MAHAKIL: Diversity Based Oversampling Approach to Alleviate the Class Imbalance Issue in Software Defect Prediction

Kwabena Ebo Bennin, *Student Member, IEEE*, Jacky Keung, *Member, IEEE*, Passakorn Phannachitta, Akito Monden, *Member, IEEE*, and Solomon Mensah

Abstract—Highly imbalanced data typically make accurate predictions difficult. Unfortunately, software defect datasets tend to have fewer defective modules than non-defective modules. Synthetic oversampling approaches address this concern by creating new minority defective modules to balance the class distribution before a model is trained. Notwithstanding the successes achieved by these approaches, they mostly result in over-generalization (high rates of false alarms) and generate near-duplicated data instances (less diverse data). In this study, we introduce MAHAKIL, a novel and efficient synthetic oversampling approach for software defect datasets that is based on the chromosomal theory of inheritance. Exploiting this theory, MAHAKIL interprets two distinct sub-classes as parents and generates a new instance that inherits different traits from each parent and contributes to the diversity within the data distribution. We extensively compare MAHAKIL with SMOTE, Borderline-SMOTE, ADASYN, Random Oversampling and the No sampling approach using 20 releases of defect datasets from the PROMISE repository and five prediction models. Our experiments indicate that MAHAKIL improves the prediction performance for all the models and achieves better and more significant *pf* values than the other oversampling approaches, based on Brunner's statistical significance test and Cliff's effect sizes. Therefore, MAHAKIL is strongly recommended as an efficient alternative for defect prediction models built on highly imbalanced datasets.

Index Terms—Software defect prediction, class imbalance learning, synthetic sample generation, data sampling methods, classification problems

1 Introduction

Software defect prediction models can conveniently identify software modules or classes that are more likely to be defective [1], [2]. Accurate prediction helps prevent defects by suggesting that developers focus more on these modules or classes to efficiently prioritize and allocate the limited testing resources to them. The accuracy of the prediction models are often hindered by the imbalanced nature of the datasets [3], [4], in which there are naturally fewer defective classes than the non-defective classes [1], [5], [6]. The main problem is that common prediction algorithms assume that the classes in any dataset are equally balanced [7], [8]. Thus, models trained on imbalanced software defect datasets are generally biased towards the non-defective class samples and ignore the defective class samples [9]. Consequently, the class imbalance issue is well-recognized as one of the major causes of the poor performance of software defect prediction models [10], [11]. Accordingly, a considerable body of research [5], [12], [13] has sought to alleviate this problem in the past decade.

The prevalent approach to solving the problem of class imbalance is to use data sampling techniques. To date, most of the data sampling approaches are synthetic based [14], [15], [16], [17]. The most popular among them being the Synthetic Minority Over-sampling Technique (SMOTE) [14], whereby new synthetic or artificial data samples are intelligently introduced into the minority class samples. These synthetic methods tend to introduce some bias towards the minority class, thus improving the performance of prediction models on the minority class.

Notwithstanding the positive effect that these synthetic methods have on classifiers, Barua et al. [17] argued that synthetic methods such as SMOTE and the Adaptive Synthetic Sampling Approach (ADASYN) [15] erroneously enlarge the minority class region. This leads to many majority class samples being misclassified by a classifier because synthetic samples are accidentally introduced into the region of the majority class. These approaches also generate data samples that are non-diverse and sometimes very similar to existing samples because they consider only nearest neighbor samples.

- *K. Ebo Bennin, J. Keung, and S. Mensah are with the Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong. E-mail: {kebennin2-c, smensah2-c}@my.cityu.edu.hk, Jacky.Keung@cityu.edu.hk.*
- *P. Phannachitta is with the College of Arts, Media and Technology, Chiang Mai University, Chiang Mai 50200, Thailand. E-mail: passakorn.p@cmu.ac.th.*
- *A. Monden is with the Graduate School of Natural Science and Technology, Okayama University, Okayama 700-0082, Japan. E-mail: monden@okayama-u.ac.jp.*

Manuscript received 1 Nov. 2016; revised 12 July 2017; accepted 19 July 2017. Date of publication 24 July 2017; date of current version 20 June 2018. (Corresponding author: Kwabena Ebo Bennin.) Recommended for acceptance by T. Zimmermann. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSE.2017.2731766

Similarly, Wong et al. [18] noted that SMOTE causes over-generalization and leads to a similar problem introduced by cross-company defect prediction which is mainly the increase in the false alarm rate (*pf*) [19] whereby non-defective modules are wrongly classified as defective. Our recent study [20] also confirmed the findings by Wong et al. [18]. A high *pf* implies that more effort is unnecessarily allocated to inspect those wrongly classified modules. In agreement with Barua et al. [17] and Wong et al. [18], Hang et al. [21] concluded with respect to anomaly detection that SMOTE increases *pf* despite the significant increase in classification accuracy. The problem of a high *pf* in highly imbalanced defect datasets was discussed in detail by Menzies et al. in [22]. Menzies et al. suggested high recall (*pd*) and low *pf* as the more stable performance measures for prediction models trained on highly imbalanced defect datasets. As such, data re-sampling approaches that result in high *pf* are undesirable.

In this study, we propose a novel resampling technique that achieves both a high *pd* and a low *pf* simultaneously. We believe that this can be achieved by generating as much diverse synthetic data as possible restricted within the region of the minority class. When the oversampling approach generates synthetic samples that are widely distributed but appropriately reside within the decision boundary or region of the minority class, the *pf* can be conveniently reduced without sacrificing the overall performance.

Our proposed oversampling technique takes inspiration from the field of biology by considering how diverse populations of humans or living organisms differ although they are of the same species. This phenomenon is explained by the Chromosomal Theory of Inheritance [23], which justifies how offspring inherit traits from their parents by obtaining chromosomes from each parent in equal quantity. The proposed technique named MAHAKIL (oversampling based on the theory of inheritance and the Mahalanobis distance), increases the diversity within the minority class by uniquely creating new synthetic minority instances based on a "typical" case (having a small diversity measure distance value) and an "atypical" case (having a large diversity measure distance value) so that the resultant instance becomes not too typical and not too atypical. The proposed technique offers several benefits over the other complex and conventional techniques. The over-generalization problem is alleviated by this technique because the newly generated samples are evenly distributed within the data and automatically induced to reside within the boundary and region of the minority class samples. Generating data instances from two dissimilar instances ensures that over fitting is avoided because this technique does not generate duplicated samples, thus in effect increasing the diversity within the resampled dataset.

We conducted empirical experiments to systematically show the performance of MAHAKIL in comparison to four common over-sampling approaches: Random Over-Sampling (ROS), SMOTE, Borderline-SMOTE and ADASYN using 20 imbalanced defect datasets from the PROMISE repository[1] and considering five classification algorithms evaluated on the resampled datasets. Based on robust Brunner statistical test and *win-tie-loss* statistics, our results show that MAHAKIL significantly improves the recall(*pd*) performance and reduces the *pf* for all five models over SMOTE, Borderline-SMOTE, ADASYN and ROS. The organization of this paper is as follows. Section 2 discusses some concerns over the typical sampling techniques and the motivations behind the MAHAKIL approach. Section 3 summarizes the chromosomal theory and its application to synthetic data generation. In Section 4, a detailed description of the MAHAKIL approach and its main components are presented. In Section 5, we describe the experimental methodology and performance evaluation measures used to validate our approach. The experimental results are presented in Sections 6 and 7 discusses the impacts of our findings, the limitations of our study and future research relating to MAHAKIL. Section 9 reviews the closely related research on imbalanced dataset and sampling techniques. Finally, a summary of the study is presented in Section 10.

