minority class. In order to clean noisy instances, CLNI method is applied. Since not only are the non-defective noisy, but also some defective instances are noisy. We clean noise instances in both non-defective and defective class.

The core idea of CLNI is based on the number of non-defective neighborhood and defective neighborhood instances for each instance. Firstly, we applied euclidean distance to get the nearest neighbors. Then, in the kn neighborhood instances of one non-defective instance, if the number of defective neighborhood instances is bigger than that of non-defective neighborhood instances, this means that non-defective instance is a noise instance, and should be removed. On the contrary, in the kn neighborhood instances of one defective instance, if the number of non-defective neighborhood instances is bigger than that of defective neighborhood instances, this means that defective instance is a noise instance, and should be removed.

Though these three phases of clustering, over-sampling and filtering, the training dataset are constructed. The pseudo-code of KMFOS is shown in algorithm 1. And based on this training dataset, we train defect prediction classifiers to predict the defect-prone of testing data. In order to avoid the influence of classifiers, Naive Bayes (NB) [20], Random Forests (RF) [21], Support Vector Machine (SVM) [22], Logistic Regression (LR) [23] and Decision Tree (DT) [24] are applied in our work.

IV. EXPERIMENTAL SETUP

As mentioned above, the aim of our study is to improve synthetic methods to enhance the prediction performance of classifiers trained on class-imbalanced dataset. So we conduct experiments to answer the followed three questions:

RQ1: How is the effect of our cluster-based over-sampling for tackling class imbalance problem?

RQ2: How is the effect of noise filtering for tackling class imbalance problem?

RQ3: Does KMFOS method obtain better performance than other class imbalance methods?

For RQ1, we compare our cluster-based over-sampling method (in step 2) with SMOTE, Borderline-SMOTE, ADASYN, ROS, K-Means SMOTE and no over-sampling. For RQ2, we compare our KMFOS method with only using cluster-based over-sampling method (in step 2), SMOTE + IPF, SMOTE + ENN and SMOTE + TL. For RQ3, we compare our KMFOS with four other class imbalanced methods including balancebaggingclassifier, RUSboostclassifier, InstanceHardnessThreshold and cost-sensitive learning methods whose misclassified cost on the basis of the number of defective instances and non-defective instances. It is noted that five classifiers are trained after balanced training dataset obtained by these sampling methods.

Next, we will report the experimental objects, the methods under study, experiment design, model evaluation.

TABLE 1. Pseudo-code for cluster-based over-sampling with filtering.

Algorithm 1 Pseudo-code for Cluster-based Over-sampling with Filtering	
Inputs:	Training data D the number of clusters: k the number of nearest neighbors considered by CLNI: kn
Begin	
Step 1: Cluster the defective instances into k clusters	
1:	Divide the training data D into non-defective data D^- and defective data D^+
2:	Calculate the number of instances in D^+ : N^+
3:	Calculate the number of instances in D^- : N^-
4:	Calculate the number of new generated instances: $N = N^- - N^+$
5:	Using K-means algorithm to divide D into k clusters
6:	Calculate the number of instances in each cluster $n_i (i = 1, 2, \dots, k)$
Step 2: Over-sampling new defective instances	
7:	Select any two clusters to form a combination, and the number of combinations is $\frac{k \times (k-1)}{2}$
8:	for $t=1 - > \frac{k \times (k-1)}{2}$ do:
9:	Cluster i, j represent two clusters in t^{th} combination respectively
10:	n_i is the number of instances in cluster i
11:	n_j is the number of instances in cluster j
12:	Calculate the number of new instances in in t^{th} combination: $\frac{n_i + n_j}{(k-1) \times N^+} \times N$
13:	for $m = 1 - > \frac{n_i + n_j}{(k-1) \times N^+} \times N$ do:
14:	Randomly sample one instance p from cluster i
15:	Randomly sample one instance q from cluster j
16:	Generate new defective instance based on p and q : $Instance(m) = \frac{n_j}{n_i + n_j} * p + \frac{n_i}{n_i + n_j} * q$
17:	end for
18:	end for
19:	Combine all new generated instances D'
Step 3: Filter the noise instances	
20:	Calculate the number of instances in D' : N'
21:	for $i = 1 - > N'$ do:
22:	get kn nearest neighbors of i^{th} instance by euclidean distance
23:	Calculate the number n_0 of non-defective instances and the number n_1 of defective instances in kn neighbors
24:	if $n_1 > n_0$ and the label of i^{th} is non-defective:
25:	delete the i^{th} instances in D'
26:	end if
27:	if $n_0 > n_1$ and the label of i^{th} is defective:
28:	delete the i^{th} instances in D'
29:	end if
30:	end for

A. EXPERIMENTAL OBJECTS

In order to build and validate the effect of our KMFOS on imbalanced dataset for SDP, we do experiments on 24 projects from NASA, AEEEM, ReLink and SOFTLAB repositories that are often used in SDP researches. Table 2 lists the detail information of these experimental objects. The first to seventh columns report the repository, the testing project, the metric granularity, number of metrics, number of total instances and the ratio of defective instances.

From table 2, we can see: i) the development languages of these projects include Java, C and C++; ii) the metrics granularity covers function, class and file; iii) the number of instances varies greatly and the ratio of instances for most projects is well below 25%. So these projects can be used as the experimental objects to evaluate these class-imbalanced approaches.

B. THE METHODS UNDER STUDY

In order to validate the effect of KMFOS, we compare KMFOS with some state-of-the-art class imbalanced techniques, which will be explained below.

SMOTE [9] generated new synthetic instances of minority class to balance imbalanced dataset. For each instance in minority class, the euclidean distance was used to find K nearest neighbors, and the new synthetic instance was created along the line segment connecting any nearest neighbors.

ADASYN [11] was also by creating new synthetic instances to balance imbalanced data. For each instance in minority class, rather than using the same amount of synthetic instances in SMOTE, the amount of synthetic instances was determined automatically. If there were more instances of majority class around an instance of minority class, ADASYN would generate more synthetic instances for this instance. That was to say these synthetic instances were

TABLE 2. The experimental datasets.

