Class Imbalance Data-Generation for Software Defect Prediction

Zheng Li, Xingyao Zhang, Junxia Guo, Ying Shang
Department of Computer Science
Beijing University of Chemical Technology
Beijing, 100029,China
shangy@mail.buct.edu.cn

***Abstract*—The imbalanced nature of class in software defect data, which including intra-class imbalance and inter-classes imbalance, increases the difficulty of learning an effective defect prediction model. Most of sampling and example generation approaches just focused on inter-class imbalanced defect data, and they are not effective to handle the issue of intra-class imbalance. This paper proposed a distribution based data generation approach for software defect prediction to deal with inter-class and intra-class imbalanced data simultaneously. First, the classified sub-regions are clustered according to the distribution in the sample feature space. Second, the data are generated by corresponding strategies according to different distribution in sub-regions, where the inter-class balance is achieved by increasing the number of defective samples, and the intra-class balance is achieved by generating different density of data in different sub-regions. Experiment results show that the proposed method can reduce the impact of data imbalance on defect prediction and improve the accuracy of software defect prediction model effectively by generating inter-class and intra-class balanced defects data.**

Keywords—software defect prediction; imbalanced data; data generation; machine learning

I. INTRODUCTION

Software defect prediction detect defect-prone software modules by mining the historical data with software metrics [1]. A wide range of machine learning methods have been studied to predict defects in software modules, i.e., to generate prediction model based on existing data training, and then to predict which modules are likely to contain defects. In the existing industrial software environment, the number of samples with defects is usually much smaller than that without defects, i.e., the class imbalance issues [2]. Research has shown that for classical machine learning method, such as support vector machine (SVM) [3], decision tree [4], random forest[5], K neighbor [6] etc, when samples are not balanced in the training data, the predictive performance of software defect prediction is dramatically reduced [7].

There are four stages in software defect prediction model construction, including data sampling, feature extraction, classifier optimization and evaluation criteria, and the various corresponding methods are proposed to improve the data imbalance. Data sampling is the initial stage of defect prediction model construction, in which the improvement of data balance can directly reduce the complexity of subsequent stages execution by avoiding usage of methods to reduce the impact of imbalanced data.

Data sampling methods have been proposed to deal with class imbalance issues, including over-sampling, under-sampling and artificial synthesis of minority class samples. Sampling and under sampling methods reduce the imbalanced data by adding samples to the minority class and reduce samples from the majority class respectively [8].But both methods do not add new samples, which may result in no enough feature information for the classifier learning, and lead to over fitting problems. The method of artificial synthesis of data is to synthesize new minority samples on the basis of the original sample information, which may add new feature information while increasing the number of minority samples. However, when there is intra-class imbalance in the original sample, the distribution of new samples generally follows the original. It is clear that the distribution overlap cannot improve the intra-class imbalance.

In order to handle the issue of inter-class and intra-class imbalanced defect data for software defect prediction, a novel data generation method is proposed based on the quantity imbalance and the distribution imbalance. According to the distribution of samples, clustering method is adopted to divide the data into sub-regions. Then different data generation methods are applied for the defective samples under different distribution conditions.

The main contributions of this paper are summarized below. First, we address the combination of inter-class and intra-class imbalanced defect data for software defect prediction. Second, a distribution based data generation method is proposed to generate both quantity and distribution balanced data simultaneously, in which two strategies are introduce to generate data for different distribution. Finally, experiment results based on benchmarks show that the proposed method can generate inter-class and intra-class balanced defects data, therefore to improve the accuracy of software defect prediction model effectively.

The rest of this paper is organized as follows. In Section II, related work is discussed. Section III introduces our proposed method. In Section IV, we perform a comprehensive set of experiments on various research questions. Finally, we conclude our work and suggest directions for future work.

II. RELATED WORK

Researchers focused on each stage in software defect prediction model. The data sampling method in the first stage corresponds to the initial stage of software defect prediction model construction. The balanced training samples obtained in this stage can be directly simplified for the following three stages. The imbalanced sample data intra-class may cause small disjunction problems [9]. However, in general, in the process of feature extraction, the distributor usually selects the samples with dense distribution and ignores the samples with sparse distribution, so the imbalance intra-class will reduce the performance of the classifier [10]. The research work about this stage will be described in detail later. In the feature selection stage, the algorithm of downsizing and principal component analysis is often adopted to select the features that tend to be the few samples [11~12]. Cost

2640-0715/19/$31.00 ©2019 IEEE
DOI 10.1109/APSEC48747.2019.00045

sensitive learning and ensemble learning are used to optimize classifier selection [13~15]. For the classified evaluation, recall rate (Pd), false alarm rate (Pf), F-measure and other indicators can reflect the classification of the minority samples [16].

There are some research results on the data sampling stage. Among them, the ROS method increases the number of samples by randomly copying a minority samples to make the two types of samples balance in number, but does not add new feature information for them. The method of artificial sampling, suggested by Chawla et al. [17] in 2002, in which the number of minority samples is balanced by the randomization of new neighbor synthesis samples, adds new characteristics to the minority samples but does not alter the distribution of the minority samples. Based on the SMOTE method, Han et al. [18] gave the borderline-smote algorithm, with particular emphasis on the establishment of the dates in the concessions. Haibo He et al. [19] proposed ADASYN's adaptive oversample method, which takes into account the distribution of adjacent samples of each sample to generate new sample data. Generate new samples for the minority class samples in the majority class samples, but this method needs to sets an appropriate distance to calculate the neighbors of each sample. Ebo Bennin et al. [20] proposed a MAHAKIL over-sampling method combined with genetic algorithm. This method generates new data through reproduction between samples, and supplements data between samples that are far away from each other, so that samples are continuously distributed, but it is easy to generate a minority types of samples in majority types of sample areas and increase the wrong classification of majority types of samples. Yen et al. [21] proposed an over-sampling method based on clustering, which adds new samples to the small number of classes after the clustering of multi-class samples. It can solve imbalanced data generation problem in multi - classification.