1. http://openscience.us/repo/

2 Motivation

Synthetic oversampling methods such as SMOTE [14], Borderline-SMOTE [16] and ADASYN [15] share similar procedures for generating synthetic data. These methods use the k nearest neighbor (k-NN) technique to aid in selecting similar neighbors for oversampling. These techniques select the data instances that are the nearest neighbors or very close to the data instance under consideration with respect to their euclidean distance values. However, selecting only data instances that are very close (neighbors) poses the potential challenge of generating less diverse data points. For example, when $k = 1$, two very similar parents are selected, thereby generating data instances that are mere duplicates of the parents. This is almost the same as the ROS approach, which simply duplicates existing data instances and thus provides no new information to the classifier during training. The minority class boundary can also be accidentally widened by the conventional k-NN methods when data instances on the boundary are considered for oversampling. From Fig. 1b, most of the new data instances generated tend to fall outside the previous minority decision boundary and fall in the region of the majority class when data point C is considered.

Lastly, if the minority class samples comprise several sub-clusters of unequal size (intra-class imbalanced), generating synthetic data using the conventional k-NN methods such as SMOTE will only increase the size of the sub-clusters because each generated data instance will fall into a specified cluster. Therefore, these samples worsen the intra-class imbalanced problem. Fig. 1a displays an example of small and fairly dense subsets of samples within the minority class samples. If data points A and B are considered for the oversampling, all new generated data points will fall within their respective clusters if the synthetic methods that use k-NN methods are applied. The estimated decision boundary (dashed line) can be largely induced towards the dominating sub-clusters (i.e., cluster of A) and will not approximate as expected with the ideal or true decision boundary (solid line), as observed from Fig. 1a. Moreover, an "empty space" between the major clusters will not be filled with data because all new data samples generated fall within a cluster. The ideal resampled data are presented in Fig. 1c, where the newly generated synthetic data are

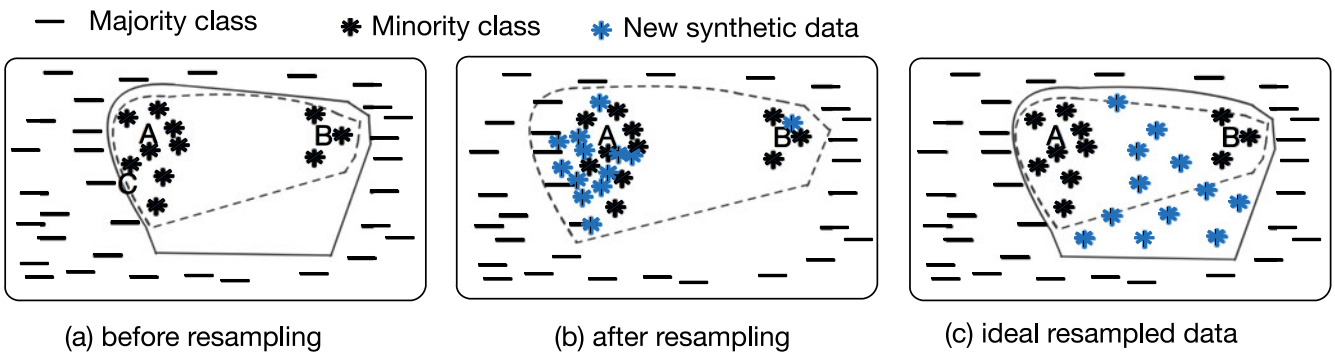

Fig. 1. Effect of sub-clusters of unequal or small sizes on the class imbalance problem; (a) the estimated decision boundary (dashed line) does not approximate the ideal or true decision boundary (solid line); (b) generating synthetic minority data without considering the position and distance of the nearest neighbors from the minority example (e.g., data point A) creates near duplicated or erroneous data and widens the estimated decision boundary; and (c) the ideal diverse data after resampling ensuring that the estimated and true decision boundaries are almost equal.

diversely spread and induce the estimated decision boundary to almost equal the true decision boundary.

These examples demonstrate that sampling methods based on the k-NN method are still deficient and are sometimes inappropriate for certain disparate scenarios of defect prediction. They can generate identical or duplicate data points and worsen the intra-class imbalanced issue. As noted by Barua et al. [17], these problems occur because these approaches select all k nearest neighbors with no regard to the position and distance of these neighbors from the data point under consideration. They thus generate erroneous samples and widen the region of the minority class. The determination of k in advance also poses a challenge for these methods.

3 Chromosomal Theory of Inheritance and Diversity

According to elementary biology and genetics, genes, the basic unit of heredity are located on the chromosomes and are passed on to children in equal amount with half from both parents during fertilization of the gametic (egg and sperm) chromosomes. This theory known as the Chromosomal Theory of Inheritance was proposed in 1902 by Walter Sutton and Theodor Boveri [23]. The theory implies that new offspring receive 50 percent of chromosomes from each parent, thereby making the new offspring similar to both parents and at the same time, unique from both parents. To maintain diversity within species, the sex(gender) of the species is of great importance because it helps in selecting two opposite members that can reproduce. The natural process of reproduction involving chromosomes can be defined as follows. Two different sets of chromosomes population $M = (m_1, m_2, m_3, \ldots\ldots\ldots m_q)$ and $F = (f_1, f_2, f_3, \ldots\ldots\ldots f_q)$ are created with the same size q. Each chromosome $c = (c_{j1}, c_{j2}, c_{j3}, \ldots\ldots\ldots c_{jp})$ represents a p-dimensional vector comprised of genes, where c_{jk} is the kth gene value $(k = 1, 2, \ldots, p)$ of the jth chromosome $(j = 1, 2, \ldots, q)$ in each of the populations. A new offspring is produced when two chromosomes, one from each set are combined with each parent randomly contributing half of the genes inherited by the offspring.

By elucidating the heredity law in animals and plants, researchers in the fields of biology and agriculture have extensively applied this theory and produced significant benefits. For example, improved methods for introducing new breeds of various plants and animals have been developed by agricultural specialists [24]. This theory has also provided valuable information about the DNA within a cell, hence facilitating the development of specific kinds of breeds in the domain of agriculture on a more rational basis whereby breeders, for example discard fish that are less productive and focus more on the productive animals [25], [26]. Farmers segregate animals and plants into groups based on their environmental and genetic features to develop (generate) specific traits in the plants and animals. Thus, the chromosomal theory of inheritance forms the basis of our understanding of the patterns of inheritance of many different traits in humans and other organisms.

The underlying principles behind selective animal and plant breeding are applied to an imbalanced defect dataset in order to generate (introduce) new data instances of faulty modules. During selective breeding, methods such as outcrossing, inbreeding and line-breeding [27] have been used to segregate the populations of animals and plants. Similarly, we can rationally oversample the faulty classes by considering a similarity measure to aid in segregating our data samples. Based on the chromosomal theory of inheritance and by adopting the segregation approach, we aim to develop an oversampling technique that can generate new samples by examining the minority defective samples and merging distinct samples to produce synthetic data instances that are maximally different from the parents and contribute to the diversity (variation) within the minority class distribution. The proposed oversampling algorithm based on inheritance theory is different from genetic algorithms because the software modules are considered to be individuals and not solutions to a specific problem. Furthermore, similar to other oversampling techniques, we do not want to lose the few minority data instances in the datasets after generating new data instances, which is in contrast to genetic algorithms whereby parents or older generations of parents die after generating new children data instances.

