Int J Syst Assur Eng Manag
https://doi.org/10.1007/s13198-023-02031-x

ORIGINAL ARTICLE

Diversity based multi-cluster over sampling approach to alleviate the class imbalance problem in software defect prediction

C. Arun[1] · **C. Lakshmi**[1]

Received: 18 May 2023 / Revised: 23 June 2023 / Accepted: 14 July 2023

Abstract Advancement in the field of Artificial Intelligence and Machine Learning has paved the way to enhance the quality of software by creating advanced testing tools and enhanced Software Defect Prediction (SDP) models. Significant growth in the need of Software component across domains impose significant challenge on the complexity and reliability of the software component. However, the Software Practitioners try to create advanced SDP models to find defect-prone modules effectively. The Performance of the prediction model is correlated with the quality and quantity of the dataset used. Over the years researchers have contributed numerous works to counter the class-imbalance issue in the SDP model by using data sampling, ensemble learning and cost-sensitive learning. However, the smaller disjuncts is also other factor which impact the performance of SDP model. To counter both class imbalance and smaller disjuncts, we proposed a multipatch cluster based oversampling approach which generating synthetic samples to balance the class and ensure the samples reside within class boundary and eliminate the possibility of minority samples evade decision boundary. Initial Population was divided into two groups majority and minority samples respectively. Mahalanobis distance is used to calculate the diversity of individual samples of the minority cluster from the population. Then, samples are placed into various clusters, and by taking into account the density, synthetic samples are introduced into each cluster. Five different machine learning models have been used to test the performance of the proposed approach. The experimental findings demonstrate the effectiveness of the algorithm by proving that the proposed approach offers better performance in terms of a reduced false alarm rate.

Keywords Software defect prediction · Mahalanobis · Class imbalance · Cluster · Oversampling · Classification · Synthetic sample generation

1 Introduction

Software modules have become a crucial component of every business model since the software industry has overgrown, regardless of the industry core product manufacturing, banking, healthcare, aviation, medicine, e-commerce, social networking, education, or any other. Delivering software components with desirable quality is a challenging task in the development process. Software Quality Assurance (SQA) aims to ensure the desired software quality at a reasonable price by monitoring and managing the development process. SQA comprises inspection, walk-through, code-review, peer-review, software fault prediction, and testing. Jones and Bonsignour (2012) reported that finding and correcting defects is one of the most expensive software development activities. The complexity of the software increases with the progress of the software development stages, making the defects to hide deeper and more challenging. Unidentified defects manifest themselves and cause the application to malfunction. Figure 1 Shows the generic Software defect prediction framework architecture. So, early prediction of defects has been a recent interest among researchers, which not only help to identify flaws in the early stages but also reduce the cost and effort. By using software metrics (Schneider et al. 1992; Kamei,

✉ C. Arun
arunc@srmist.edu.in

C. Lakshmi
lakshmic@srmist.edu.in

[1] Department of Computational Intelligence, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Chennai 603203, Tamilnadu, India

Published online: 27 July 2023

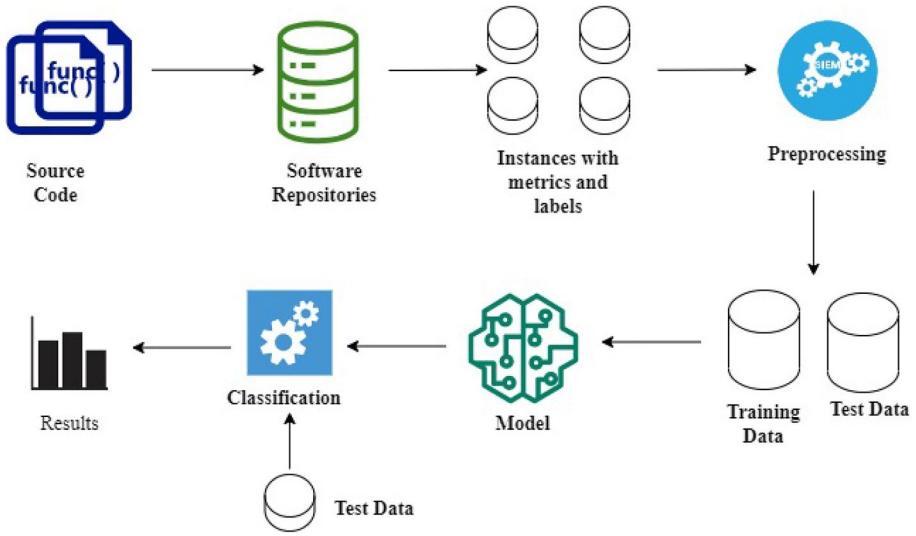


Fig. 1 Software defect prediction framework

et al. 2012; Hall et al. 2011; Pandey et al. 2021), software defect prediction (Weiss and Provost 2001; Yoon and Kwek 2007) aims to narrow down the software project's most likely fault-prone modules. Early prediction assists the Software Quality assurance team in allocating finite resources effectively to build quality software.

Regression, SVM, KNN, Decision Tree, and other techniques have been used for years to build SDP models, but these models depend on a good amount of training data. Unfortunately,the Defect data set suffers due to the class imbalance problem. The amount of data in one class outnumbers the number of instances in another class, leading to biased learning (Provost 2000; Diez-Pastor et al. 2015; Thabtah et al. 2020). Researchers have proposed a wide range of approaches to confronting the class imbalance, including class rebalancing through sampling, algorithmic, and ensemble-based approaches. Sampling techniques either keep control of the number of majority instances, i.e., under-sampling, or increases the minority instances by introducing synthetic samples (Gao et al. 2014).