Most of the existing methods to solve the sampling problem of software defect prediction achieve the balance of the number of defective samples and non-defective samples by artificial synthesis of defective samples. However, in practical applications, the distribution of defective samples is uneven, so the newly generated samples will follow the distribution of the original samples, and the composite samples, though balanced in number, are still imbalanced in distribution, that is, the intra-class imbalance problem we pointed out. In this paper, based on the distribution of defective samples, a new method of software defect prediction data generation is proposed. By clustering samples based on different distributions, different data generation strategies are adopted in different distribution regions to achieve intra-class balance while increasing the number of defective samples, thus improving the performance of classifiers.

III. METHOD

In this section we introduce our novel method to generate new sample data for Software defect prediction. First extracts relevant features from existing data sets, which are usually related measures of software code analysis, such as the number of lines of code, Halstead science measure, McCabe loop complexity and so on. Code features can represent the possibility of software errors. It is generally believed that the higher the code size or complexity of the program module, the higher the possibility of internal defects [22].

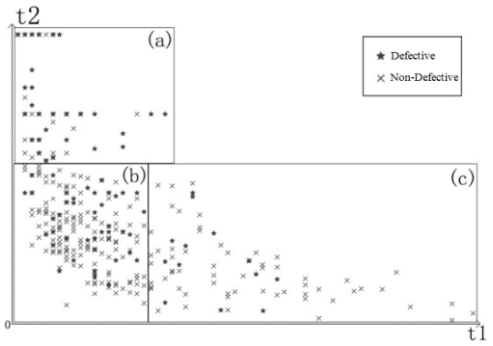

Fig. 1. Promise Dataset Distribution

Take the distribution of the two characteristics of module average complexity and inter-module cohesion in the commonly used open source dataset Promise as an example. As shown in Fig.1, the data distribution can be described in three regions. In Fig. 1(a), the number of samples with defects is larger than that of non-defects. In classification, the defective samples in this part are easy to be found. In Fig. 1(b), defects and non-defects samples are mixed. However, the number of defective samples is smaller than that of non-defective ones. In this case, in order to ensure the accuracy of overall classification, the defective samples may be misclassified. The number of defective samples in Fig. 1(c) region is much smaller than non-defective ones, and the number of defective samples is very small compared with that in region (a) (b). In classification, these defective samples may be ignored as noise data. According to the distribution in Fig. 1, the number of samples with defects is smaller than that without defects, and there is an inter-class imbalance. In addition, the distribution of defective samples is not uniform, and there is intra-class imbalance. The results show that both inter-class imbalance and intra-class imbalance affect the accuracy of classification results.

In order to reduce the impact of data imbalance on software defect prediction, this paper studies the initial data sampling stage of prediction model construction, and proposes a new data generation method.

Generally the defective sample is much smaller than the non-defective sample, If we divide the samples in the feature space, the distribution of sample data may appear the following four conditions: non-defective sample is far greater than defective sample, non-defective sample is greater than defective sample, defective sample is greater than non-defect sample and defective sample is far greater than non-defective sample.

However, because the small number of defective samples in the overall data set, the possibility of the fourth case is extremely small. Therefore, this paper discusses the first three distributions, and generates new defective sample data according to the new method to solve the problem of inter-class imbalance. In order to further solve the intra-class imbalance problem, the paper divides the sample into several sub-regions according to their distribution in the feature space. More new samples are generated in the sparsely distributed regions while fewer new samples are generated in the densely distributed regions to achieve an overall distribution balance of the defective samples. Fig. 2 shows the method framework, which first clusters all the samples in the original dataset and divides them into several sample regions.

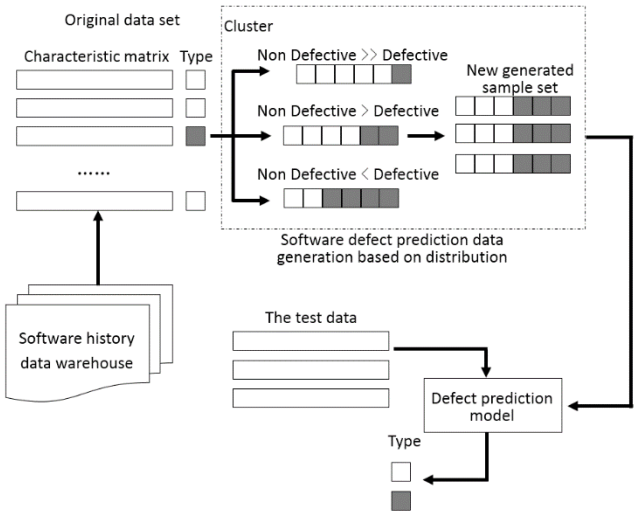

Fig. 2. Software defect prediction framework

Then, according to the different distribution of the defective and non-defective samples in each region, the generation strategy of the defect sample data is selected, and the inter-class balance and the intra-class balance are realized at the same time. Finally, build a defect prediction model and predict the test data.