4 MAHAKIL - New Oversampling Technique for Software Defect Prediction

4.1 Overview

We exploit the challenge of the above mentioned synthetic methods in formulating our novel MAHAKIL oversampling

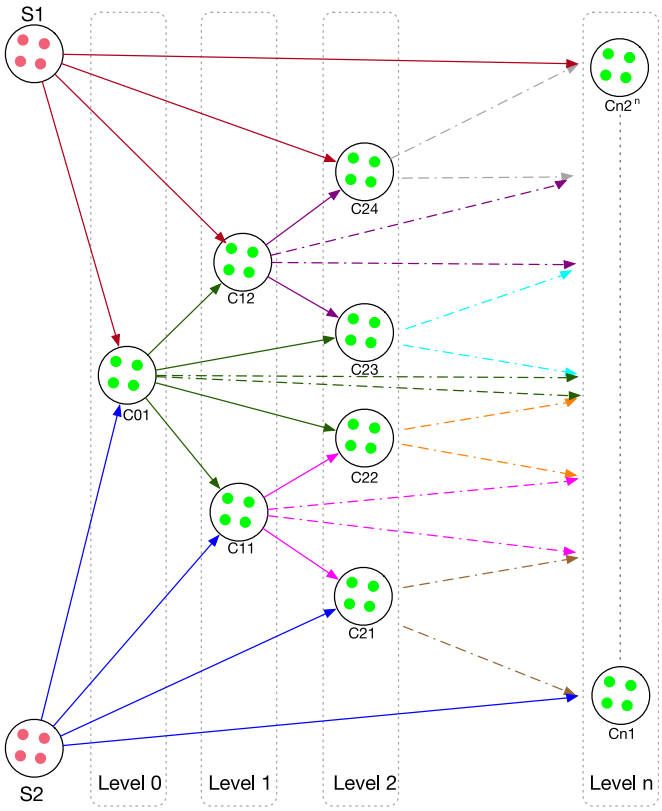

Fig. 2. Illustration of MAHAKIL. The red and green dots represent the minority samples with the latter representing the synthetic data generated by MAHAKIL.

method. We take inspiration from Walter Sutton's [23] chromosomal theory of inheritance by considering the features (metrics) of the defective modules as the chromosomes. The theory fits our objective of endeavoring to generate diverse data instances. We seek to generate new minority instances that take features from two different data instances but are also uniquely different and not a direct duplicate of the data instances involved. The basic intuition is that two different instances, which are not necessarily close in distance, can produce a new instance that is similar to both instances. MAHAKIL aims to generate a synthetic instance that has both peculiar and common attributes, and it involves three simple and fundamental phases.

In the first phase, the segregation measure values for the minority samples are computed after separating the minority samples from the majority samples. This involves computing of the Mahalanobis distance (D^2) for the minority samples and arranging the samples in a descending order according to their D^2 values. In the second phase, the ordered samples are partitioned into two bins by using the midpoint of the distance matrix (D^2) or central instance to split the samples and sequentially assign a unique label to each member in both bins. In the last phase, the synthetic sample is generated by finding the average of two selected data samples (samples with same unique labels) from each partition. Algorithm 1 presents the complete process, in which phase 1 is described in steps 1 to 6, the second phase is from steps 7 to 9 and concludes with steps 10 to 14 for the final stage. Each component or phase is described in the following sections.

Fig. 2 shows MAHAKIL in greater detail. The figure shows the complete generation of minority samples based on the number of samples required, which are represented as levels, and focuses on the minority class samples extracted from the whole data. In the first step, based on the Mahalanobis distance, the first and main parents (ancestors) are found and partitioned into two groups (node S1 and node S2) which are used to generate the set of new synthetic samples C01. To avoid new synthetic samples penetrating into the region of the majority class, the first parent nodes are used as the boundary such that all subsequent children generated automatically reside within the range of the parents. Each new child node with the help of its immediate parent, grand parent or ancestor is used to generate two or more extra nodes if more data instances are required. In step 2 (Level 0), the new synthetic node or children are included in the next process of data generation by pairing the children with the first node of parents (S1) and later with the second node (S2) to generate two new nodes of children. Consequently, in the next levels, the location of the children nodes is considered. New samples are generated by considering the immediate parent node of each children node. If the samples generated by pairing the children and immediate parent nodes are less than the required number of samples, the predecessor nodes of the parent nodes are paired with the current/children node. If the generated samples are still less than the required number of samples, the process of pairing the child nodes with older generations is repeated until the samples are sufficient (greater than or equal to the required number of samples). From level 1 onwards, the pairing is done using the sequential labels inherited from the immediate parents of the nodes and thus, the Mahalanobis distances are not computed for any two samples selected for data generation. Only steps 10-14 of Algorithm 1 are invoked for all levels starting from level 0.

4.2 Phase 1: Diversity Measurement

To define a diversity measure for a set of minority defective samples, we initially consider the diversity between the smallest sample set possible, which will be a set of two defective instances. To declare two p-dimensional feature vectors in the defective class as being the most diverse, the diversity value between them should be very large. We measure the diversity of two data instances using the Mahalanobis distance measure proposed by P. C. Mahalanobis [28]. This measure is adopted because it eliminates several limitations of the euclidean metric. Defect datasets are known to contain noise or duplicates [3] due to the collection process of the metrics. In this case, the use of the euclidean distance measure would fail to recognize the highly correlated or duplicated data samples that provide no new information during the classifier training. As a unit-less measure, the Mahalanobis distance measure provides a relative measure of an instance distance or residual from a common selected point and helps identify and detect the similarity between an unknown sample set and a known one, which also helps in detecting outliers. This measure is preferred because of its multivariate effect size, which considers the correlation of the dataset and is therefore, not scale dependent. The covariance among the data samples is also considered when calculating the distances thus, overcoming the inherent scale and correlation problems associated with the euclidean distance measure.

Considering two data instances or vectors of the defective class $x = (x_1, x_2, x_3, \ldots\ldots\ldots x_n)^T$ and $y = (y_1, y_2, y_3, \ldots\ldots\ldots y_n)^T$, the Mahalanobis distance between them is defined as:

$d(x, y) = \sqrt{(x-y)^T S^{-1}(x-y)}$, where S^{-1} is the covariance matrix

We use this measure to help rank and sort the data instances according to their distance in a decreasing order. By sorting the data, we are able to distinguish data samples that are far or close from the central data instance.