Group	Project	Language	granularity	Number of metrics	Number of total instances	% of defective instances
NASA	CM1	C	function	37	327	12.84
	MW1				253	10.67
	PC1				705	8.65
	PC3				1077	12.44
	PC4				1458	12.21
	PC2				745	2.1
	JM1				7782	21.5
	PC5	1711		27.5		
	MC1	C++		39	1988	2.3
	MC2				125	35.2
	KC3	Java	40	194	18.6	
SOFTLAB	AR1	C		29	121	7.44
	AR3				63	12.70
	AR4				107	18.69
	AR5				36	22.22
	AR6				101	14.85
AEEEM	Equinox Framework (EQ)	Java		61	324	39.81
	Eclipse JDT core (JDT)				997	20.66
	Apache Lucence (LC)				691	9.26
	Mylyn (ML)				1862	13.16
	Eclipse PDE UI (PDE)				1497	13.96
ReLink	Apache HTTP Server		File	26	194	50.52
	OpenIntents Safe				56	39.29
	ZXing				399	29.57

mostly from those instances of minority class that were closer to the majority class.

Borderline-SMOTE [16] divided all instances in minority class into three categories including noise, danger and safe, and it only randomly selected instances in danger category to create new instances by SMOTE. Because the instances in danger category represented instances near the boundary, and these instances were easy to be misclassified. That was to say, Borderline-SMOTE only created new instances for these instances in minority class near the boundary, while SMOTE treated all instances in minority class in the same way.

ROS was to randomly sample instances from minority class, and added these sampling instances into the dataset. But the repeated sampling often led to severe overfitting. SMOTE + ENN, SMOTE + TL and SMOTE + IPF were the methods which also used the noise filtering. They applied SMOTE algorithm to balance the dataset, and then used different noise filtering techniques to clean the noise instances.

Noted that all over-sampling approaches generated as enough defective instances to meet the same number of instances in defective and non-defective class.

BalancedBaggingClassifier and RUSBoostClassifier were a combination of random under-sampling (RUS) and ensemble learning. RUSBoostClassifier combined boosting and RUS methods, while BalanceBaggingClassifier combined

bagging and RUS methods. InstancehardnessThreshold was to exploit instance hardness during the learning process to alleviate the effects of class overlap and instance hardness.

Cost-sensitive learning method assigned different misclassified cost to instances based on class label. If the defective instances were misclassified, the weight of these instances would be assigned bigger cost. In our experiment, the proportion of the cost in defective instances and non-defective is based on the number of defective instances and non-defective instances.

C. EXPERIMENT DESIGN

In order to prove the fairness of the experiment, we apply five classifiers which have been used for SDP. These classifiers include NB, RF, SVM, LR and DT. Noted that our experimental environment is Python3.6 and Scikit-learn (0.19.2). The Hyper-parameter configurations for the five classifiers are used in the default parameters in Scikit-learn. For the over-sampling methods, the parameters are set as follow:

- SMOTE: $kn \in \{5, 15, 20\}$
- Borderline-SMOTE1: $kn \in \{5, 15, 20\}$
- K-Means SMOTE: $kn \in \{5, 15, 20\}$ and $k \in \{3, 5, 20, 50\}$
- SMOTE + ENN: $kn \in \{5, 15, 20\}$
- SMOTE + TL: $kn \in \{5, 15, 20\}$

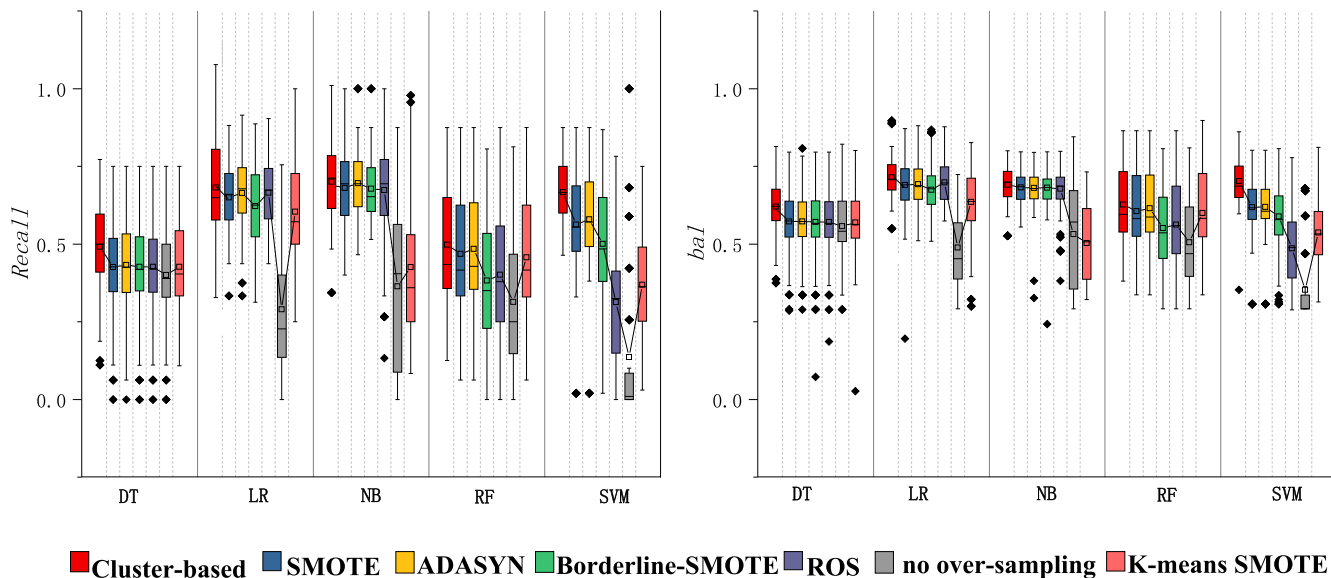


FIGURE 3. Box plot of compared sampling methods on five classifiers for Recall and bal over 24 imbalanced projects by 20 running.

- SMOTE + IPF: $kn \in \{5, 15, 20\}$
- KMFOs: $kn \in \{5, 15, 20\}$ and $k \in \{3, 5, 20, 50\}$

For each testing project, we apply 10×10 -fold cross-validation method to reduce the impact of randomness.

D. MODEL EVALUATION

For a good class imbalance method in SDP, the defective instances should be correctly classified, while the non-defective instances should not be misclassified. In order to evaluate the effect of these imbalanced methods, we use Recall and balance (bal) as the performance measures.