Algorithm 1 defect data set partition method based on k-means clustering

Input: Training set sample X, training set sample number n

Output: Three clusters after clustering: Xa,Xb,Xc

(1) K=3 // Clustering cluster number

(2) Select Center1 // randomly select 1 sample as the first cluster

(3) For *i*←1 to 3

(4) For *j*←1 to n

(5) $D_j \leftarrow \sum_1^i \sqrt{(X_j\text{-Center i})^2}$

(6) Select Center *i*

(7) End

(8) End

(9) NewCenter ← Center

(10)） Repeat

(11) For j←1 to n

(12) For i←1 to 3

(13) $d_{ji} \leftarrow \sqrt{(X_j\text{-Center i})^2}$

(14) Divide(Xj) // A cluster divided into the nearest cluster

(15) End

(16) End

(17) NewCenter ← $\bar{X}$

(18) Until Center = NewCenter

(19) Return Cluster after clustering Xa,Xb,Xc.

A. Data Set Partitioning Model Based on K-Means Clustering

The first step of the method in this paper is to partition the data set. Among the existing partitioning methods, k-means clustering is the most widely used efficient partitioning algorithm [23], which can be used to partition large-scale data and high-dimensional data. The samples in the similar feature regions are divided into the same sub-cluster according to the distance between the samples. In this paper, the k-means partitioning algorithm is used to cluster the defective samples and non-defective samples in the original sample to understand the distribution of sample data.

According to the previous analysis, after sample partition, the distribution of defective and non-defective samples in each region will correspond to one of the three situations in Fig. 1 (a), (b) and (c). In this paper, K=3 was selected to divide the data set.

The specific steps of the defect data set partitioning method based on k-means clustering are shown in algorithm 1. Firstly, a cluster center is randomly selected from all samples, and the distance from the sample to the cluster center is calculated. The second and third cluster centers are selected according to the distance. The principle of selection is that the larger the distance, the higher the probability of being selected. Then calculate the distance between the samples and each cluster center, and divide the samples into clusters with the nearest cluster center. After partitioning, the average of the samples in each cluster is calculated as the new cluster center. The clustering of the samples is adjusted according to the new cluster center until the cluster center no longer changes and the final clustering is obtained.

B. Data Generation Method Based on Distribution Division

After partitioning the data set with the k-means algorithm, it is necessary to generate data for defective samples within three clusters. In order to ensure the inter-class balance, this method first calculates the number of defective samples to be generated, and calculates the number of defective samples to be generated for each cluster according to the ratio between the number of non-defective samples and the number of defective samples. In order to ensure the balance within the new sample set, according to the characteristics of data distribution within each cluster, this paper selects different data generation methods to generate different quantities of defective samples.

The specific implementation process is shown in algorithm 2. Firstly, the ratio between the number of non-defective samples and the number of defective samples in each cluster is calculated, and the calculation formula is as follows:

$$Ki = \frac{Ni_{xn}}{Ni_{xd}} \quad (1)$$

Where, Ni_{xd} represents the number of defective samples in cluster *i;* Ni_{xn} represents the number of non-defective samples in cluster *i*. When *Ki* is taken as the generation ratio, the larger *Ki* is, the larger the number of non-defect samples in this cluster, the smaller the number of defective samples, and the larger the number of defective samples to be generated. On the contrary, the smaller *Ki* is, the fewer non-defective samples are in this cluster, and the more defective samples are, the fewer defective samples need to be generated. Therefore, the final data generation ratio in each cluster is:

$$\mathrm{Ti} = Ki / \sum_{i=1}^{3} Ki \quad (2)$$

Next, set the defect proportion *p*, that is, the proportion of defective samples in the data set, and generally set *p = 50%* in the case of inter-class balance. According to the defect proportion *p*, the number of defective samples to be generated *C* can be calculated:

$$C = N * P - N_{xd} \quad (3)$$

Where N is the total number of samples, and N_{xd} is the total number of defective samples. Then, we can calculate the number of new samples to be generated in each cluster, Ci:

$$Ci = C * Ti \tag{4}$$

In the establishment of new samples, in Fig. 1(a)(b), there were many defective samples in the two distributions. Therefore, SMOTE method was used directly in the two clusters to obtain the new samples, in other words, using pairwise the defective samples in the same cluster to give the new samples, as shown in Fig. 3.

As for the distribution in Fig. 1(c), the number of defective samples was very small, and there was not enough of the same kind of neighbor samples to give the dates. The SMOTE method no longer suited, so a circle method was used to give the new data, as shown in Fig 4. Firstly, randomly select a defective sample in the cluster, find the adjacent defective sample and calculate the distance r between the two samples, take the selected sample point as the center of the circle and r as the radius to make a circle, and generate a new sample randomly in the circle. The circular domain generation method is used to generate more new samples evenly around the defective samples.

Algorithm 2 describes the data generation method based on distribution division. In the case of three different defect sample distributions in Fig. 1(a)(b)(c), different strategies are adopted to generate new sample data. The number of defective samples generated in Fig. 1(a) region is small, while the number of defective samples generated in Fig. 1(c) is large. There may be inter-class imbalance in the divided small area, that is, the number of samples with defects in Fig. 1(a) is larger than the number of non-defective samples, but the data set as a whole is balanced between classes. The data generation method based on clustering division adopts different data generation strategies for different distribution situations, so that the number of defective samples in the three regions (a), (b) and (c) of the generated data set is not significantly different, and the imbalance intra-class is solved, which can ensure that the defective samples are relatively balanced in the overall distribution.

Algorithm 2 data generation method based on clustering partition

Input: Xa,Xb,Xc clustering training set; Defect ratio p

Output: newly generated sample set Xnew

(1) For i←1 to 3

(2) $\mathrm{Ki} \leftarrow \frac{Ni_{xn}}{Ni_{xd}}$

// the ratio of Ni_{xd} (defects)samples to Ni_{xn}(non-defects)

(3) End

(4) $\mathrm{K} \leftarrow \sum \mathrm{Ki}$

(5) For i←1 to 3

(6) $\mathrm{Ti}=(\mathrm{Ki} / \mathrm{K})$

(7) End

(8) $\mathrm{C}=\mathrm{N}*\mathrm{p} - \mathrm{N}_{xd}$

(9) For i←1 to 3

(10) $\mathrm{Ci}=\mathrm{C}*\mathrm{Ti}$

(11) IF Xi=Xa OR Xi=Xb THEN

(12) Smote(Xi, Ci) // use the SMOTE method

(13) ELSE

(14) Circle(Xi, Ci) // new sample is generated

(15) End

(16) End

(17) output the newly generated sample set Xnew.