Algorithm 1. Overview of MAHAKIL(N, P)

Input: Dataset of minority and majority class samples N; *Pfp* value P

Output: Balanced dataset at a set *Pfp* value

Procedure Begin

1) Split dataset N into arrays of minority class N_{min} and majority class N_{maj}
2) Compute the number of additional minority classes to be generated (T) to attain *Pfp* P and $k = arraylength(N_{min})$
3) $Xnew$: array for generated sample, initialized to 0
4) X_{newchk}: keeps count of number of synthetic samples generated,
5) Compute Mahalanobis distance (D^2) for N_{min}, $D^2 = (x-\mu)^{'}\Sigma^{-1}(x-\mu)$ where μ=mean and Σ = covariance matrix
6) Save D^2 matrix and it's respective data samples in a decreasing order in $N_{mindist}$
7) Find the midpoint or middle instance, i.e., $N_{mid} = k/2$
8) Create two partitions from $N_{mindist}$ making N_{mid} the last record for partition 1 and N_{mid+1} the starting record for partition 2, i.e., $N_{bin1} = \{y_1, y_2 \ldots\ldots\ldots\ldots y_{mid}\}$ and $N_{bin2} = \{y_{mid+1}, y_{mid+2} \ldots\ldots\ldots\ldots y_k\}$, where $y_i \epsilon N_{mindist}$
9) For each $y_i \epsilon N_{bin1}$ and $y_i \epsilon N_{bin2}$, sequentially assign unique labels (l_i) from $i = 1 \ldots\ldots\ldots\ldots mid$
10) **for** $i = 1 \ldots\ldots mid$
11) - select instances y_a, y_b from N_{bin1} and N_{bin2} where label (l_i) of y_a, y_b are same, i.e., $y_a(l_i) == y_b(l_i)$
    - Generate a minority class synthetic sample x by $average(y_a, y_b)$
    - Add x to X_{new} and increment X_{newchk}:
12) **end for**
13) If $X_{newchk} < T$, pair instances in X_{new} with instances in each parent bin N_{bin} and repeat steps 10 to 12 to create two different sets of $X_{new[j]}$ ($j = 1, 2$). If still $X_{newchk} < T$, pair each set of current X_{new} with it's immediate parent set and later with predecessor set (i.e., set that contributed to generating that current set) of immediate parent and repeat steps 10 to 12 for subsequent rounds.
14) If $X_{newchk} >= T$, assign all samples in X_{new} with the label of the minority class (defective) and add to dataset N.

End

4.3 Phase 2: Data Partitioning and Pairing

To avoid the problems associated with the k-NN based techniques, MAHAKIL uses a partitional cluster approach. Based on the concept of the equal chromosomes from two parents established in the chromosomal theory of inheritance, we partition or segregate the data samples into two bins based on having a small Mahalanobis distance (typical case) and having a large Mahalanobis distance (atypical case). Using two bins to partition the sample also results in the best and generalized output of synthetic data after several runs of simulation experiments. We believe that it is practical to use two parents because this approach still works even when there are very few defective instances. The two partitions are obtained through sorting using the middle instance found during the ranking to partition them. All data samples with Mahalanobis distance values higher than or equal to the middle data sample are grouped into one partition while the second partition comprises the other data samples. Individual samples within the two partitions are sequentially labeled and then paired. Two instances from each bin are assigned the same label to uniquely identify two independent instances from both bins that have been systematically and orderly paired as a "couple" (Steps 7- 9 of Algorithm 1). The pairing is done in a systematic order such that the first instances of both partitions are paired followed by the second to last instances in sequential order. These pairs are identified by their assigned labels. This is done to ensure that there are no overlapping samples and that the consequent samples to be generated reside within the minority decision boundary and fill the space between the two clusters formed by the partitioning process.

4.4 Phase 3: Synthetic Sample Generation

In the final phase, the synthetic samples are generated by aggregating and computing the mean or average between two paired instances from each partition, which, is then added to the original data N_{min} (steps 10 to 12 of Algorithm 1). Unlike the SMOTE algorithm and k-NN based techniques, which use the k-nearest neighbor to determine and generate a new synthetic instance, this approach considers two instances that may not be close, as in being neighbors, and orderly merges two unfamiliar instances by considering them as "male" and "female" instances. In other words, based on the two partitions created in the previous phase, the resultant instances that are generated will be distinctly unique but related to the parents from both partitions. Maintaining an exact order between the pair of bins ensures that the generated samples also follow the trend of their parent instances. Thus, all of the synthetic data instances are positioned within both bins so that no data instance falls out of the decision boundary of the minority class. This approach is beneficial because the synthetic samples generated do not overlap in the majority region. Another advantage of MAHAKIL over the k-NN based techniques is that, whereas the synthetic samples generated by the k-NN based techniques tend to cluster or organize themselves in a particular group within the minority class samples, thus providing limited information to the trained classifier, the synthetic samples generated by MAHAKIL are evenly distributed to exhaust all possible minority samples within the two clusters thus, providing more information to the classifier. Analytically, MAHAKIL generates synthetic data samples of the minority class within the convex hull of the minority samples. By strictly working within the boundary of minority class, MAHAKIL ensures no data samples are created outside the decision boundary of the minority class. Additionally, the synthetic samples are convex combinations of well segregated parents that are dissimilar regarding the Mahalanobis distance measure ensuring duplicate instances are not generated.

TABLE 1
Description of the Static Code Metrics [32]

Abbreviation	Description
WMC	Weighted methods per class
DIT	Depth of Inheritance Tree
NOC	Number of Children
CBO	Coupling between object classes
RFC	Response for a Class
LCOM	Lack of cohesion in methods
LCOM3	Lack of cohesion in methods, different from LCOM
NPM	Number of Public Methods
DAM	Data Access Metric
MOA	Measure of Aggregation
MFA	Measure of Functional Abstraction
CAM	Cohesion Among Methods of Class
IC	Inheritance Coupling
CBM	Coupling Between Methods
AMC	Average Method Complexity
Ca	Afferent couplings
Ce	Efferent couplings
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
LOC	Lines of Code
Bug	Number of detected bugs in the class

Based on the *Pfp* value, the algorithm computes the amount of synthetic data instances to be generated and prunes other data that are generated but not required. All data instances in the nodes or levels above the last level are included in the final dataset. The pruning starts from the last level of data generated by dividing the amount of data samples left to make our required final set complete by the number of nodes on that level. The value computed is then taken from each node on that level, which means that each node on the last level will contribute equal amount to the final set.

5 Experimental Methodology

The research question we ask in this study is "Can MAHAKIL perform better than other oversampling approaches?" The primary goal of this study is to improve performance of defect prediction models trained on imbalanced datasets using our proposed oversampling approach. Comparative experiments are conducted in order to answer the research question. The performance of MAHAKIL is compared with that of SMOTE [14], Borderline-SMOTE [16], Random Over-Sampling (ROS), ADASYN [15] and the no sampling approach (NONE).

Below, we describe the datasets, selected oversampling approaches, predictive models, evaluation metrics and statistical tests used for the study. For a comprehensive study, the datasets chosen vary in terms of the imbalance ratio and sample size. Predictive models spanning different domains of machine learning and statistics are also applied and a further comparison is made using statistical tests to observe the statistical significance and dominance of our sampling approach on predictive models over the other four data sampling approaches considered.