Recall is the ratio of the number of correctly predicted defective instances to the total number of defective instances. The bigger the Recall value is, the better the performance of method is. It is defined as

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

bal is the balance between Recall and pf. The bigger the bal value is, the better the performance of this method is. It is defined as:

$$bal = 1 - \frac{\sqrt{(1 - Recall)^2 + pf^2}}{\sqrt{2}} \tag{3}$$

where, pf is the ratio of the number of incorrectly predicted non-defective instances to the total non-defective instances.

In addition, to statistically evaluate these class imbalance methods, we apply the non-parametric Friedman test with the Nemenyi test at a confidence level of 95% by multiple runs on 24 projects. This statistical method has been widely used in SDP [14], [26]. Firstly, the Friedman test is used to decide whether compared methods have statistically significant difference. Then, the post-hoc Nemenyi test is used to calculate

the difference, if these methods have statistically significant difference.

In order to quantify the differences in the compared methods, Cliff’s Delta is applied to calculate the effect size. Cliff’s Delta is a non-parametric effect size measure, which includes four levels based on the value d . $|d| < 0.147$ (Negligible, N), $0.147 \leq |d| < 0.333$ (Small, S), $0.333 \leq |d| < 0.474$ (Medium, M) and $|d| \geq 0.474$ (Large, L).

V. EXPERIMENTAL RESULTS

A. RQ1: HOW IS THE EFFECT OF OUR CLUSTER-BASED OVER-SAMPLING FOR TACKLING CLASS IMBALANCE PROBLEM?

1) OVERALL RESULTS

The experimental results of compared over-sampling methods on five classifiers over 24 projects are presented in figure 3, which is a box plot. Box plot is a statistical plot used to show the dispersion of experimental results. The little square in each method represents the mean value in each method. A good sampling method should achieve high Recall and bal values. From figure 3, we can conclude that:

- For DT, LR and SVM classifiers, cluster-based method obtains higher Recall and bal values than SMOTE, ADASYN, Borderline-SMOTE, ROS, K-means SMOTE methods. For NB and RF classifiers, cluster-based method obtain the similar Recall and bal values.
- The mean values of Recall and bal by cluster-based method on five classifiers are highest.

These experimental results indicate that our Cluster-based method could improve the recognition rate of defective instances and reduce the misclassified rate of non-defective instances for compared classifiers. Possible reasons are that (i) SMOTE, ADASYN and Borderline-SMOTE all

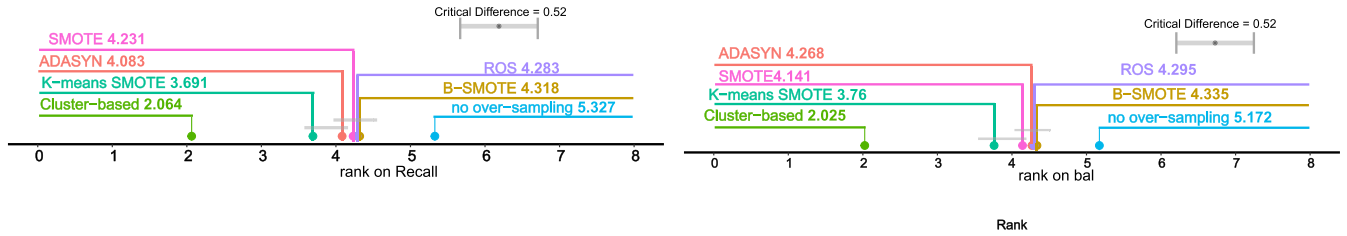


FIGURE 4. The ranks of compared sampling methods on DT classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

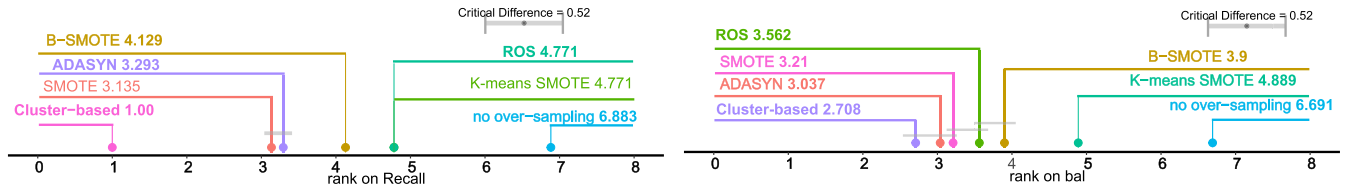


FIGURE 5. The ranks of compared sampling methods on LR classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

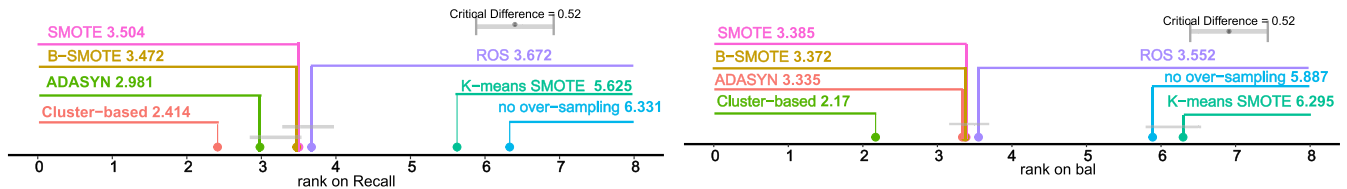


FIGURE 6. The ranks of compared sampling methods on NB classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

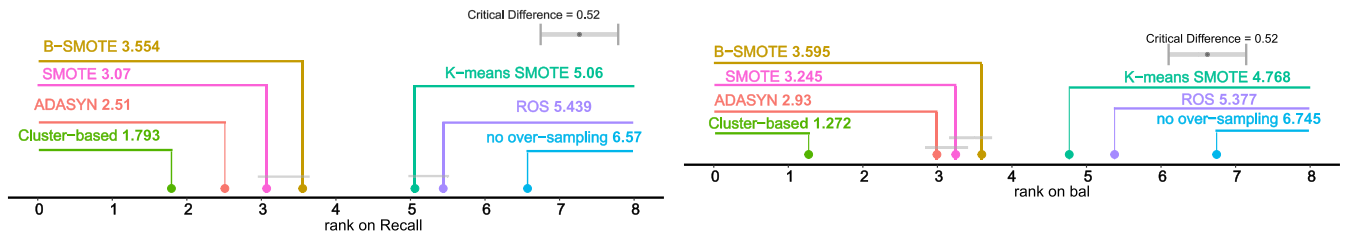


FIGURE 7. The ranks of compared sampling methods on SVM classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

use K-nearest neighbors to create new instances and new instances are distributed in small area. (ii) ROS method only simply copied defective instances. (iii) K-means SMOTE divided all instances to k clusters, which was regardless of the class label.