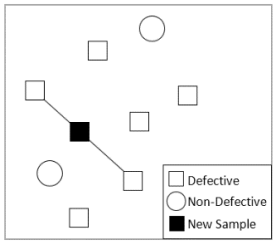

Fig. 4. The illustration for sample generation for (a) and (b) regions

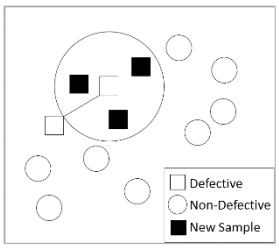

Fig. 3. The illustration for sample generation for (c) region

IV. EXPERIMENT AND ANALYSIS

A. Experimental design

In order to verify the effectiveness of the distributed software defect prediction data generation method proposed in this paper, the ten-fold intersection method is used to verify the results. The experiment results are compared with the existing 4 sampling algorithms: ROS(oversampling), SMOTE, ADASYN, MAHAKIL, and experiment with KNN, SVM, Random forest, C4.5. The experimental hardware environment was CPU Intel(R) Core(TM) i7-7700 3.6GHz, RAM 8.00GB, the operating system was Win10, and the program compilation environment was MATLAB R2016b.

1) The subjects

In this paper, 9 items in Promise Repository public data set are used to build software defect prediction model for experiments. Table I lists all the characteristics used by the data set, which are common static characteristics.

The items in the dataset are written in JAVA, with the defective module marked as 1 and the non-defective module marked as 0. Download from http://www.madeyski.com/. Table I lists all the characteristics used by the data set, which are common static characteristics in software engineering.

Table II lists the basic information of the data set, including project size, number of defect modules contained, and proportion of defect modules.

TABLE I. THE PROMISE DATASET FEATURE

Name	feature	Name	feature
NOC	Number of Children	DIT	Depth of Inheritance Tree
DAM	Data Access Metric	IC	Inheritance Coupling
RFC	Response for a Class	Ca	Afferent couplings
MOA	Measure of Aggregation	Ce	Efferent couplings
WMC	Weighted methods per class	NPM	Number of Public Methods
LOCM	Lack of cohesion in methods	CBM	Coupling Between Methods
LOC	Lines of Code	AMC	Average Method Complexity
CBO	Coupling between object classes		
LOCM3	Lack of cohesion in methods, different from LCOM		
MFA	Measure of Functional Abstraction		
CAM	Cohesion Among Methods of Class		
CC	McCabe's cyclomatic complexity		
Max(CC)	Maximum value of CC methods of the investigated class		
Avg(CC)	Arithmetic mean of the CC value in the investigated class		
BUG	Number of detected bugs in the class		

TABLE II. THE PROMISE DATASET INFORMATION

Data sets	module	Defective module	Defect module ratio
Xerces	588	151	25.68%
Tomcat	858	77	8.97%
Redaktor	176	27	15.34%
Log4j	205	16	7.80%
Jedit	492	11	2.24%
Ivy	352	40	11.36%
Camel	339	13	3.83%
Arc	234	27	11.54%
Ant	745	166	22.28%

TABLE III. CONFUSION MATRIX OF SOFTWARE DEFECT PREDICTION

Predicted /	Defective	Non Defective
Defective	A	B
Non Defective	C	D

2) Evaluation indicators

The evaluation criteria used in this paper are several common metrics in machine learning classification problems, including: Pd, Pf, and F-measure. These indicators are calculated according to the confusion matrix. Table III is the confusion matrix of software defect prediction model.

The case where the defective data prediction is correctly predicted as defective data is denoted as A, and the erroneous prediction is expressed as non-defective data as B.

The case where the defective data prediction is correctly predicted as defective data is denoted as A, and the erroneous prediction is expressed as non-defective data as B. Where the prediction of the non-defective data, the erroneous prediction is represented as C, and the correct prediction is represented as D.

$$Pd = \frac{A}{A+B}, Pf = \frac{C}{C+D}, Presion = \frac{A}{A+C},$$
$$F-measure = \frac{A2 \times Pd \times Presion}{Pd+Presion} \quad (5)$$

Where Pd value represents the proportion of defective samples correctly classified, so the larger the Pd, the better; Pf represents the proportion of defect-free samples that are mis-predicted, so smaller Pf is better. F-measure is the harmonic average value of Pd and Presion, It is a comprehensive evaluation index, and the larger the measure is, the better the classification effect will be.

B. Experimental results and analysis

This paper conducts experiments and analysis on the following three research questions:

- RQ1: Can distributed software defect prediction data generation method reduce the influence of class imbalance on software defect prediction classification process?
- RQ2: What is the performance of the distribution-based software defect prediction data generation method when using different classification algorithms?
- RQ3: When using this method to generate data, what proportion of defective sample data can make the best classification effect?

In order to verify the effectiveness of the method presented in this paper, the method is compared with the sampling methods of ROS, SMOTE, ADASYN and MAHAKIL, using SVM, KNN, C4.5 and RF classification algorithms.