TABLE 2
Summary Statistics of Selected 20 Imbalanced Datasets Extracted from the PROMISE Data Repository

#	Release	# modules	# defects	defects (%)
1	ant-1.3	125	20	16.0
2	ant-1.4	178	40	22.5
3	ant-1.5	293	32	10.9
4	ant-1.6	351	92	26.2
5	arc	234	27	11.5
6	camel-1.4	875	145	16.6
7	camel-1.6	965	188	19.5
8	ivy-1.4	241	16	6.6
9	ivy-2.0	352	40	11.4
10	jedit-4.0	306	75	24.5
11	jedit-4.1	312	79	25.3
12	jedit-4.2	367	48	13.1
13	log4j-1.0	135	34	25.2
14	pbeans2	51	10	19.6
15	redaktor	176	27	15.3
16	synapse-1.0	157	16	10.2
17	systemdata	65	9	13.8
18	tomcat	858	77	9.0
19	xerces-1.2	440	71	16.1
20	xerces-1.3	453	69	15.2

5.1 Benchmark Datasets

We conduct an experiment on 20 different benchmarked datasets extracted from PROMISE Repository[2] which is freely available online. Jurezcko and Madeyski [29], and Jurezcko and Spinellis [30] kindly donated the projects, which are stored in the PROMISE Repository. The PROMISE Repository stores a collection of datasets that are commonly used by the software engineering research community to construct predictive software models [31]. Twenty static code metrics (Table 1) are measured from classes or modules of these projects which are written in JAVA. Non-defective modules are labeled as zero and defective modules are labeled with the total number of defects present in the module. A summary of the data sets, including the total sample size, number of features and percentage of fault-prone modules (*Pfp*) or defects for each dataset, is presented in Table 2. With varying degrees of sample size, features and imbalance ratios, these datasets provide an extensive domain for the evaluation of the sampling techniques.

5.2 Baselines

MAHAKIL is compared to four other common oversampling techniques and the no sampling approach (NONE). A brief explanation of how these techniques work is provided below.

SMOTE. Proposed by Chawla et al. [14], this technique over-samples the minority class in a dataset by creating "synthetic" samples. It over-samples the minority class in a bid to make the dataset as balanced as possible based on the configuration parameter values. To generate these synthetic samples, each minority class sample is considered and the new samples are introduced along the line segments that join any of the k minority class nearest neighbors.

Borderline-SMOTE. This is a modification of the SMOTE technique but with the main focus on harder-to-classify minority class data instances, which are referred to as

2. http://openscience.us/repo/

TABLE 3
Classification Models and their Hyperparameter Configurations

Model	Overview	Hyperparameters
C4.5	J48 Decision Tree	$c = \{0.05, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70\}$
NNET	3-layer Neural Network	size = {4, .. ,28}, decay = {0.10, 0.20}
KNN	K- Nearest Neighbor	$c = \{2 * (0, \ldots, 7) + 1\}$
RF	Random Forest	mtry = { 10, 50, 100, 200, 250, 500, 1000}
SVM	Support Vector Machine	$c = \{2^{-6}, \ldots, 2^{10}\}$

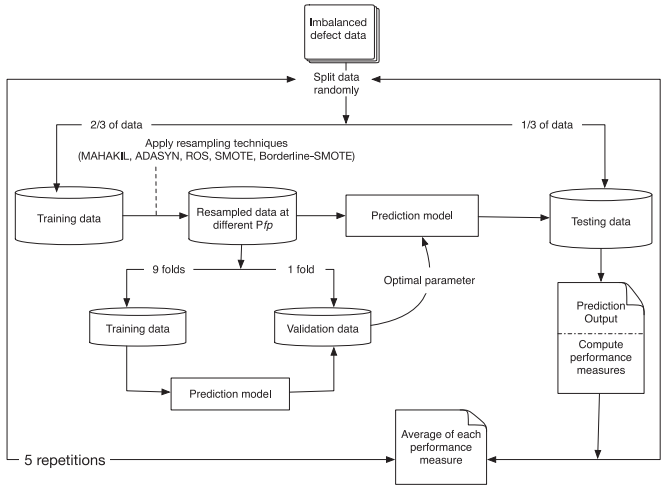

Fig. 3. Framework of experiments.

borderline data instances [16]. The algorithm first finds minority class instances that have more majority class instances as nearest neighbors than minority class instances and applies the SMOTE technique to such instances. This approach has the advantage of strengthening the borderline between the majority and minority class data instances.

ADASYN. ADASYN was proposed by He et al. [15] and it assigns weights to the minority classes and dynamically adjusts the weights in a bid to reduce the bias in the imbalanced dataset by considering the characteristics of the data distribution. The ADASYN algorithm incorporates a density distribution in automatically deciding the number of synthetic samples needed for each minority class sample. The learning algorithm is induced to focus on the hard to learn (classify) examples within the minority class samples. Therefore, the samples generated are not equal for all samples.

ROS. ROS increases the number of minority instances by randomly replicating the already existing minority class instances. A simple and easy approach to implement, ROS has performed favorably in previous empirical studies [4], [12].

5.3 Experimental Design

For the experiments, we use five classification algorithms (Table 3) which have been used in similar class imbalance studies [17], [33], [34] to evaluate the effect of each sampling approach on the datasets. Conventional methods such as Neural Networks [35], SVM [36] and the C4.5 decision tree [37] are used. We also consider Random Forests (RF) which is are known to improve classification accuracy because they grow an ensemble of classification trees and allow them vote on the classification decision [34]. Lessman et al. [38] showed that RF was significantly better than 21 other prediction models. KNN [39] is also used because it is easy to implement and it classifies an unknown instance by considering its nearest neighbors. Hyper-parameter configurations for the classification models estimated using a 10-fold cross-validation method are presented in Table 3.

The training data are randomly selected using 2/3 of the sample size and the remaining 1/3 as the testing data for each dataset. The division is done using stratification such that the proportion of minority class distribution or *Pfp* is kept intact within both the training and testing datasets. Fig. 3 shows the steps followed for the empirical evaluation of the data re-sampling methods for each dataset. To reduce the impact of bias caused by the random splitting method, the whole experiment is conducted 5 times. The results are averaged across the independent runs and presented. The sampling techniques are applied only to the training datasets. Because a significant number of the projects have fewer minority samples than the number of features, we use a multicollinearity technique to eliminate features with very high correlations (>90 percent) and zero-variance predictors. This helps to successfully invert the singular matrices during the computation of the Mahalanobis distance for the MAHAKIL approach.

We also consider a 10-fold cross-validation in choosing the parameters of the validation set during the model construction on the preprocessed or re-sampled training datasets i.e., we randomly partition the whole dataset into 10 folds of data and use all but one for training and validating on the remaining fold. This procedure ensures that the optimal parameters are found and selected after validating them on each fold of data. The experiments are conducted on each dataset after re-sampling with each over-sampling approach at different levels of *Pfp*. With all datasets having a *Pfp* of less than 30 percent, we conduct three different experiments for those datasets by applying each over-sampling approach until the *Pfp* reaches 30, 40 and finally 50 percent. We stop over-sampling at 50 percent based on the assumption of a balanced dataset in most algorithms and also based on the conclusion of Ahmad Abu, et al. [40] that over-sampling to at most 50 percent achieves better results.

The experiments were all conducted using R [41], an open source statistical tool. The library packages Caret [42] and DMwR [43] were used for the model construction and SMOTE sampling approach respectively. The ADASYN technique was implemented following the instructions given by the authors [15] using the MATLAB tool. For SMOTE, Borderline-SMOTE and ADASYN which considers a k-neighbor value, k was set to the default value of five.