2) STATISTICAL ANALYSIS

In order to statistically investigate these results, the non-parametric Friedman test with post-hoc Nemenyi test at a confidence level of 95% is used to analyze compared sampling methods on five classifiers over the 24 imbalanced projects. Figures 4, 5, 6, 7 and 8 show the rank results in terms of *Recall* and *bal*.

From these figures, we can observe that:

- In terms of *Recall*, the performance of cluster-based method always ranks first and superior to the compared

over-sampling methods on DT, LR, SVM and NB classifiers. On RF classifier, the performance of cluster-based method ranks the same as ADASYN and superior to other five over-sampling methods. For the five classifiers, ROS and no over-sampling methods always rank the last two in terms of *Recall*.

- In terms of *bal*, the performance of cluster-based method ranks always the first and superior to other six compared sampling methods on SVM, NB and DT classifiers. On LR and RF classifier, the performance of cluster-based method ranks the same as ADASYN and superior to other five over-sampling methods.

Further, we apply Cliff’s Delta to calculate the effect size between cluster-based and compared over-sampling methods on five classifiers over 24 projects. The results are shown in table 3.

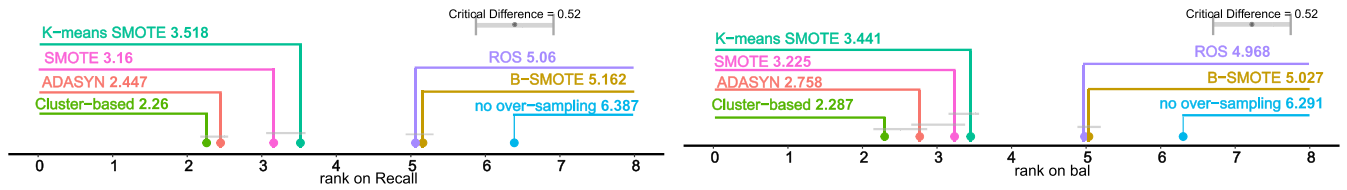


FIGURE 8. The ranks of compared sampling methods on RF classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

TABLE 3. The effective size results of compared method in terms of bal and Recall.

Compared methods	DT		LR		NB		RF		SVM	
	Recall	bal	Recall	bal	Recall	bal	Recall	bal	Recall	bal
Cluster-based Vs. SMOTE	0.29(S)	0.28(S)	0.215(S)	0.036(N)	0.129(N)	0.129(N)	0.09(N)	0.11(N)	0.392(M)	0.552(L)
Cluster-based Vs. ADASYN	0.257(S)	0.293(S)	0.281(N)	0.043(N)	0.126(N)	0.126(N)	0.03(N)	0.05(N)	0.312(S)	0.563(L)
Cluster-based Vs.B-SMOTE	0.286(S)	0.283(S)	0.103(N)	0.108(N)	0.135(N)	0.135(N)	0.34(M)	0.336(M)	0.509(L)	.61(L)
Cluster-based Vs.ROS	0.283(S)	0.284(S)	0.278(N)	0.088(N)	0.127(N)	0.127(N)	0.278(S)	0.283(S)	0.81(L)	0.838(L)
Cluster-based Vs. no over-sample	0.378(M)	0.362(M)	0.743(L)	0.766(L)	0.596(L)	0.596(L)	0.506(L)	0.506(L)	0.851(L)	0.926(L)
Cluster-based Vs. K-means SMOTE	0.296(S)	0.309(S)	0.102(N)	0.272(S)	0.646(L)	0.826(L)	0.126(N)	0.128(N)	0.815(L)	0.773(L)

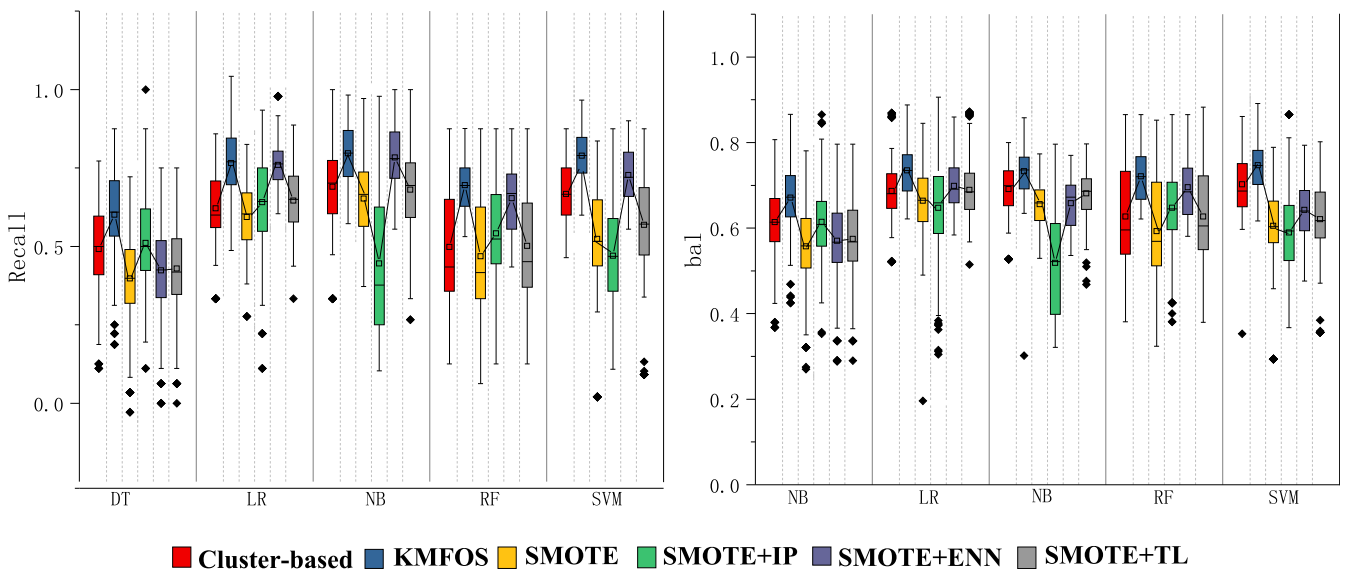


FIGURE 9. Box plot of compared sampling methods with filtering on five classifiers for Recall and bal over 24 imbalanced projects by 20 running.