TABLE IV. PD VALUE RESULTS

Pd		xercs	tom	reda	log4j	jedit	ivy	came	arc	ant
ROS	**KNN**	0.89	0.40	0.45	0.50	0.83	0.20	0.62	0.40	0.44
	C4.5	0.93	0.70	0.86	0.80	0.66	0.43	0.70	0.48	0.54
	RF	0.93	0.73	0.80	0.92	0.69	0.56	0.79	0.57	0.62
	SVM	0.91	0.83	0.88	0.85	0.73	0.46	0.92	0.56	0.23
SMOTE	**KNN**	0.80	0.90	0.74	0.75	0.83	0.83	0.84	0.59	0.72
	C4.5	0.89	0.85	0.79	0.86	0.93	0.78	0.87	0.84	0.78
	RF	0.88	0.89	0.76	0.89	0.93	0.82	0.91	0.81	0.76
	SVM	0.84	0.82	0.73	0.57	0.62	0.66	0.75	0.21	0.66
ADASYN	**KNN**	0.86	0.91	0.93	0.92	0.98	0.90	0.97	0.90	0.80
	C4.5	0.92	0.93	0.90	0.95	0.98	0.91	0.96	0.90	0.84
	RF	0.92	0.92	0.91	0.93	0.98	0.89	0.97	0.87	0.82
	SVM	0.91	0.88	0.87	0.88	0.91	0.85	0.97	0.82	0.81
MAHAKIL	**KNN**	0.60	0.41	0.81	0.80	0.87	0.68	0.87	0.76	0.30
	C4.5	0.84	0.67	0.94	0.93	0.92	0.87	0.94	0.92	0.93
	RF	0.90	0.77	0.94	0.97	0.94	0.89	0.94	0.92	0.78
	SVM	0.84	0.55	0.57	0.71	0.79	0.56	0.77	0.53	0.63
DBDGM	**KNN**	0.88	0.93	0.86	0.95	0.99	0.95	0.98	0.89	0.83
	C4.5	0.94	0.93	0.88	0.95	0.99	0.89	0.98	0.89	0.87
	RF	0.94	0.97	0.95	0.97	0.99	0.90	0.99	0.95	0.88
	SVM	0.94	0.95	0.94	0.90	0.92	0.86	0.96	0.95	0.85

TABLE V. PF VALUE RESULTS

Pf		xerc	tom	reda	log4	jedit	ivy	cam	arc	ant
ROS	**KN**	0.10	0.07	0.09	0.07	0.03	0.05	0.03	0.08	0.12
	C4.5	0.10	0.07	0.12	0.09	0.04	0.11	0.05	0.10	0.15
	RF	0.05	0.03	0.07	0.03	0.03	0.05	0.02	0.04	0.08
	SV	0.06	0.01	0.01	0.01	0.02	0.01	0.00	0.05	0.00
SMOTE	**KN**	0.05	0.07	0.06	0.10	0.07	0.07	0.10	0.13	0.04
	C4.5	0.08	0.08	0.12	0.13	0.07	0.11	0.11	0.13	0.14
	RF	0.03	0.05	0.08	0.03	0.05	0.07	0.04	0.10	0.09
	SV	0.06	0.08	0.09	0.05	0.07	0.06	0.06	0.01	0.09
ADASYN	**KN**	0.07	0.05	0.13	0.08	0.03	0.09	0.07	0.14	0.07
	C4.5	0.08	0.07	0.13	0.05	0.03	0.11	0.07	0.11	0.16
	RF	0.03	0.03	0.05	0.02	0.03	0.03	0.02	0.06	0.08
	SV	0.08	0.03	0.12	0.11	0.06	0.07	0.06	0.08	0.08
MAHAKIL	**KN**	0.08	0.08	0.09	0.11	0.04	0.13	0.03	0.23	0.09
	C4.5	0.10	0.11	0.17	0.11	0.05	0.17	0.08	0.16	0.10
	RF	0.06	0.05	0.10	0.03	0.03	0.06	0.02	0.10	0.08
	SV	0.03	0.01	0.00	0.01	0.02	0.01	0.00	0.05	0.07
DBDGM	**KN**	0.07	0.05	0.10	0.14	0.06	0.11	0.05	0.18	0.11
	C4.5	0.08	0.10	0.15	0.07	0.05	0.14	0.05	0.17	0.15
	RF	0.03	0.06	0.08	0.02	0.03	0.04	0.03	0.12	0.08
	SV	0.05	0.05	0.05	0.02	0.04	0.03	0.01	0.08	0.05

The experimental results show that the new method generates new sample data, and the proportion of defects in the data set is 50%, which achieves the inter-class balance. Table IV~VI shows the experimental results of Pd value, Pf value and F-measure value respectively.

Fig. 5~7 corresponds to Table IV~VI, and different graphs are used to represent the results of different Data Generation algorithms on each Data set.