Algorithmic-based approaches involve adaptive weight learning, cost-sensitive, and threshold adjustment methods (Guo and Viktor 2004; Sri Kavya 2020; Ozturk 2017; Zhang et al. 2015). Finally, ensemble-based strategies, including bagging, boosting, and voting, are applied to improve learners' performance (Song et al. 2019; Freund and Schapire 1997; Jo and Japkowicz 2004, 2004). However, the minority class's small disjunct (Rathore and Kumar 2019; Li and Henry 1993) is also equally responsible for the poor performance of SDP models. Minority class samples are widely dispersed in the distribution space. When modifying sampling methods, researchers consider the single cluster approach, where all minority samples are a part of one cluster, which cause overgeneralization due to class overlapping.

Initial research has been done to comprehend and assess how well different metrics measure fault proneness (Briand et al. 2001; Ohlsson et al. 1998; Menzies et al. Jan. 2007; Gray et al. 2010). Li and Henry (1993) conducted a study to analyse the relationship between the Object-oriented metrics proposed CK using multiple linear regression and state that there exist and strong relation between the proposed metrics and effort towards maintenance activity. Briand (2001) validated Coupling and Cohesion's usefulness in fault prediction and proved they are strong candidates for fault predictions. As the new metrics are introduced, the researchers focus on their ability towards fault proneness. Static code metrics have been found to be highly correlated with the defect which can be further investigated by various researchers (Menzies et al. Jan. 2007; Gray et al. 2010).

Japkowicz and Stephen (2002) studied the class imbalance problem and discussed two major sampling techniques to counter the imbalance issues: oversampling and under-sampling. Oversampling approach focuses on increasing the number of samples in the minority class by random duplication until the desired level of balance is attained, which will result in overfitting due to the repetition of samples. To compensate for the randomness, an information-based approach is proposed where the samples' closeness is considered for duplication. Second technique discussed was random under-sampling, which will eliminate the samples from majority class until the desired level of balance is attained, resulting in poor performance due to loss of information. Information-based under-sampling techniques focus on the samples far away from boundaries and are considered for elimination.

Chawla et al. (2002) proposed a novel oversampling technique called SMOTE which operates on the feature space to generate synthetic samples by choosing each minority sample and introducing synthetic samples along the line, joining

any/all minority samples. Depending on the level of balance required the synthetic samples are chosen. Since, samples along the line are chosen will results in overlapping of decision boundary and results in overgeneralization. Different variation of SMOTE such as BSMOTE (Han et al. 2005) which focus on the samples lies on the decision boundary, resulting in more dense samples near the decision boundary. Safe-Level SMOTE (Bunkhumpornpat et al. 2009) is quite similar to SMOTE, but all synthetic samples are introduced along the safe line.

MW-SMOTE (Barua et al. 2014) introduces weight for the instances which are hard to classify and focuses on more challenging samples alone. ADASYN (He et al. 2008) is similar to MW-SMOTE, which assigns weight to instances based on their complexity to classify. Kamthorn Puntumapon et al. (2016) proposed cluster-based minority oversampling, which uses TRIM criteria to identify all minority class regions to form clusters and combine multiple small clusters to form larger clusters. Ebo Bennin et al. (2017) proposed a diversity-based oversampling technique that introduces synthetic samples using diversity measure, i.e., Mahalanobis distance, by averaging two minority samples.

2 Method

2.1 Overview

The majority of oversampling approach doesn't consider the distribution of minority samples and consider all the instance of the minority class of a single cluster. Since the minority samples are more dispersed in reality, generated synthetic samples may overlap with representatives from the majority or may manage to evade the minority decision boundary, which leads to overgeneralization. The poor performance of the SDP model is equally attributable to smaller disjuncts of minority instances as it is to the class imbalance. By considering the distribution of minority data samples and partitioning them into multiple clusters, we can eliminate the possibility of overlapping.

2.2 Diversity measurement

Researchers have often opted for distance metrics such as Hamming, Manhattan, Minkowski, Euclidean, and Cosine distance is a mere statistical distance which computes the difference between the two points and introduces synthetic samples between them. Most statistical measures don't consider the sample distribution and suffer when applied to high-dimensional data. To alleviate the issue, we opted for Mahalanobis distance (D). This diversity-based measure computes how far the individual samples is diverse from the population already experimented with outlier detection. By considering $\times 1$ and $\times 2$ as the two instances of the minority class the Mahalanobis distance is defined by 1.

$$D^2 = (x_1 - y_1)^t * C^{-1} * (x_1 - y_1) \tag{1}$$

where D^2 is the square of the Mahalanobis distance, $\times 1$ is the observation vector, i.e., the data set row, and C^{-1} is the inverse of the covariance matrix of independent variables.

The covariance matrix of the vector with n dimension is computed using

$$C = \begin{matrix} var(n1) & cov(n1,n2)\ldots & cov(n1,nn) \\ cov(n1,n2) & var(n2)\ldots & cov(n2.nn) \\ cov(n1,nn) & cov(n2,nn)\ldots & var(nn) \end{matrix}$$

Variance (v) computes the diversity of the data samples over its mean, and covariance (c) measures the relationship between the variables, which aids in measuring the diversity of the samples from its population given by 2.

$$\mathrm{V} = \frac{\sum_1^n (xi - \mu)^2}{n} \tag{2}$$

where xi refers to the column vector, μ is the mean of the column, and n is the number of rows. Similarly, covariance is computed using 3

$$c(x,y) = \frac{\sum_1^n (xi - \mu x)(yi - \mu y)}{n} \tag{3}$$

2.3 Clustering of minority samples

Based on the computed Mahalanobis distance matrix, we grouped the minority cluster into multiple clusters based on Calinski-Harabasz (CH) index and Davies-Bouldin (DB) index.Both these metrics helps to identify the correct number of clusters based on the compactness and cluster separation factor from 4 and 5.

$$\mathrm{CH_k} = \frac{BCSM}{k-1} * \frac{n-k}{WCSM} \tag{4}$$

where k is the number of clusters, n is the number of minority samples, BCSM measures of separation between cluster and WCSM is the measure of cluster compactness.

$$\mathrm{DB} = \frac{1}{n}\sum_{i=1}^{n} max_{\mathrm{j}\neq\mathrm{i}}\left(\frac{\sigma i + \sigma j}{d(ci,cj)}\right) \tag{5}$$

where n is the number of cluster and σi is the average distance of all vectors in cluster i from its cluster center ci. We compute the number of clusters (nc) by finding the minimum of both the index CH and CB using 6

$$\mathrm{nc} = \min(\mathrm{CH}, \mathrm{DB}) \tag{6}$$

2.4 Synthetic sample generation

Based on the computed diversity measure (MD) the instances are grouped into multiple clusters. The feature vector, made

up of metrics components extracted from software artifacts, is thought of as chromosomes per Walter's theory of inheritance.