So we can observe that (i) For SVM classifier, Cluster-based method can improve Larger effect size comparing other six over-sampling methods in terms of bal. That is to say using Cluster-based method can improve large performance of SVM classifier. (ii) For all classifiers, Cluster-based method can improve Larger or Medium effect size comparing no over-sampling method in terms of Recall and bal.

B. RQ2: HOW IS THE EFFECT OF NOISE FILTERING FOR TACKLING CLASS IMBALANCE PROBLEM?

1) OVERALL RESULTS

The experimental results of compared over-sample with filtering methods on five classifiers over 24 projects are presented in figure 9. The little square in each method

represents the mean value in each method. From figure 9, we can discover that: (i) For DT and RF classifiers, KMFOS method obtains higher Recall and bal values than compared over-sampling with filtering methods. That is to say, KMFOS method improves the recognition rate of defective instances, while it doesn't increase the error rate of non-defective instances. CLNI method is more suitable for SDP to tackling noise data. (ii) For LR, NB and RF classifiers, through SMOTE + ENN achieves similar Recall values to KMFOS values, the bal values are lower than KMFOS method. That is to say, SMOTE + ENN method gets good recognition rate of defective instances, while it increases the error rate of non-defective instances. (iii) For the five classifiers, KMFOS method can get better bal and Recall values than cluster-based method. SMOTE + ENN, SMOTE + TL and SMOTE + IPF

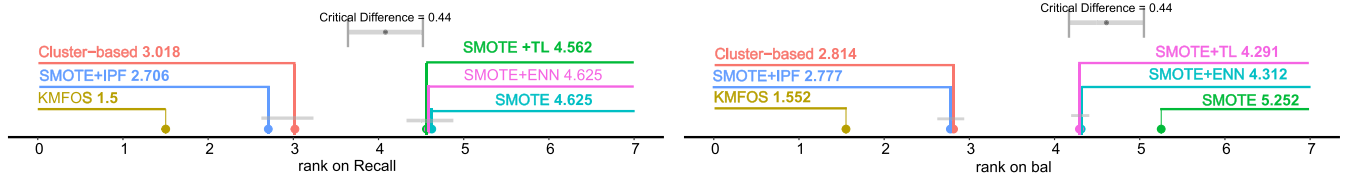


FIGURE 10. The ranks of compared sampling with filtering methods on DT classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

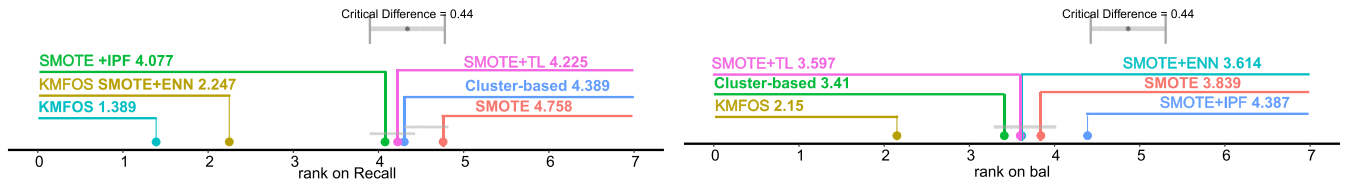


FIGURE 11. The ranks of compared sampling with filtering methods on LR classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

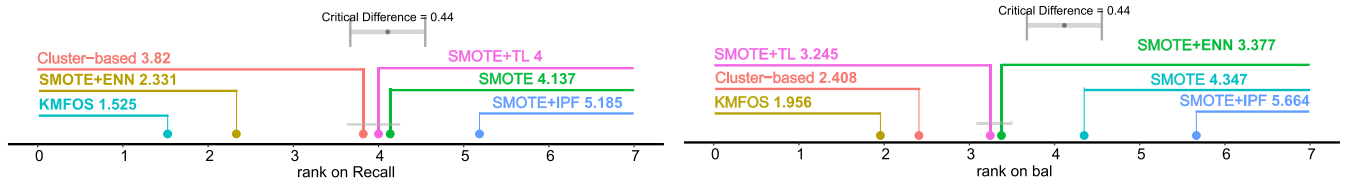


FIGURE 12. The ranks of compared sampling with filtering methods on NB classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

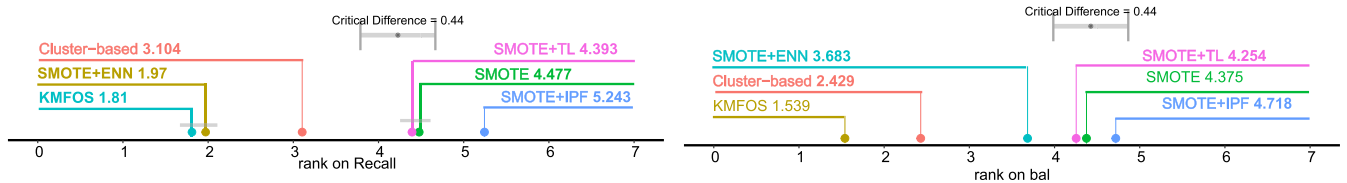


FIGURE 13. The ranks of compared sampling with filtering methods on SVM classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

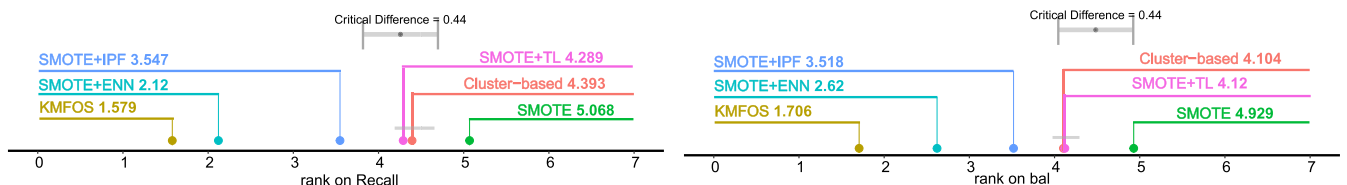


FIGURE 14. The ranks of compared sampling with filtering methods on RF classifier with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

methods can get better *bal* and *Recall* values than SMOTE method. That is to say, noise filtering is effect on tackling the class imbalance problem.