TABLE VI. F-MEASURE VALUE RESULTS

F-measure		xe	tom	reda	log4	jedit	ivy	cam	arc	ant
ROS	**KNN**	0.9	0.43	0.55	0.53	0.69	0.26	0.67	0.41	0.48
	C4.5	0.9	0.67	0.77	0.71	0.53	0.41	0.66	0.45	0.53
	RF	0.9	0.72	0.82	0.88	0.62	0.60	0.79	0.64	0.67
	SVM	0.9	0.84	0.80	0.92	0.67	0.71	0.94	0.75	0.37
SMOTE	**KNN**	0.8	0.90	0.81	0.78	0.86	0.87	0.84	0.67	0.80
	C4.5	0.8	0.86	0.83	0.84	0.91	0.82	0.86	0.84	0.79
	RF	0.9	0.90	0.80	0.91	0.93	0.85	0.93	0.82	0.80
	SVM	0.8	0.82	0.80	0.69	0.71	0.77	0.80	0.50	0.73
ADASYN	**KNN**	0.8	0.94	0.91	0.93	0.97	0.92	0.96	0.90	0.84
	C4.5	0.9	0.92	0.88	0.95	0.96	0.85	0.95	0.89	0.85
	RF	0.9	0.94	0.93	0.96	0.97	0.93	0.97	0.90	0.86
	SVM	0.9	0.92	0.88	0.89	0.91	0.89	0.95	0.85	0.86
MAHAKIL	**KNN**	0.6	0.49	0.81	0.84	0.88	0.73	0.91	0.76	0.82
	C4.5	0.8	0.66	0.90	0.91	0.91	0.82	0.91	0.90	0.75
	RF	0.8	0.73	0.94	0.97	0.93	0.89	0.95	0.92	0.79
	SVM	0.8	0.52	0.70	0.83	0.87	0.73	0.87	0.70	0.67
DBDGM	**KNN**	0.9	0.94	0.87	0.92	0.97	0.92	0.97	0.86	0.85
	C4.5	0.9	0.92	0.87	0.94	0.97	0.87	0.96	0.86	0.85
	RF	0.9	0.96	0.93	0.98	0.98	0.93	0.96	0.92	0.89
	SVM	0.9	0.95	0.94	0.94	0.94	0.91	0.97	0.93	0.89

The distribution-based Data Generation Method (DBDGM) proposed in this paper is represented by black five-pointed stars in the figure, and the dotted line in the figure represents the average value of the five data generation method.

1) RQ1

The purpose of research question 1 is to verify the improvement effect of distribution-based data generation method in software defect prediction. The method in this paper deals with the imbalanced data in the data sampling stage, so it is compared with the existing four imbalanced data processing methods in the data sampling stage.

Fig.5 shows the results of Pd values of 5 data generation methods under 4 classification algorithms. The higher the Pd value, the better the effect. It can be seen that all the results of the method in this paper are above the average level, and the results are the best under all the classification algorithms on the xerces, tomcat, log4j and jedit data sets, among which the proportion of defects in the tomcat, log4j and jedit data sets is less than 10%, indicating that the method in this paper performs best under the condition of serious class imbalance. On other data sets, Pd values of the method in this paper are in the top two in all classification algorithms, indicating that the method in this paper has good performance in data distribution with different defect ratios, which is conducive to improving the accuracy of classification of defective samples.

Fig. 6 shows the results of Pf value of 5 data generation methods under 4 classification algorithms. The lower Pf value, the better effect. In this method, most Pf values are lower than the average value, and a few are slightly higher than the average value, but the difference is all within 0.1. Although a few samples without defects are misclassified, the Pf value loss is small. It shows that the method in this paper not only improves the classification accuracy of defective samples, but also does not increase the wrong classification of defective samples.

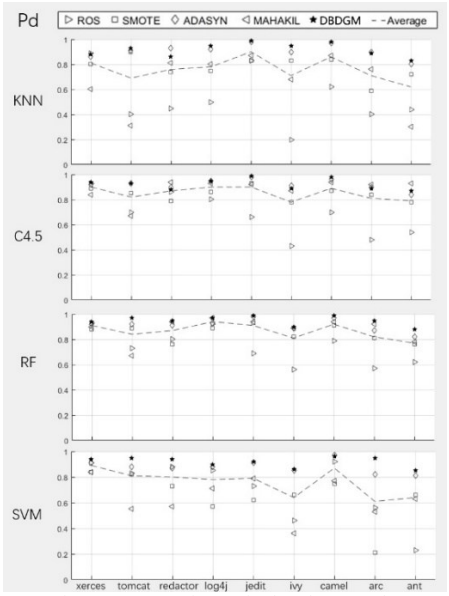

Fig. 5. Comparison of Pd Value Results

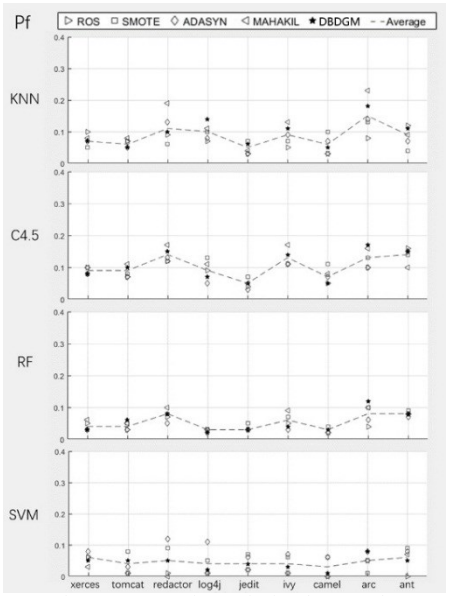

Fig. 6. Comparison of Pf Value Results

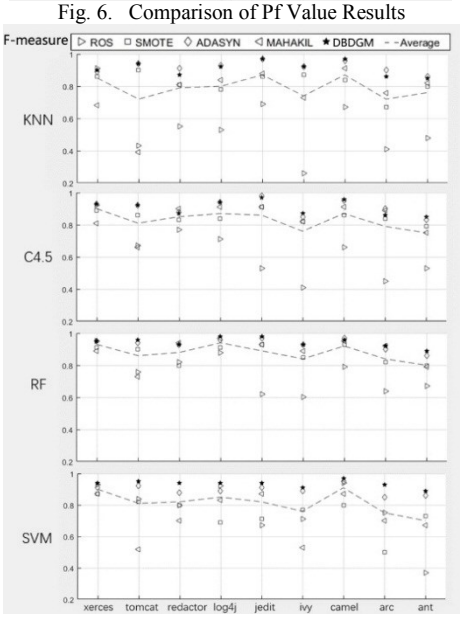

Fig. 7. Comparison of F-measure Value Results

Fig. 7 shows the results of F-measure values of five data generation methods under four classification algorithms. The higher the F-measure value, the better the effect. Method in this paper on the full results are higher than average, the method in the RF classification algorithm and SVM classification algorithm of F-measure value on all the data set is the highest, remaining in the top two, most of the results is relatively average has a lot to improve, tomcat, log4j, jedit, ivy and other defects of small data sets F-measure to gain maximum, this method to the class serious imbalance data set classification has a lot to ascend.