5.4 Evaluation Performance Metrics

A typical defect dataset is categorized into two groups: defective and non-defective. Conventionally, minority or defective classes are labeled as positive and majority classes labeled as

TABLE 4
Confusion Matrix for the Two-Class Problem

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

negative in a two-class problem. From the confusion matrix (Table 4), True Positives (TP) are the number of positive samples correctly classified as positive, False Positives (FP) are the number of negative samples wrongly classified as positive, True Negatives (TN) are the number of negative samples correctly classified as negative and False Negatives (FN) are the number of positive samples wrongly classified as negative. Indicators such as Precision, Recall or probability of detection (*pd*) and F-measure have also been used to assess the performance of models on the minority class [44], [45]. Precision and the F-measure indicator which computes the harmonic mean between recall and precision are excluded from our experiment because precision is known to be an unstable indicator for models trained on highly imbalanced datasets [2]. We adopt the recall (*pd*) and *pf* measures for our experiment. Recall (*pd*) is a measure of completeness that attains a high value of 100 percent if the number of false negatives is zero. The probability of false alarms (*pf*) which measures how many of the data instances that activated or prompted the detector did not actually have the defect concept [46] is also computed. A lower *pf* implies that less time and effort will be required in determining the true defects. A high *pd* and low *pf* indicate a better prediction model [2]. The evaluation metrics used for this study are defined in equations (1) to (2) computed from Table 4.

$$Recall(pd) = \frac{TP}{TP + FN} \tag{1}$$

$$pf = \frac{FP}{TN + FP} \tag{2}$$

5.5 Performance Comparison

To statistically evaluate and compare MAHAKIL to the other four techniques, we adopt the *win-tie-loss statistics* which has been used in several empirical studies [47], [48]. Three counters-*wins*, *ties* and *losses* are used to assess the overall performance of a predictor (model*sampling technique) in a pairwise manner for all possible pairs of predictors. By using Brunner's statistical test ($p < 0.05$) [49], two predictors are compared and if the performance distributions of the two predictors are not significantly different, the *ties* counters of both estimators are increased by 1. Otherwise, the *wins* counter of the predictor with a higher performance is increased and the *losses* counter of the other predictor with low performance is also increased. All of the counters are summed per the *Pfp* value to find the best performing predictors. Furthermore, to ascertain the practical significance of the results, we compute the effect size using Cliff's method with Hochberg's method proposed by Brunner et al. [49] and recommended by Kitchenham et al. [50] as a more robust statistical test for non-normal data. The effect sizes ($\widehat{\delta}$) are interpreted using the magnitude labels (small ($\widehat{\delta} <= 0.112$), medium ($0.112 > \widehat{\delta} < 0.428$) and large ($\widehat{\delta} >= 0.428$)) proposed by Kraemer and Kupfer [51].

6 Empirical Results

6.1 Overall Results

In this section, we present the results of each sampling approach on the prediction models and compare the performance of MAHAKIL with ROS, SMOTE, Borderline-SMOTE, ADASYN and No Sampling (NONE). We present the results for each sampling technique on each dataset per each percentage of fault-prone modules (*Pfp*) value. For each dataset, we present the performance values for all of the *Pfp* values and the prediction models in Figs. 4 and 5. In the figures, the dot plot compares the performance of each sampling method (represented with different shapes) across the various *Pfp* values (represented with different colors) per prediction model. It can be clearly seen that the application of the sampling methods at different *Pfp* rates produces a better performance than NONE regarding the *pd* performance values across all prediction models. However, NONE has the lowest and better *pf* values compared to the application of the sampling methods across most of the prediction models and datasets.

From Figs. 4 and 5, we observe the following.

(1) Across all prediction models, SMOTE and MAHAKIL consistently performed better regarding the *pd* values than Borderline-SMOTE, ADASYN and ROS. Their distributions were quite similar with SMOTE slightly edging MAHAKIL.
(2) SMOTE resulted in the worst *pf* values whiles NONE and MAHAKIL resulted in the best *pf* values across all prediction models.
(3) Across all prediction models and datasets, the best *pd* performance was achieved when *Pfp* was increased to 50 percent for all sampling methods.

Across all models and datasets, MAHAKIL performed relatively better with high *pd* and low *pf* values. This can be explained by the uniqueness and closeness or the representativeness of the new synthetic data injected into the dataset. The models trained on the SMOTE resampled datasets resulted in the highest mean *pd* but with the highest mean *pf*. This implies that the application of SMOTE results in the models detecting more defective models, although this approach is not cost effective. The results of the statistical tests conducted to validate the results are presented in the next section.

6.2 Statistical Comparison of Techniques

To investigate in detail the performance of MAHAKIL and demonstrate the effectiveness of MAHAKIL in achieving better (significant) performance measures, we present the win-tie-loss results of MAHAKIL against those of SMOTE, Borderline-SMOTE, ADASYN, ROS and NONE per each *Pfp* for each performance measure across all 20 datasets in Fig. 6. Additionally, the practical effect of the statistical results is also computed and presented in 6. We compute the wins, ties and losses for each predictor on all datasets per each round of experiment (i.e., *Pfp* value). For each *Pfp* rate, the win-tie-loss statistic is presented together with the magnitude of the effect size. We observe from the figure that the models trained on MAHAKIL resampled data significantly outperformed most of the sampling techniques, with the *Pfp* increasing from 30 to 50 for both performance measures.