2) STATISTICAL ANALYSIS

In order to statistically investigate these results, the non-parametric Friedman test with post-hoc Nemenyi test at a confidence level of 95% is used to analyze compared sampling with filtering methods on five classifiers over the 24 imbalanced projects. Figures 10, 11, 12, 13 and 14 show the rank results in terms of *Recall* and *bal*. Further, we apply Cliff's

Delta to calculate the effect size between cluster-based and compared over-sampling methods on five classifiers over 24 projects. The results are shown in table 4.

From these figures and table 4, we can discover that (i) KMFOS always ranks the first on five classifiers in terms of *bal* and *recall*. For DT and RF classifiers, KMFOS improves Large on *bal* ad *Recall* compared with Cluster-based method. That is to say, noise filtering is useful for over-sampling method o DT and RF classifiers. (ii) For DT, LR and RF classifiers, SMOTE + ENN, SMOTE + TL and SMOTE + IPF are in the better rank

TABLE 4. The effective size results of compared over-sampling with filtering method in terms of *bal* and *Recall*.

Compared methods	DT		LR		NB		RF		SVM	
	<i>Recall</i>	<i>bal</i>	<i>Recall</i>	<i>bal</i>	<i>Recall</i>	<i>bal</i>	<i>Recall</i>	<i>bal</i>	<i>Recall</i>	<i>bal</i>
KMFOS Vs. Cluster-based	0.453(M)	0.4(M)	0.475(L)	0.227(S)	0.237(S)	0.136(S)	0.599(L)	0.491(L)	0.356(M)	0.173(S)
KMFOS Vs. SMOTE+IPF	0.72(L)	0.72(L)	0.309(S)	0.334(M)	0.666(L)	0.781(L)	0.565(L)	0.464(M)	0.814(L)	0.748(L)
KMFOS Vs. SMOTE+ENN	0.628(L)	0.613(L)	0.278(S)	0.154(S)	0.238(S)	0.341(M)	0.219(S)	0.232(S)	0.148(N)	.596(L)
KMFOS Vs. SMOTE+TL	0.613(L)	0.593(L)	0.398(M)	0.232(S)	0.302(S)	0.164(S)	0.603(L)	0.509(L)	0.635(L)	0.644(L)
KMFOS Vs. SMOTE	0.627(L)	0.606(L)	0.489(M)	0.287(S)	0.311(S)	0.438(M)	0.672(L)	0.581(L)	0.641(L)	0.653(L)

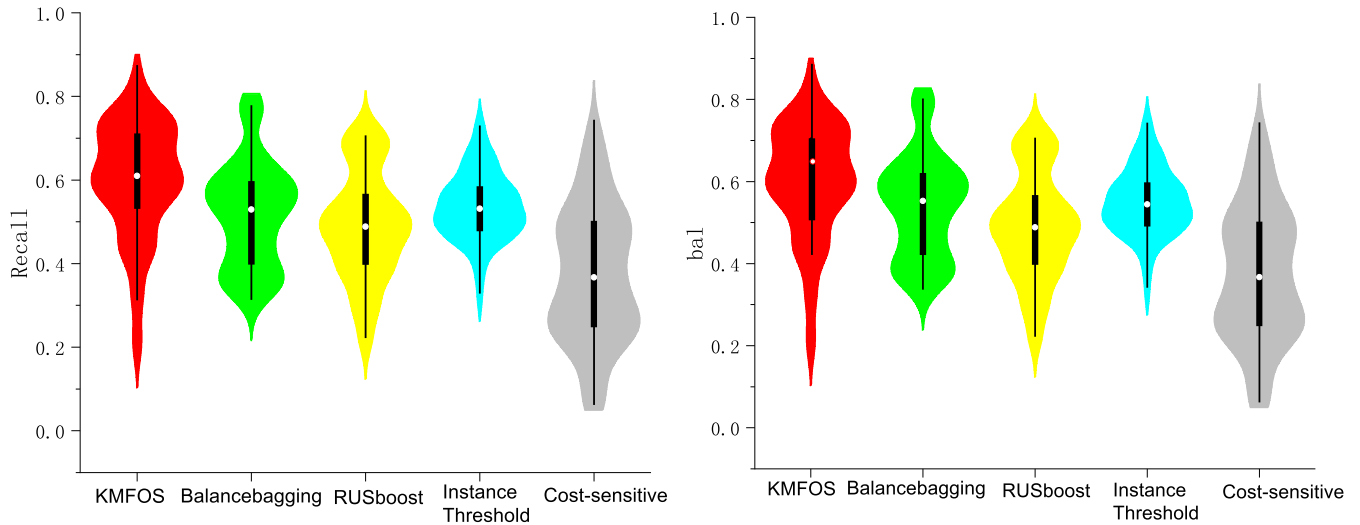


FIGURE 15. The violin plot of the compared methods in terms of *Recall* and *bal*.

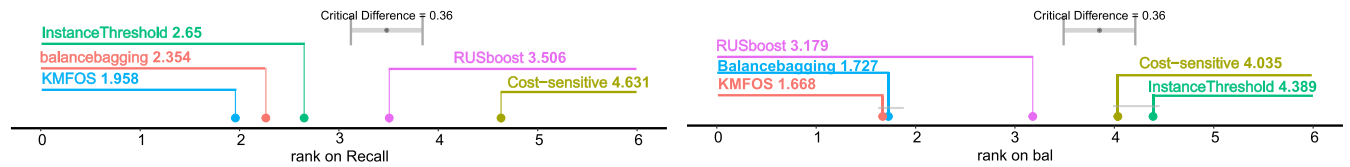


FIGURE 16. The ranks of other state-of-the-art class imbalance methods with post-hoc Nemenyi test. Methods connected by gray lines are not significantly different.

than SMOTE method. (iii) For DT, RF and SVM classifiers, KMFOS improves Large effect size on *bal* and *Recall* compared with SMOTE + IPF, SMOTE + ENN and SMOTE + TL methods.

So we can observe that using noise filtering can improve the performance of over-sampling methods.