Algorithm: Multipatch cluster based Oversampling algorithm

Input: Software Defect Dataset of N, Pfp

Output: DataSet sampled at given Pfp Value

Procedure:

1) *Partition the Data set into groups of Minority class Nmin and Majority class Nmax*
2) *Calculate how many synthetic samples must be produced for the specified Pfp value as T*
3) *Compute the ideal number of clusters (nc) using CH and DB index*
4) *Set the counter Ccnt to 0 and create a new array Xnew for the synthetic samples.*
5) *Ccnt keeps the count of the synthetic sample generated*
6) *Calculate the Mahalanobis Distance(D) for Nmin D= prod(X_mu,inv_cov) where X_mu =* $\sum(x - mean(x))$ *inv_cov = inv(cov)*
7) *Save the Distance(D) matrix and combine it with the corresponding samples of data in an array (Nmd)*
8) *Cluster the D matrix into nc number of clusters based on the computed D value*
    a. *Arbitrarily choose nc data point from D matrix as centroids*
    b. *Repeat*
        i. *Assign each datapoint from D to one the cluster which is nearest to the centroid*
        ii. *Calculate the new mean and assign as centroid for each cluster*

    Until converge criteria met.
9) *For each cluster c in nc*
    a. *Sort the datapoints in the individual cluster C in reducing order*
    b. *Compute the midpoint of the Distance in the cluster*
    c. *Make two bins from C with the last point of partition 1 being midpoint and the starting point of partition 2 being midpoint+1.For i =1 to mid*
        a. *Select parent p1i in partition 1 and p2i in partition2*
        b. *Generate a minority class synthetic sample x by average p1 and p2*
        c. *Increase the value of Ccnt in relation to the number of newly synthesised children*

    end for
10) *end for*
11) *if Ccnt < T, repeat the steps from 10 to 11 to generate the new synthetic samples to arrive at the required Pfp value. If still Ccnt < T pair each new synthetic child with its immediate parent and repeat the step 10 to 11 to generate synthetic sample to balance the set.*
12) *If Ccnt > T, assign all the new childs with the label of minority class and add the result to the Nmin*

End

By computing the midpoint of cluster, the data's partitioned into two subgroups where g1 represents the instance above the center and g2 represents the instance below the mean. The parents are indexed from 1 to n in groups g1 and g2. Considering p1 as the parent1 from g1 with index1 and p2 as the parent2 from g2 induced into sampling process which yields new offspring. Considering that the samples are far apart from each other eliminates the possibility of duplication and alleviates the over-generalization by introducing a more diverse sample. The process of synthetic generation is illustrated in Fig. 2 which is repeated until the desired level balance is attained.

The objective of the proposed methodology is to improve the performance of SDP model by reducing the false alarm rate and reduce overgeneralization. To justify the performance a comparative experiment is conducted with five other prominent oversampling approaches such as SMOTE, B-SMOTE, ADASYN, S and MAHAKIL using five different classification models such as Naive Bayesian, Decision Tree, KNN, SVM and Random Forest on 20 different imbalanced datasets obtained from promise repository with varied imbalance ration from 2 to 14%. The datasets chosen to conduct a comprehensive investigation vary in terms of the imbalance ratio and sample size. Figure 3 shows the experimental setup to validate the performance of the proposed over-sampling approach.

The dataset comprises of 20 features which is indicated the various measure of the software component measured from modules or classes of those projects. The class labels are converted to binary values of 0 and 1 using label encoding, where 0 represent the negative instance and 1 represents the positive instance. Encoded dataset is separated into two bins where bin1 represent the positive samples and bin2 represent the negative samples. The positive samples are provided to the oversampling module to add synthetics samples using the proposed approach. The oversampled dataset is divided into ten folds, with the first eight folds being used for training, the ninth for model validation, and the tenth for model testing. The division is accomplished through stratification, so that the proportion of minority class distribution, or pfp, remains constant across both the training and testing datasets. The sampled dataset is trained on five different machine learning models such as Naive Bayesian, Decision Tree, KNN, SVM and Random Forest and the performance of the models are evaluated using different measures.