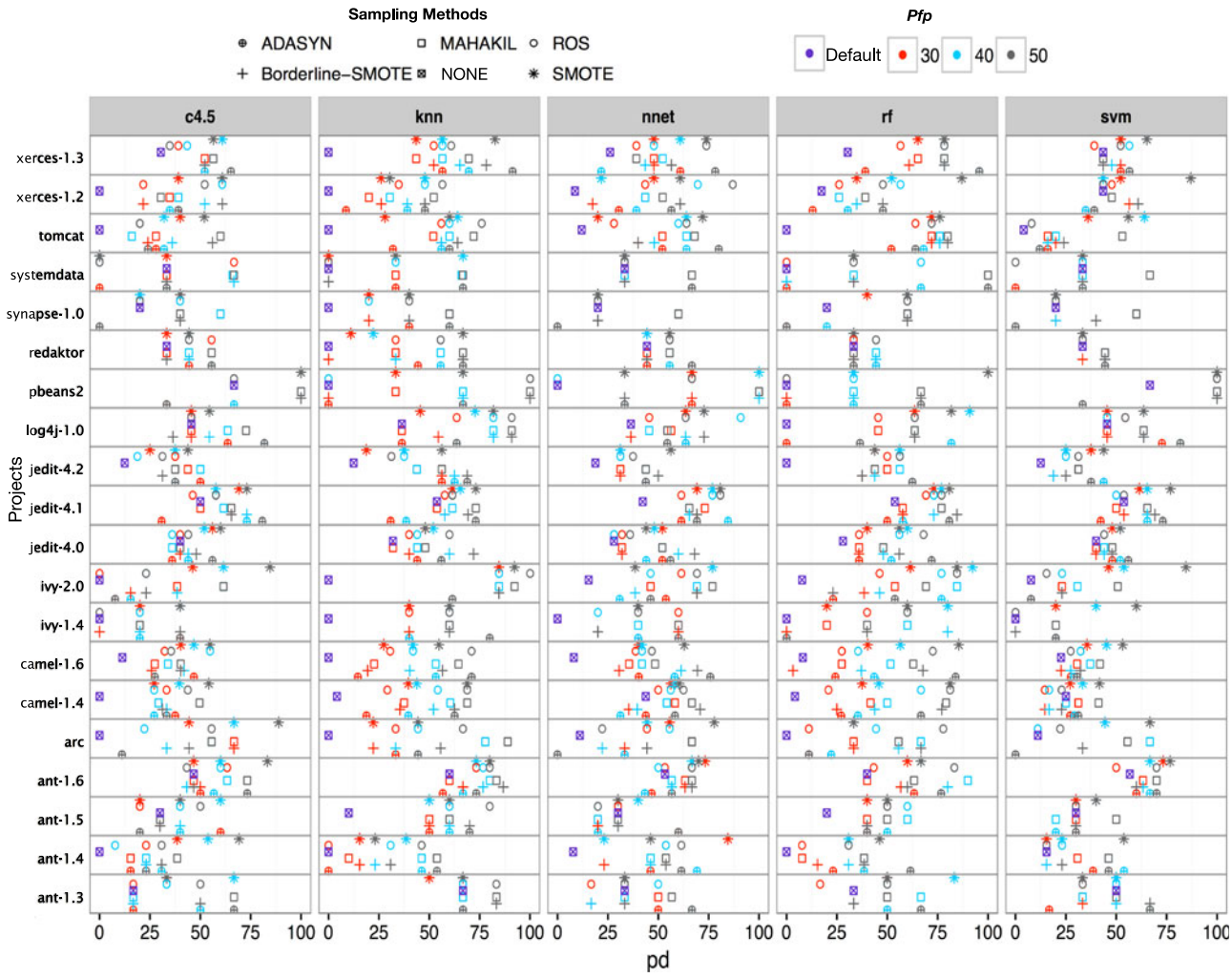

Fig. 4. Performance plot of the sampling techniques per each prediction model for the *pd* performance metric on 20 imbalanced datasets across different percentage of fault-prone modules (*Pfp*) values.

MAHAKIL also proved to be statistically and significantly better in terms of the *pf* value having atleast a win across all of the prediction models. Most of the wins were practically significant as confirmed by the magnitudes of the effect size. However, NONE consistently outperformed MAHAKIL achieving magnitudes of small or medium effect sizes indicating that for the lowest *pf*, the no sampling method produces better results than using the sampling methods.

MAHAKIL consistently outperformed the NONE method with significant and practical effect sizes regarding the *pd* results. There were several ties between MAHAKIL and each of the other four sampling techniques regarding the *pd* results across all models. From the figure, we can conclude that MAHAKIL achieves better or comparable prediction results than the other four sampling techniques. The results also show that for best results, MAHAKIL should be applied at a *Pfp* of 50 percent because all of the models trained on datasets resampled at this *Pfp* value produced significant and practical results.

For more details about the overall performance of MAHAKIL and to determine which predictors (combination of prediction models and sampling methods) produce the best performance, the total win-tie-loss results per *Pfp* value are reported in Table 5. For each performance measure, we compute the total wins, ties and losses for each predictor on all datasets per each round of experiment (i.e., *Pfp* value). Each single predictor is compared with 29 other predictors (5 models * 6 Sampling techniques - 1). We rank the results by their total wins-losses. Out of the 58 (29*2) overall win-tie-loss comparisons aggregated from two performance measures per *Pfp* value, the application of MAHAKIL always resulted in a positive win-loss value. The models trained on MAHAKIL resampled datasets consistently had positive wins-losses values and were in the top half of the predictors. The models trained on data resampled with ROS and SMOTE performed worse with the highest numbers of losses and negative wins-losses values. Overall, the results shows that MAHAKIL clearly outperforms all four sampling techniques and the no sampling method. All five prediction models combined with MAHAKIL performed better than any other combinations, with an impressive number of wins and minimum losses when *Pfp* was 50 percent. Overall, the combination of RF and KNN with MAHAKIL produced the best performance with the highest number of wins and few losses. Based on the results, RF with MAHAKIL is recommended for highly imbalanced defect datasets when a *Pfp* of 40 or less is applied. For a balanced dataset (*Pfp* of 50), the KNN prediction model combined with MAHAKIL is

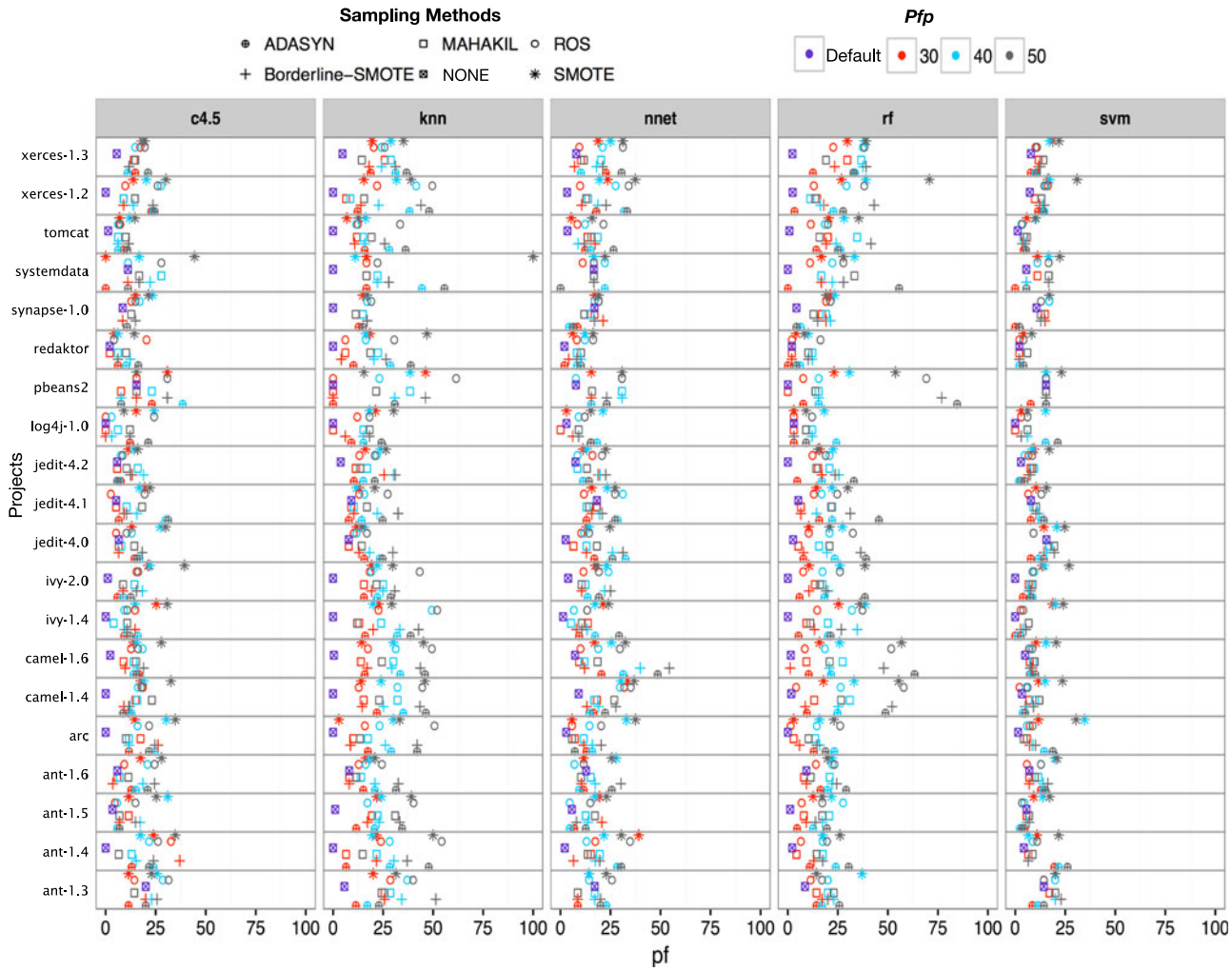

Fig. 5. Performance plot of the sampling techniques per each prediction model for the *pf* performance metric on 20 imbalanced datasets across different percentage of fault-prone modules (*Pfp*) values.

recommended. In agreement with Fig. 6, we observe that for the best results, MAHAKIL should be applied at a *Pfp* of 50 percent because the models trained at this *Pfp* value occupied the top five positions.