C. RQ3: DOES OUR KMFOS OBTAIN BETTER PERFORMANCE THAN OTHER STATE-OF-THE-ART CLASS IMBALANCE METHODS?

In order to answer the RQ3, we compare our KMFOS with four class imbalance methods including balancebaggingclassifier [45], RUSboostclassifier [46], InstanceHardnessThreshold [47] and cost-sensitive methods with DT as the basic classifier, and report the violin plots of the results on 24 projects with 20 running for each compared methods in figure 15. The violin plot combines the characteristics of box plot and density of results. In addition, we apply the non-parametric Friedman test with post-hoc Nemenyi test to

statistically analyze the compared methods. Figure 16 lists the rank results in terms of *Recall* and *bal*.

From these figures, we can observe that (i) our KMFOS can get better *Recall* and *bal* than four compared methods, and always ranks the first level. (ii) The values of *Recall* and *bal* centrally distributed in KMFOS are higher than four compared methods.

In general, our KMFOS method could improve the predicting performance of classifiers by balancing the dataset in term of *bal* and *Recall*. That is to say, KMFOS method could generate good new instances to identify more defective instances without misclassifying non-defective instances.

VI. THREATS TO VALIDITY

In this section, we describe the threats to validity of our study in construct validity, internal validity and external validity.

A. THREATS TO CONSTRUCT VALIDITY

These datasets used in our study were collected from binaries rather than source code files, whose defect matching

was relied on SZZ algorithm. In SZZ algorithm, the discovered defective modules may be incomplete. However, these projects were widely used in SDP research, we also have the hypothesis that defect matching criteria in SZZ are the best extent possible.

B. THREATS TO INTERNAL VALIDITY

In our study, we use *Recall* and *bal* to present the prediction performance. Other measures, such as *G-Means* and *MCC* are not presented.

Also, we executed the compared methods with the default setting which may lead to possible bias of results.

C. THREATS TO EXTERNAL VALIDITY

The 24 experimental objects are from four groups which have been widely used in SDP. But these finds would not be generalizable to commercial software projects. In the future, we will employ our KMFOS to some commercial projects to evaluate the performance.

VII. CONCLUSION

In real software development, defect datasets have more non-defective instances than defective instances, which hinders classifiers performance. Many sampling methods have been used to reduce the impact of imbalanced datasets. These methods used resampling or generated new instances to balance the imbalanced datasets. So these sampling methods would have randomness and new synthetic instances were non-diverse and distributed in a small area. At the same time, they didn't consider noise instances. Although they may improve classifiers to identify more defective instances, they also may make classifiers misclassify more non-defective instances. Exploiting these challenges, we present a KMFOS method to generate new instances diversely spread in the space of defective space. KMFOS firstly applies K-means method to divide defective instances into *K* clusters. Then, it generates new instances by interpolation between instances of each two clusters. Lastly, it clean noise instances by CLNI filtering.

We conduct extensive experiments to evaluate KMFOS by comparing to (i) five over-sampling methods (SMOTE, ADASYN, Borderline-SMOTE, ROS, and K-means SMOTE) through five classifiers including RF, SVM, NB, LR and DT (ii) three over-sampling with filtering methods (SMOTE + IPF, SMOTE + ENN and SMOTE + TL) (iii) four other class imbalanced methods (balancebagging-classifier, RUSboostclassifier, InstanceHardnessThreshold and cost-sensitive methods) on 24 software projects. The experimental results make clear that our KMFOS is superior to compared methods.

In the further works, we will employ KMFOS method to other software defect prediction models and some commercial projects to evaluate the performance. We also compare it to some other class imbalanced methods.

ACKNOWLEDGMENT

The authors are grateful to editors and reviewers for their valuable comments and useful suggestions. Special thanks to all the individuals who participated and contributed to improve the quality and readability of this paper.

REFERENCES

- [1] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007.
- [2] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Trans. Rel.*, vol. 62, no. 2, pp. 434–443, Jun. 2013.
- [3] G. M. Weiss and F. Provost, *The Effect of Class Distribution on Classifier Learning: An Empirical Study*. Camden, NJ, USA: Rutgers Univ., 2001.
- [4] K. Yoon and S. Kwek, *A Data Reduction Approach for Resolving the Imbalanced Data Issue in Functional Genomics*. Berlin, Germany: Springer-Verlag, 2007.
- [5] F. Provost, "Machine learning from imbalanced data sets 101," in *Proc. AAAI Workshop Imbalanced Datasets*, Jul. 2000, pp. 1–3.
- [6] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1806–1817, Nov. 2012.
- [7] Y. Kamei, A. Monden, S. Matsumoto, T. Kakimoto, and K.-I. Matsumoto, "The effects of over and under sampling on fault-prone module detection," in *Proc. 1st Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Sep. 2007, pp. 196–204.
- [8] L. Pelayo and S. Dick, "Applying novel resampling strategies to software defect prediction," in *Proc. Annu. Meeting North Amer. Fuzzy Inf. Process. Soc.*, San Diego, CA, USA, Jun. 2007, pp. 69–72.
- [9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.
- [10] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE—Majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 405–425, Feb. 2014.
- [11] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Hong Kong, Jun. 2008, pp. 1322–1328.
- [12] G. Blanchard and R. Loubere, "High-Order Conservative Remapping with a Posteriori MOOD stabilization on polygonal meshes," *J. Comput. Phys.*, 2016, pp. 1–43.
- [13] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sep. 2013.
- [14] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empir. Softw. Eng.*, vol. 17, nos. 4–5, pp. 531–577, 2011.
- [15] R. Wu, H. Zhang, S. Kim, and S.-C. Cheung, "ReLink: Recovering links between bugs and changes," in *Proc. 19th ACM SIGSOFT Symp. 13th Eur. Conf. Found. Softw. Eng.*, Szeged, Hungary, Sep. 2011, pp. 15–25.
- [16] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput.*, Berlin, Germany: Springer, 2005, pp. 878–887.
- [17] X. Zhang, Q. Song, G. Wang, K. Zhang, L. He, and X. Jia, "A dissimilarity-based imbalance data classification algorithm," *Appl. Intell.*, vol. 42, no. 3, pp. 544–565, Apr. 2015.
- [18] S. Herbold, A. Trautsch, and J. Grabowski, "Global vs. Local models for cross-project defect prediction," *Empirical Softw. Eng.*, vol. 22, no. 4, pp. 1866–1902, Aug. 2017.
- [19] H. Li, P. Zou, X. Wang, and R. Z. Xia, "A new combination sampling method for imbalanced data," in *Proc. Chin. Intell. Automat. Conf.*, in Lecture Notes in Electrical Engineering, vol. 256, 2013, pp. 547–554.
- [20] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Softw. Eng.*, vol. 14, no. 5, pp. 540–578, 2009.
- [21] L. Hribar and D. Duka, "Software component quality prediction using KNN and Fuzzy logic," *Inf. Softw. Technol.*, vol. 58, no. 2, pp. 388–402, 2010.