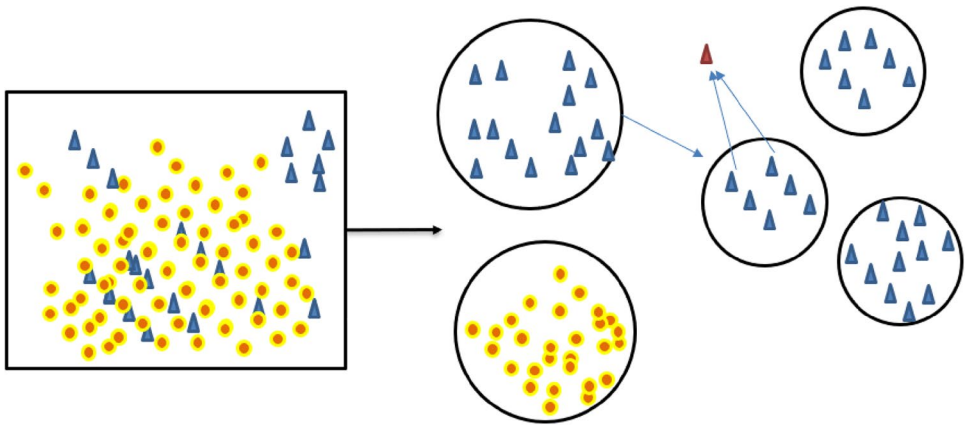


Fig. 2 Illustration of synthetic sample generation

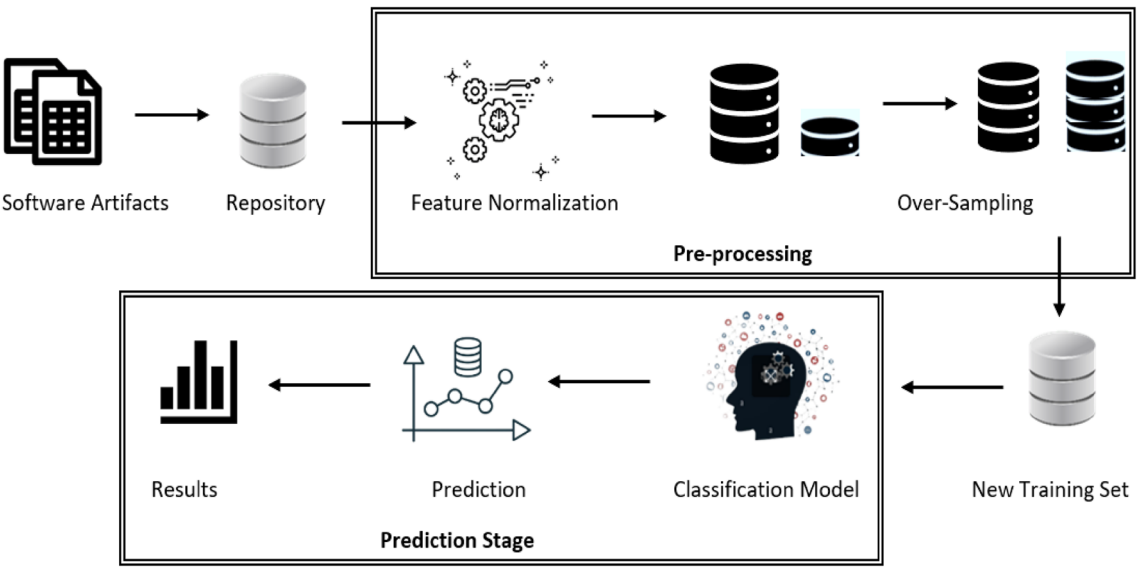


Fig. 3 Framework of proposed model

3 Results and discussion

The SDP is a two-class problem, where the samples of interest are in the minority class or defective and are regarded as positive. In contrast, the majority class is viewed as unfavorable. According to the confusion matrix, the terms True Positive (TP) and False Positive (FP) refer to the number of defective samples that are classified as defective and non-defective, respectively similarly, False Negative (FN) and True Negative (TN), which refers to the number of non-defective samples that are classified as defective and non-defective respectively. The Performance of the machine learning models is evaluated using measures such as accuracy, precision, recall, F1-Score, AUC, falsealarm and so on. Since the model is trained on imbalance data, accuracy has proved to be biased.

Similarly, precision is also unstable for imbalanced data (Shihab 2012). So, we opted for recall (pd) and false-alarm (pf) rate to evaluate the performance. Recall estimates the number of false negatives and false-alarm is closely correlated with defective instance identification. We consider those two measures to assess the performance (Table 1).

$$\text{Recall (pd)} = \text{TP}/(\text{TP} + \text{FN}) \tag{7}$$

$$\text{False Alarm (pf)} = \text{FP} / (\text{TN} + \text{FP}) \tag{8}$$

3.1 Comparison with other oversampling techniques

To validate the performance of the proposed oversampling, we conducted the experiment using five different oversampling approaches SMOTE, BSMOTE, ADASYN, MAHAKIL, and ROS using the sample five machine learning models and the across all the 20 different datasets in Table 2. The Proposed sampling approach outperforms other oversampling approaches in terms of reduced false alarm rate in all three-balance ratios. The proposed method when tested with RF provides better performance in terms of recall and false alarm rate.

The experiment was conducted using five different oversampling approaches on three different balance ratios across 20 different datasets and evaluated using five machine learning models. From the above figure KNN and RF consistently outperform all other models concerning recall and false alarm rate. Also, it's evident from the Fig. 4 that NB and SVM has no effect on balancing i.e., even after balancing the performance of the model doesn't have any effect. They are increasing Pfp to 50% for all sampling methods resulting in the best pd performance across all prediction models and datasets.

Across all models, proposed approach, MAHAKIL and SMOTE has consistently performs better recall (pd) and false-alarm rate (pf). Proposed approach performed considerably better with high pd and low pf values across all models and datasets. This is because the more diverse nature of synthetic samples also ensures the samples reside intact within the decision boundary, eliminating the possibility of misclassification.

The outcomes demonstrate that the proposed model outperforms all five sampling strategies. According to Lessman et al. research's (2008) RF significantly outperformed the other 21 prediction models, which is evident from these results. RF is more resilient to overfitting and parameterization as an ensemble method. Our proposed approach effectively splits the minority samples into multiple clusters based on the computed diversity measure which also considers the samples original distribution. Also, the diversity-based measure ensures the synthetic samples are more diverse, eliminating duplication of samples.

Oversampling, which entails producing additional synthetic instances of the minority class to balance it with the majority class, is a popular strategy for dealing with unbalanced datasets. Several algorithms have been developed to perform oversampling effectively, and in this content, we will compare the distribution of data samples after applying two cluster based oversampling algorithms: KMeans-SMOTE and MBOA algorithm along with Initial distribution. Figure 5 depicts the distribution of data samples of an imbalanced dataset before and after sampling.

In comparing the distribution of data sample after oversampling using the two cluster-based algorithms with original distribution, we observe the following:

KMeansSMOTE tends to create synthetic samples that are distributed around the clusters identified by the K-means algorithm. It creates synthetic samples around the borderline of cluster so the data samples are more clustered around the boundary region of the cluster. Hence, the generated synthetic samples are highly deviated from the original distribution which leads to poor performance.