The experimental results show that this method is superior to the other four data generation methods, greatly improves the classification performance of the classifier, and is an effective data generation method.

2) RQ2

The purpose of research question 2 is to verify whether the distribution-based data generation method can improve the prediction accuracy under different classification algorithms. Therefore, the results of software defect prediction under four common classification algorithms are compared.

From the results provided in Fig. 5~7, it can be observed that the Pd values of this method under the four classification algorithms are all higher than the average level, which is greatly improved, and all the data sets rank the first two places of several methods. Experimental results of Pf value show that although this method does not always obtain the best Pf value, it is still in the average level and most of them are lower than the average value.

TABLE VII. COMPARISON OF THE BEST AND WORST RESULTS

Classifier	Sampling method	Best	Worst	Best - Worst
RF	DBDGM	20	2	18
SVM	DBDGM	17	1	16
C4.9	DBDGM	13	1	12
KNN	DBDGM	12	1	11
C4.7	ADASYN	10	1	9
RF	ADASYN	9	0	9
KNN	ADASYN	9	1	8
SVM	MAHAKIL	7	5	2
RF	MAHAKIL	5	4	1
C4.6	SMOTE	3	3	0
KNN	SMOTE	2	3	-1
C4.8	MAHAKIL	6	8	-2
SVM	ROS	5	7	-2
RF	SMOTE	1	4	-3
RF	ROS	5	9	-4
SVM	ADASYN	1	5	-4
KNN	MAHAKIL	1	6	-5
C4.5	ROS	5	14	-9
SVM	SMOTE	1	10	-9
KNN	ROS	6	17	-11

This method has very high F-measure values under different classification algorithms, and the F-measure values on most data sets are the highest. The F-measure results show that this method improves the comprehensive performance of the classifier.

TABLE VIII. EXPERIMENTAL RESULTS OF DIFFERENT DEFECT PROPORTIONS

	Defect ratio	30%	Change	40%	Change	50%	Change	60%
KNN	pd	0.80	+0.08	0.88	+0.04	0.92	+0.02	0.94
	pf	0.08	-0.01	0.09	-0.01	0.10	-0.02	0.12
	F-measure	0.80	+0.07	0.87	+0.05	0.92	0.00	0.92
C4.5	pd	0.83	+0.06	0.89	+0.03	0.92	+0.03	0.95
	pf	0.08	-0.01	0.09	-0.02	0.11	-0.02	0.13
	F-measure	0.81	+0.07	0.88	+0.03	0.91	+0.01	0.92
RF	pd	0.85	+0.05	0.90	+0.05	0.95	+0.02	0.97
	pf	0.04	-0.01	0.05	0.00	0.05	-0.02	0.07
	F-measure	0.86	+0.05	0.91	+0.03	0.94	+0.01	0.95
SVM	pd	0.80	+0.06	0.86	+0.06	0.92	+0.02	0.94
	pf	0.04	0.00	0.04	0.00	0.04	-0.02	0.06
	F-measure	0.84	+0.06	0.90	+0.04	0.94	+0.00	0.94

Table VII makes statistics on the results of Fig. 5-7, and calculates the optimal and worst times of Pd, Pf and F-measure values of 5 data generation methods under 4 classification algorithms in 9 projects, and sorts them according to the difference between the optimal and worst times. The results of the optimal times and the worst times of the method in this paper are all positive, and the results of the combination of the top 4 and 4 different classification algorithms with this method are better than other combination methods. The combination of RF classification algorithm and the method in this paper can get the best experimental results. Therefore, this method has good performance under different classification algorithms.

3) RQ3

Research question 3 discusses the data generation ratio. When the proportion of the two types of samples is greater than 1:4, the classification results will be greatly affected. When the proportion of defects in the experimental discussion is 30%, 40%, 50% and 60%, even if the proportion of the two types of samples is less than the proportion of 1:4, the smallest proportion of data generation is found when the classification results are the best.

In order to find out the most suitable proportion of data generation, the classification results of the classification algorithm under different proportion of defects after data generation (that is, the proportion of defective samples in the data set) are discussed in the experiment.

Table VIII records the average values of Pd, Pf and F-measure of 9 items under the four classification algorithms when the proportion of defects is 30%, 40%, 50% and 60%. The improvement of Pd value and the loss of Pf value during the change of defect proportion are recorded in the table. "+" is used to represent the improvement of experimental results, and "-" is used to represent the loss of experimental results. In the table, light gray means Pd value increase is greater than Pf value loss, and dark gray means Pd value increase is less than Pf value loss or both are equal. When the proportion of defects is 30% to 40%, the experimental results are greatly improved. When the proportion of defects was 40% to 50%, the experimental results were also slightly improved.

When the proportion of defects changes from 50% to 60%, due to the increase of Pf value loss, the improvement of experimental results is very small, or even the loss is greater than the improvement.