- [22] L. Miao, M. Liu, and D. Zhang, "Cost-sensitive feature selection with application in software defect prediction," in *Proc. Int. Conf. Pattern Recognit.*, Tsukuba, Japan, Nov. 2012, pp. 967–970.
- [23] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Inf. Softw. Technol.*, vol. 27, pp. 504–518, Feb. 2015.
- [24] J. Wang, B. Shen, and Y. Chen, "Compressed C4.5 models for software defect prediction," in *Proc. 12th Int. Conf. Qual. Softw.* Xi'an, Shaanxi, China, Aug. 2012, pp. 13–16.
- [25] S. Kim, H. Zhang, and R. L. Wu, and Gong, "Dealing with noise in defect prediction," in *Proc. 33rd Int. Conf. Softw. Eng.* Honolulu, HI, USA, May 2011, pp. 481–490.
- [26] S. Herbold, A. Trautsch, and J. Grabowski, "A comparative study to benchmark cross-project defect prediction approaches," *IEEE Trans. Softw. Eng.*, vol. 44, no. 9, pp. 811–833, Sep. 2018.
- [27] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "MAHAKIL: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction," *IEEE Trans. Softw. Eng.*, vol. 44, no. 6, pp. 534–550, Jun. 2018.
- [28] P. Cholmyong, W. T. Tian, and S. X. Hong, "An empirical study on software defect prediction using over-sampling by SMOTE," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 6, pp. 811–830, Jun. 2018.
- [29] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," *IEEE Trans. Softw. Eng.*, to be published.
- [30] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "The impact of automated parameter optimization on defect prediction models," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 683–711, Jul. 2019.
- [31] J. Nam, P. J. Sinno, and S. Kim, "Transfer defect learning," in *Proc. 35th Int. Conf. Softw. Eng.*, San Francisco, CA, USA, May 2013, pp. 382–391.
- [32] X.-Y. Jing, S. Ying, Z.-W. Zhang, S.-S. Wu, and J. Liu, "Dictionary learning based software defect prediction," in *Proc. IEEE Int. Conf. Softw. Eng.*, Hyderabad, India, Jun. 2014, pp. 414–423.
- [33] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect prediction from static code features: Current results, limitations, new approaches," *Automated Softw. Eng.*, vol. 17, no. 2, pp. 375–407, Dec. 2010.
- [34] R. A. Coelho, F. dos R. N. Guimarães, and A. A. A. Ahmed, "Applying swarm ensemble clustering technique for fault prediction using software metrics," in *Proc. Mach. Learn. Appl.* Detroit, MI, USA, Dec. 2014, pp. 356–361.
- [35] M. Park and E. Hong, "Software fault prediction model using clustering algorithms determining the number of clusters automatically," *Int. J. Softw. Eng. Appl.*, vol. 8, no. 7, pp. 199–204, 2014.
- [36] P. S. Bishnu and V. Bhattacharjee, "Software fault prediction using quad tree-based K-means clustering algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 1146–1150, Jun. 2012.
- [37] T. Wang, Z. Zhang, X. Jing, and L. Zhang, "Multiple kernel ensemble learning for software defect prediction," *Automated Softw. Eng.*, vol. 23, no. 4, pp. 569–590, Dec. 2016.
- [38] H. Wan, G. Wu, C. Ming, H. Qing, W. Rui, and Y. Mengting, "Software defect prediction using dictionary learning," in *Proc. Int. Conf. Softw. Eng. Knowl. Eng. (SEKE)*, Pittsburgh, PA, USA, Jul. 2017, pp. 335–340.
- [39] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
- [40] D. Georgios, B. Fernando, and L. Felix, "Oversampling for imbalanced learning based on K-means and SMOTE," *Inf. Sci.*, vol. 465, pp. 1–20, Oct. 2018.
- [41] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Inf. Sci.*, vol. 291, pp. 184–203, Jan. 2015.
- [42] G. Batista, B. Bazzan, and M. Monard, "Balancing training data for automated annotation of keywords: A case study," in *Proc. WOB*, 2003, pp. 10–18.
- [43] T. M. Khoshgoftaar and P. Reboours, "Improving software quality prediction by noise filtering techniques," *J. Comput. Sci. Technol.*, vol. 22, no. 3, pp. 387–396, May 2007.
- [44] K. Napierala, J. Stefanowski, and S. Wilk, "Learning from imbalanced data in presence of noisy and borderline examples," in *Rough Sets Current Trends Computing (Lecture Notes in Computer Science)*, vol. 6086. Berlin, Germany: Springer, 2010, pp. 158–167.
- [45] G. Louppe and P. Geurts, "Ensembles on random patches," in *Machine Learning and Knowledge Discovery in Databases*. Berlin, Germany: Springer, 2012, pp. 346–361.
- [46] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010.
- [47] M. R. Smith, T. Martinez, and C. Giraud-Carrier, "An instance level analysis of data complexity," *Mach. Learn.*, vol. 95, no. 2, pp. 225–256, 2014.



LINA GONG is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, China University of Mining and Technology. Her research interests include software analysis and testing, and machine learning.



SHUJUAN JIANG received the Ph.D. degree from Southeast University, in 2006. She was a Visiting Scholar with the Georgia Institute of Technology, from September 2008 to April 2009. She is currently a Professor and a Supervisor of the Ph.D. degree with the School of Computer Science and Technology, China University of Mining and Technology. Her research interests include compilation techniques and software engineering.



LI JIANG is currently pursuing the M.S. degree with the School of Computer Science and Technology, China University of Mining and Technology. Her research interests include software analysis and testing, and machine learning.

...