The proposed multi-cluster based oversampling approach tends to create synthetic samples which closely matches the original distribution since, the synthetic samples are created in each cluster based on the original cluster density. To enhance diversity in the synthetic samples, it then applies crossover to create new combinations of synthetic samples within clusters. The resulting distribution is more diverse, with synthetic samples that represent a broader range of characteristics of the minority class which results in better performance of the prediction models.

Also, we performance quantitative analysis on the performance of both cluster-based oversampling approach using XGBoost ensemble model. Performance of XGBoost model on five different imbalance datasets with varied imbalance ratio in terms of recall and false-alarm

Table 1 Performance comparison proposed model

Dataset	Model	30% balance		40% balance		50% balance	
		Recall	False alarm	Recall	False alarm	Recall	False alarm
Ant 1.4	NB	0.250	0.171	0.250	0.147	0.250	0.136
	KNN	0.250	0.107	0.250	0.107	0.375	0.143
	DT	0.375	0.143	0.375	0.136	0.375	0.136
	SVM	0.125	0.171	0.125	0.182	0.125	0.143
	RF	0.250	0.086	0.375	0.071	0.375	0.104
Ant 1.6	NB	0.316	0.154	0.211	0.154	0.211	0.154
	KNN	0.579	0.135	0.579	0.154	0.579	0.154
	DT	0.579	0.231	0.474	0.173	0.474	0.173
	SVM	0.474	0.135	0.477	0.185	0.474	0.135
	RF	0.632	0.096	0.632	0.115	0.684	0.115
Arc	NB	0.237	0.132	0.211	0.130	0.210	0.105
	KNN	0.289	0.128	0.216	0.173	0.316	0.173
	DT	0.316	0.212	0.229	0.192	0.237	0.212
	SVM	0.105	0.019	0.105	0.192	0.212	0.192
	RF	0.389	0.090	0.240	0.109	0.237	0.103
Camel 1.6	NB	0.237	0.171	0.211	0.138	0.211	0.138
	KNN	0.237	0.141	0.263	0.141	0.263	0.141
	DT	0.395	0.205	0.395	0.154	0.289	0.141
	SVM	0.105	0.226	0.105	0.126	0.105	0.126
	RF	0.407	0.083	0.263	0.083	0.263	0.103
Ivy 2.0	NB	0.624	0.148	0.250	0.132	0.250	0.132
	KNN	0.750	0.127	0.429	0.127	0.750	0.127
	DT	0.625	0.159	0.375	0.148	0.500	0.111
	SVM	0.500	0.132	0.500	0.132	0.500	0.132
	RF	0.660	0.097	0.666	0.088	0.625	0.077
Jedit-4.1	NB	0.313	0.129	0.250	0.164	0.250	0.164
	KNN	0.463	0.185	0.563	0.116	0.563	0.158
	DT	0.375	0.128	0.438	0.159	0.438	0.158
	SVM	0.438	0.143	0.500	0.142	0.500	0.143
	RF	0.500	0.064	0.500	0.098	0.500	0.098
Jedit-4.2	NB	0.100	0.131	0.250	0.133	0.200	0.178
	KNN	0.200	0.078	0.286	0.126	0.200	0.178
	DT	0.200	0.094	0.300	0.138	0.300	0.178
	SVM	0.200	0.137	0.200	0.133	0.200	0.131
	RF	0.300	0.047	0.300	0.125	0.300	0.047
Log4j-1.0	NB	0.286	0.148	0.129	0.172	0.125	0.190
	KNN	0.483	0.148	0.143	0.188	0.143	0.141
	DT	0.429	0.165	0.429	0.167	0.429	0.141
	SVM	0.429	0.188	0.429	0.175	0.439	0.170
	RF	0.489	0.048	0.429	0.092	0.429	0.107
Pbeans2	NB	0.325	0.145	0.300	0.222	0.375	0.220
	KNN	0.430	0.157	0.500	0.222	0.500	0.222
	DT	0.350	0.222	0.333	0.222	0.500	0.222
	SVM	0.350	0.283	0.500	0.287	0.500	0.222
	RF	0.500	0.111	0.500	0.111	0.500	0.111

Table 1 (continued)

Dataset	Model	30% balance		40% balance		50% balance	
		Recall	False alarm	Recall	False alarm	Recall	False alarm
Redaktor	NB	0.190	0.287	0.197	0.183	0.197	0.183
	KNN	0.167	0.239	0.167	0.139	0.167	0.139
	DT	0.167	0.182	0.167	0.133	0.167	0.133
	SVM	0.333	0.199	0.333	0.133	0.333	0.133
	RF	0.349	0.133	0.366	0.099	0.396	0.103
Synapse-1.0	NB	0.500	0.134	0.500	0.134	0.250	0.134
	KNN	0.500	0.169	0.500	0.169	0.500	0.167
	DT	0.250	0.138	0.500	0.135	0.250	0.103
	SVM	0.250	0.135	0.250	0.134	0.250	0.134
	RF	0.500	0.090	0.500	0.069	0.500	0.069
Tomcat	NB	0.250	0.145	0.063	0.145	0.063	0.138
	KNN	0.250	0.176	0.250	0.176	0.250	0.128
	DT	0.313	0.170	0.313	0.196	0.313	0.137
	SVM	0.350	0.197	0.063	0.186	0.063	0.145
	RF	0.500	0.089	0.363	0.105	0.500	0.105

rate are shown in Table 3. The proposed multi-cluster based oversampling approach outperforms the KMeans-SMOTE cluster algorithm in both the measures because of the diversity of synthetic samples.