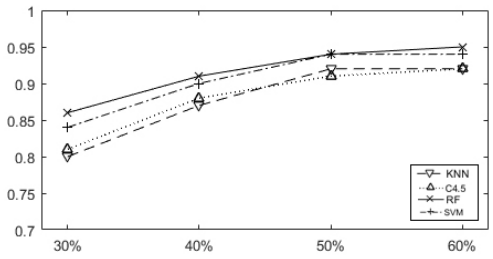

Fig. 8. F-measure value of different defect proportion

Fig. 8 corresponds to the F-measure value in table 5, intuitively showing the change of F-measure value with the proportion of defects. Before the proportion of defects reaches 50%, the F-measure value is promoted rapidly, and in the process of 50% to 60%, the F-measure value is promoted very little. KNN and SVM classification algorithms reach the maximum value when the proportion of defects is 50%, while C4.5 and RF classification algorithms method the maximum value when the proportion of defects is 50%, but with the increase of the proportion, there will be a small increase. Therefore, in the case of no specific regulation of classification algorithm, 50% of defects is the most appropriate.

V. CONCLUSIONS AND PROSPECTS

In the software defect data set, there are serious intra-class imbalance and inter-class imbalance, which affect the accuracy of software defect prediction. In order to reduce the influence of data imbalance on software defect prediction, this paper proposes a novel imbalanced data generation method. The method clusters the samples according to the distribution of the samples, and then generates data for the defective samples under different distribution conditions, so that the generated data sets reach the inter-class balance and intra-class balance. In this paper, an actual software project is used for experiments, and the results show that after data balancing by this method, more defective samples are correctly classified, and the comprehensive performance of the prediction model is significantly improved. In the data imbalance problem of software defect prediction, this method can solve both the intra-class imbalance and inter-class imbalance, and effectively improve the classification accuracy.

In the future work, the application of cross-project defect prediction problem can be further explored. There is still class imbalance in cross-project defect prediction problem. Whether the method in this paper can be used to deal with cross-project class imbalance problem needs to be further studied.

ACKNOWLEDGMENT

The work described in this paper is supported by the National Natural Science Foundation of China under Grant No. 61872026, 61672085 and 61702029, and the Fundamental Research Funds for the Central Universities (XK1802-4)

REFERENCES

[1] Li-Na G, Shu-Juan J, Li J. Research Progress of Software Defect Prediction [J]. Journal of Software, 2019, 30(10): 3090-3114.

[2] Xiaofeng Z, Depi Z. Software defect prediction based on imbalanced datasets [J]. Application research of computer, 2017, 34(7): 2027-2031.

[3] Elish K O , Elish M O . Predicting defect-prone software modules using support vector machines[J]. Journal of Systems & Software, 2008, 81(5):649-660.

[4] Lessmann S , Baesens B , Mues C , et al. Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings[J]. IEEE Transactions on Software Engineering, 2008, 34(4):485-496.

[5] Ho T K. Random decision forests[J]. 1995.

[6] Cover T, Hart P. Nearest neighbor pattern classification[J]. IEEE Trans.inf.theory, 1953, 13(1):21-27.

[7] Wang S , Yao X . Using Class Imbalance Learning for Software Defect Prediction[J]. IEEE Transactions on Reliability, 2013, 62(2):434-443.

[8] Barua S , Islam M M , Yao X , et al. MWMOTE--Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(2):405-425.

[9] Jo T, Japkowicz N. Class imbalances versus small disjuncts[J]. Acm Sigkdd Explorations Newsletter, 2004, 6(1):40-49.

[10] He H, Garcia E A. Learning from Imbalanced Data[J]. IEEE Transactions on Knowledge & Data Engineering, 2009, 21(9):1263-1284.

[11] Jiehua W , Hong X . Dual feature selection imbalanced complex network link classification model[J]. Application Research of Computers, 2018.

[12] Jing X Y , Wu F , Dong X , et al. An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project Class-Imbalance Problems[J]. IEEE Transactions on Software Engineering, 2016, 43(4):1-1.

[13] Khan S H , Hayat M , Bennamoun M , et al. Cost Sensitive Learning of Deep Feature Representations from Imbalanced Data[J]. IEEE Transactions on Neural Networks & Learning Systems, 2015, 29(8):3573-3587.

[14] Stolfo W F S J . AdaCost: Misclassification Cost-sensitive Boosting[C]// Sixteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc. 1999.

[15] Ryu D , Jang J I , Baik J . A transfer cost-sensitive boosting approach for cross-project defect prediction[J]. Software Quality Journal, 2017, 25(1):235-272.

[16] Pelayo L , Dick S . Evaluating Stratification Alternatives to Improve Software Defect Prediction[J]. IEEE TRANSACTIONS ON RELIABILITY R, 2012, 61(2):516-525.

[17] Chawla N V , Bowyer K W , Hall L O , et al. SMOTE: Synthetic Minority Over-sampling Technique[J]. Journal of Artificial Intelligence Research, 2002, 16(1):321-357.

[18] Han H, Wang W Y, Mao B H. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning[C]// International Conference on Advances in Intelligent Computing. 2005.

[19] He H , Bai Y , Garcia E A , et al. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning[C]// Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on. IEEE, 2008..

[20] Ebo Bennin K , Keung J , Phannachitta P , et al. MAHAKIL:Diversity based Oversampling Approach to Alleviate the Class Imbalance Issue in Software Defect Prediction[J]. IEEE Transactions on Software Engineering, 2017:1-1.

[21] Yen S J, Lee Y S. Cluster-based under-sampling approaches for imbalanced data distributions[J]. Expert Systems with Applications, 2009, 36(3-part-P1):5718-5727.

[22] Xiang C , Qing G U , Shu L W , et al. Survey of Static Software Defect Prediction[J]. Journal of Software, 2016.

[23] Nguyen B , De Baets B . Kernel-Based Distance Metric Learning for Supervised k-Means Clustering[J]. IEEE Transactions on Neural Networks and Learning Systems, 2019:1-12.