4 Threats to validity

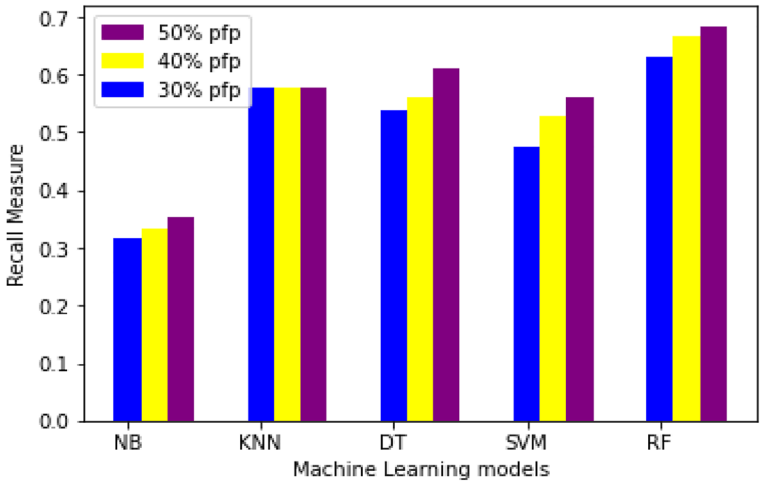
The Outcome of the proposed model depends on the Mahalanobis distance. The computation of MD for sizeable dimensional data is challenging because of the covariance matrices. Also, when the features are closely related to each other, covariance calculation is impossible because of high correlation. Additionally, covariance matrices for datasets that contain fewer minority samples than the features is an impossible task which is compensated using the generalized inverse method. To obtain more stable performance irrespective of the imbalance ratio generalized metric can be adopted. Also, the model's performance has not been tested across cross-project defect prediction.

5 Conclusion

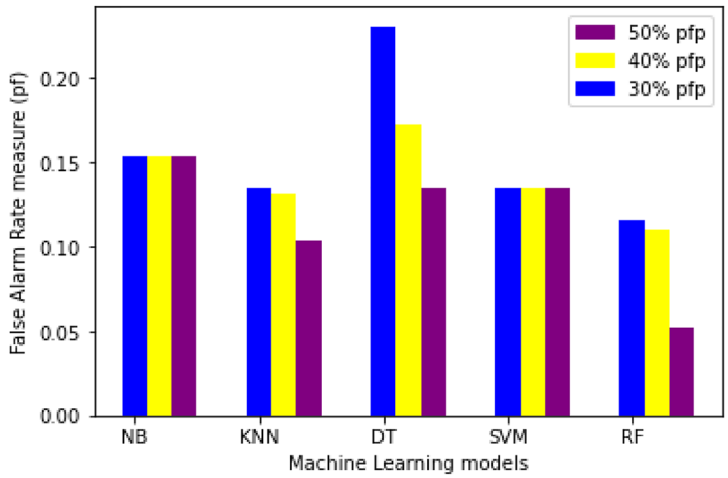
The proposed approach improves the performance of SDP by generating more diverse synthetics sample also ensure the samples within decision boundaries. It outperforms other over-sampling approachesconcerning false alarm rate (pf) and recall (pd) measures. But, in cross project defect prediction domain where the datasets are readily not available, finding project with similar characteristics and domain to perform transfer learning is yet another challenging task. The performance of the proposed model has been tested using five different models across 20 different datasets was superior to other over-sampling approaches.

Table 2 Performance comparison of different oversampling techniques

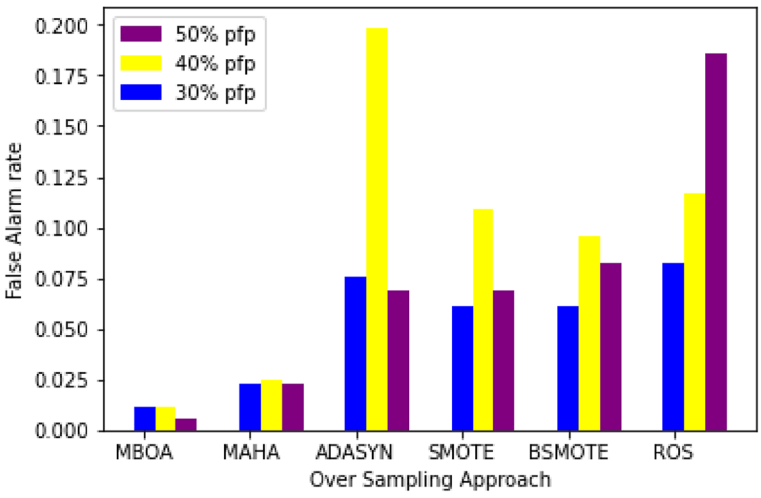
Dataset	Sampling Algo	30%		40%		50%	
		pd	pf	pd	pf	pd	pf
Ant 1.4	Multi-Clu	0.632	0.096	0.632	0.115	0.684	0.115
	MAHA	0.125	0.125	0.505	0.129	0.505	0.125
	ADASYN	0.500	0.143	0.613	0.132	0.613	0.130
	SMOTE	0.465	0.171	0.471	0.143	0.471	0.143
	B-SMOTE	0.412	0.171	0.471	0.143	0.471	0.143
	ROS	0.250	0.175	0.575	0.159	0.575	0.150
Arc	Multi-Clu	0.389	0.090	0.240	0.109	0.237	0.103
	MAHA	0.289	0.109	0.230	0.121	0.109	0.108
	ADASYN	0.250	0.179	0.222	0.178	0.179	0.179
	SMOTE	0.250	0.147	0.220	0.128	0.147	0.173
	B-SMOTE	0.250	0.160	0.200	0.122	0.160	0.179
	ROS	0.200	0.147	0.220	0.178	0.147	0.171
Camel 1.6	Multi-Clu	0.407	0.083	0.263	0.083	0.263	0.080
	MAHA	0.263	0.096	0.250	0.096	0.250	0.087
	ADASYN	0.249	0.173	0.250	0.200	0.222	0.186
	SMOTE	0.279	0.154	0.222	0.128	0.220	0.153
	B-SMOTE	0.289	0.135	0.220	0.135	0.200	0.141
	ROS	0.250	0.128	0.111	0.141	0.200	0.131
Ivy 2.0	Multi-Clu	0.660	0.097	0.666	0.088	0.625	0.077
	MAHA	0.625	0.048	0.500	0.095	0.500	0.110
	ADASYN	0.500	0.095	0.500	0.149	0.500	0.127
	SMOTE	0.500	0.095	0.330	0.159	0.465	0.159
	B-SMOTE	0.500	0.095	0.330	0.095	0.465	0.152
	ROS	0.500	0.095	0.330	0.158	0.400	0.213
Jedit-4.1	Multi-Clu	0.500	0.064	0.500	0.098	0.500	0.098
	MAHA	0.464	0.064	0.500	0.108	0.500	0.108
	ADASYN	0.451	0.085	0.470	0.109	0.400	0.109
	SMOTE	0.511	0.085	0.300	0.149	0.400	0.149
	B-SMOTE	0.411	0.085	0.300	0.144	0.350	0.144
	ROS	0.438	0.085	0.300	0.199	0.358	0.199
Synapse-1.0	Multi-Clu	0.500	0.090	0.500	0.069	0.500	0.069
	MAHA	0.500	0.090	0.500	0.109	0.500	0.108
	ADASYN	0.400	0.103	0.475	0.093	0.475	0.100
	SMOTE	0.435	0.345	0.375	0.248	0.375	0.200
	B-SMOTE	0.375	0.345	0.375	0.233	0.375	0.249
	ROS	0.300	0.403	0.400	0.303	0.400	0.299
Tomcat	Multi-Clu	0.500	0.089	0.363	0.105	0.500	0.105
	MAHA	0.500	0.106	0.448	0.089	0.500	0.089
	ADASYN	0.409	0.162	0.400	0.229	0.300	0.229
	SMOTE	0.489	0.125	0.500	0.108	0.300	0.108
	B-SMOTE	0.409	0.157	0.484	0.178	0.186	0.178
	ROS	0.409	0.191	0.400	0.038	0.186	0.138



(a)

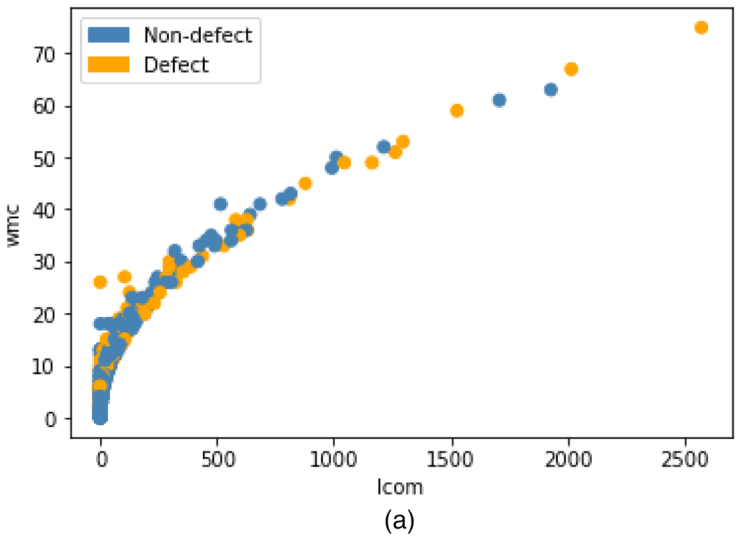


(b)

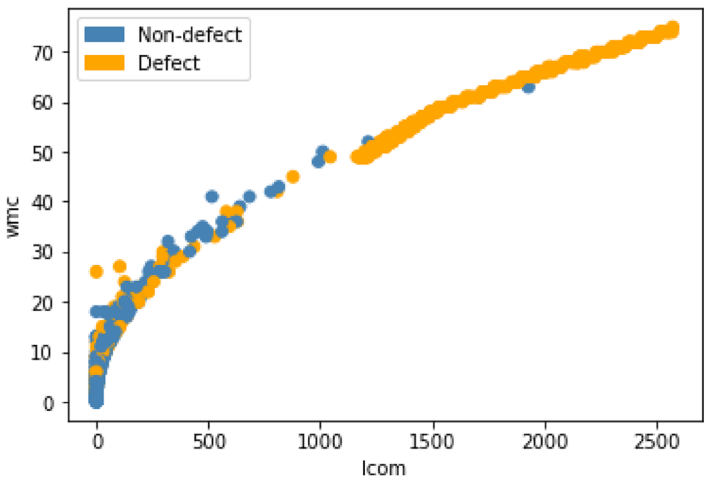


(c)

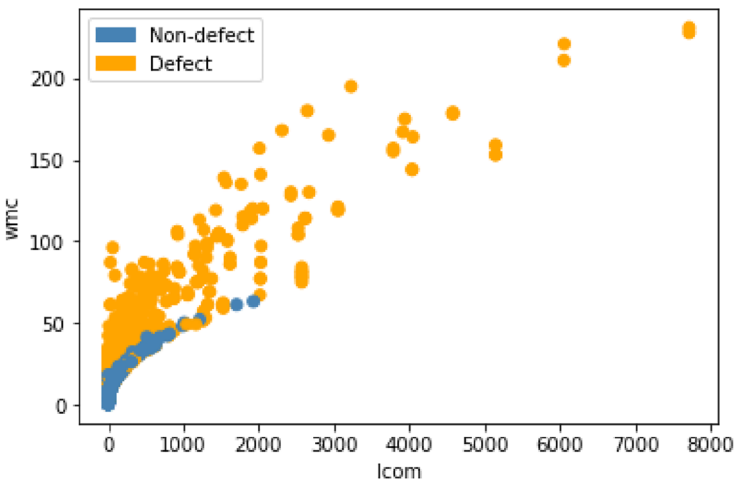
Fig. 4 **a** Recall measure comparison proposed model using different machine learning algorithms. **b** False alarm rate measure comparison proposed model using different machine learning algorithms. **c** False alarm measure comparison different over sampling approaches using Random Forest algorithm



(a)



(b)



(c)

Fig. 5 **a** Distribution of data sample without sampling. **b** Distribution of data samples after sampling using KMeans-SMOTE. **c** Distribution of data samples after sampling using MBOA

Table 3 Performance comparison of cluster-based oversampling approach using XGBoost classifier

Dataset	Recall (pd)		False alarm rate (pf)	
	KmeansS-MOTE (%)	MBOA (%)	KmeansS-MOTE (%)	MBOA (%)
ant-1.6	61	71.5	22.9	12.5
camel-1.6	63	69	18.1	5.2
jedit-4.1	69	72	16.21	6.38
Tomcat	68	79	19.1	10.09
xerces-1.3	68	78	18	3.89

Acknowledgment No acknowledge

Author contributions Authors Contributed equally.

Funding No funding is involved in this work.

Data availability Data available on request from the authors.

Declarations

Conflict of interest Conflict of Interest is not applicable in this work.

Ethical approval and consent to participate No participation of humans takes place in this implementation process.

Human and animal rights No violation of Human and Animal Rights is involved.

References

Barua S, Islam MM, Yao X, Murase K (2014) MWMOTE– majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans Knowl Data Eng 26(2):405–425

Briand L, Wst J, Lounis H (2001) Replicated case studies for investigating quality factors in object-oriented designs. Empir Softw Eng Int J 1:11–58

Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2009) Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. Advances in knowledge discovery and data mining. PAKDD 2009. Lecture notes in computer science, vol 5476. Springer, Berlin, Heidelberg

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Diez-Pastor JF, Rodriguez JJ, Garcia-Osorio C, Kuncheva LI (2015) Random balance: ensembles of variable priors classifiers for imbalanced data. Knowl-Based Syst 85:96–111

Ebo Bennin K, Keung J, Phannachitta P, Monden A, Mensah S (2017) MAHAKIL: diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. IEEE Trans Softw Eng 44:534–550

Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139

Gao K, Khoshgoftaar T, Napolitano A (2014) The use of ensemble-based data preprocessing techniques for software defect prediction. Int J Software Eng Knowl Eng 24:1229–1253

Gray D, Bowes D, Davey N, Sun Y, Christianson B (2010) Software defect prediction using static code metrics underestimates defect-proneness. Int Jt Conf Neural Netw (IJCNN) 2010:1–7. https://doi.org/10.1109/IJCNN.2010.5596650

Guo H, Viktor HL (2004) Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. SIGKDD Explor Newsl 6(1):30–39

Hall T, Beecham S, Bowes D, Gray D, Counsell S (2011) A systematic literature review on fault prediction performance in software engineering. IEEE Trans Softw Eng 38(6):1276–1304

Han H, Wang W-Y, Mao B-H, (2005) Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: International conference on intelligent computing, pp 878–887

He H et al (2008) Adasyn: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), pp 1322–1328

Japkowicz N, Stephen S (2002) The class imbalance problem: a systematic study. Intell Data Anal 6:429–449. https://doi.org/10.3233/IDA-2002-6504

Jo T, Japkowicz N (2004) Class imbalances versus small disjuncts. ACM SIGKDD Explor Newsl 6(1):40–49

Jones C, Bonsignour O (2012) The economics of software quality. Pearson Education, Inc

Kamei Y et al (2012) A large-scale empirical study of just-in-time quality assurance. IEEE Trans Softw Eng 39(6):757–773

Lessmann S, Baesens B, Mues C, Pietsch S (2008) Benchmarking classification models for software defect prediction: a proposed framework and novel findings. IEEE Trans Softw Eng 34(4):485–496

Li W, Henry S (1993) Object-oriented metrics that predict maintainability. J Syst Softw 23(2):111–122

Menzies T, Greenwald J, Frank A (2007) Data mining static code attributes to learn defect predictors. IEEE Trans Softw Eng 33(1):2–13

Ohlsson N, Zhao M, Helander M (1998) Application of multivariate analysis for software fault prediction. Softw Qual J 7(1):51–66

Ozturk MM (2017) Which type of metrics are useful to deal with class imbalance in software defect prediction? Inf Softw Technol 92:17–29

Pandey SK, Mishra RB, Tripathi AK (2021) Machine learning based methods for software fault prediction: a survey. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2021.114595

Prati RC, Batista GE, Monard MC (2004) Learning with class skews and small disjuncts. In: Brazilian symposium on artificial intelligence. Springer, Berlin, Heidelberg. pp 296–306

Provost F (2000) Machine learning from imbalanced data sets 101. In: Proceedings of the AAAI' workshop imbalanced data sets, pp 1–3

Puntumapon K, Rakthamamon T, Waiyamai K (2016) Cluster-based minority over-sampling for imbalanced datasets. IEICE Trans Inf Syst E99.D(12):3101–3109

Rathore SS, Kumar S (2019) A study on software fault prediction techniques. Artif Intell Rev 51(2):255–327

Schneider GM, Martin J, Tsai W-T (1992) An experimental study of fault detection in user requirements documents. ACM Trans Softw Eng Methodol (TOSEM) 1(2):188–204

Shihab E (2012) An exploration of challenges limiting pragmatic software defect prediction. PhD thesis

Song Q, Guo Y, Shepperd M (2019) A comprehensive investigation of the role of imbalanced learning for software defect prediction. IEEE Trans Softw Eng 45(12):1253–1269

Sri Kavya K (2020) An ensemble deepboost classifier for software defect prediction. Int J Adv Trends Comput Sci Eng 9:2021–2028

Thabtah F, Hammoud S, Kamalov F, Gonsalves A (2020) Data imbalance in classification: experimental evaluation. Inf Sci 513:429–441

Weiss GM, Provost F (2001) The effect of class distribution on classifier learning: an empirical study. Rutgers University

Yoon K, Kwek S (2007) A data reduction approach for resolving the imbalanced data issue in functional genomics. Neural Comput Appl 16(3):295–306

Zhang X, Song Q, Wang G, Zhang K, He L, Jia X (2015) A dissimilarity-based imbalance data classification algorithm. Appl Intell 42:544–565