7 Discussion

7.1 Stability of MAHAKIL

The datasets used for the experiments vary in terms of the imbalance ratio. We consider datasets with an imbalance ratio of less than 15 percent as severely/highly imbalanced and the remaining ones as lowly imbalanced. Severely imbalanced datasets imply that more artificial data instances will be generated compared to the low imbalanced data when we apply the sampling methods at a specified *Pfp* rate. From the *pd* plot in Fig. 4, we observe that as the *Pfp* increases, the *pd* values increase across all of the prediction models. For example, when we consider the systemdata project, which is severely imbalanced, there are only nine default defective instances, with the test data consequently comprising only three instances. NONE achieves a minimum value of 0 percent and a maximum value of 33 percent across all prediction models. The sampling methods except MAHAKIL obtained a *pd* value of 0 percent across all the prediction models. MAHAKIL achieved the highest *pd* value as the *Pfp* increased to 50 percent, which could be due to the quantity of diverse and important data instances introduced into the training data by MAHAKIL as the *Pfp* increased. When we consider a dataset that is lowly imbalanced, such as pbeans2, we observe similar results regarding the MAHAKIL method whiles the other methods except SMOTE obtain a 0 percent *pd* value across the prediction models. We argue that our approach is more stable and even the smallest amount of synthetic data introduced into the training data improves the *pd* performance of the prediction models. Similarly, the *pf* plot in Fig. 5 shows a similar pattern to the *pd* plot, with the values of *pf* increasing as *Pfp* increases. However, MAHAKIL obtains the lowest values across all sampling methods.

7.2 Choice of Prediction Models

The KNN and RF models achieved the best results when trained on MAHAKIL resampled datasets. We attribute this to how these models work and the positioning of the synthetic generated samples within the minority class by MAHAKIL. MAHAKIL generates minority samples that are distributed within the sample space. Consequently, the test set's similar instances are found in the generated minority space more easily by these prediction models, especially the KNN model which selects samples closer to

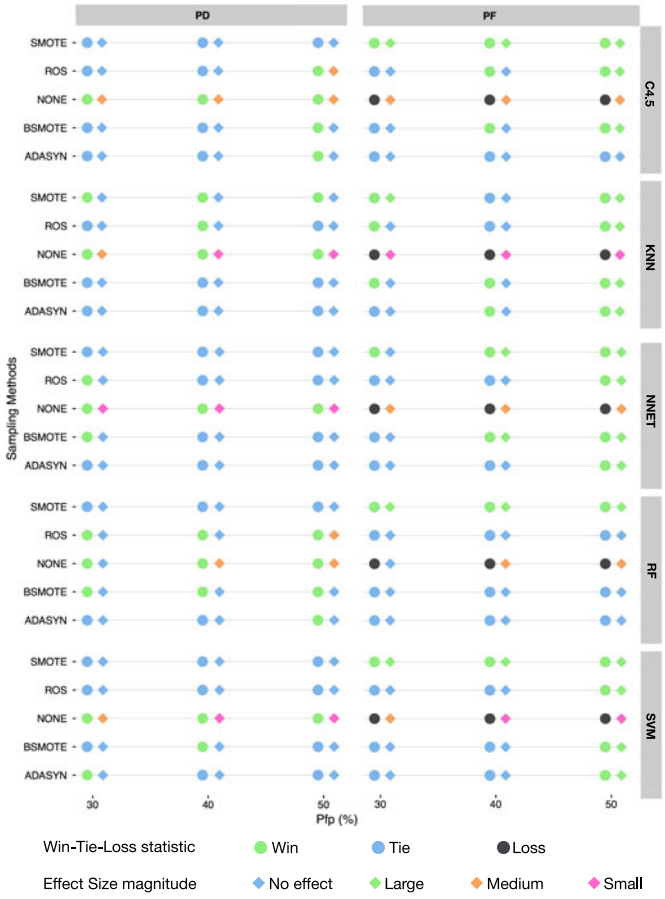

Fig. 6. Brunner's statistical test win-tie-loss and effect-size comparison of MAHAKIL versus ROS, ADASYN, Borderline-SMOTE, SMOTE and NONE across all 20 datasets per each defect prediction model, performance measure (*pd, pf*) and *Pfp* value (30, 40, 50 percent).

the test instance. As an ensemble method, RF is more robust against overfitting and the parameter settings [52]. Based on the internal parameters, RF is able to compute the importance of each predictor and choose the most informative predictor. It is also able to measure the internal structure and pattern of the training data [53], which makes it easy to identify the patterns of the MAHAKIL resampled datasets because the minority region is easy to identify. Lessman et al. [38] confirmed that the RF classifier was a more effective classifier for software defect prediction studies. Impressively, at a *Pfp* rate of 50 percent, all of the prediction models trained on MAHAKIL resampled datasets outperformed the other sampling methods.

7.3 Why MAHAKIL Performs Best

The findings of this study provide substantial empirical support demonstrating that unique data instances can be generated with predetermined characteristics. For all models, MAHAKIL performed relatively better with high *pd* and low *pf* values. This can be explained by the uniqueness and closeness or the representativeness of the new synthetic data injected into the dataset.

The performance of MAHAKIL can be attributed to the succinct and almost uniformly spread of the newly generated samples within the boundary of the minority class. Based on the strategy used to generate new data instances, all "empty spaces" within the minority data are filled if more data instances are generated which, produces diverse data instances. To demonstrate the diversity, we compute the diversity (variety of different data instances) for each metric in the minority class of the resampled dataset. The diversity for each metric was computed using the Shannon-Wiener diversity index (H)[54], which is a widely accepted metric for computing and quantifying diversity in ecological and environmental studies. This method assigns the same weight to common and rare species. H ranges between 0 and 5, with a higher H value implying higher diversity and a lower likelihood of selecting two instances of the same species, and hence less duplicates. The line plots in Fig. 7 present the average diversity values of the metrics for each dataset and sampling method.

We plot the diversity for the data when no sampling is applied (NONE) at the *Pfp*-20 rate. For all sampling methods, Fig. 7 shows that the diversity within the minority class increases as the *Pfp* increases or more data instances are produced. However, the diversity value of the SMOTE resampled datasets decreases as the *Pfp* reaches 40 percent. Across all datasets, MAHAKIL achieves the highest diversity values for all the metrics per each *Pfp* especially at a *Pfp* of 50 percent. Based on these results, the success of MAHAKIL confirms that prediction models not only require ample training data but also very relevant and exhaustive data that contribute more information to the model. Data quality is a very crucial issue in the domain of defect prediction and using sampling approaches that generate less informative data tends to only worsen the situation.

7.4 Threats to Validity

Threats to Validity refers to the extent to which a research study actually studies what the researcher purports to investigate [55]. To address the observed potential limitations in our findings, the following discusses the threats to the construct, internal, statistical conclusion and external validity of the study as recommended by Wohlin et al. [56].

7.4.1 Construct Validity

Our proposed approach assumes that defective classes can be appropriately separated into two clusters, which may not always be the case for defect datasets. Some defect datasets consist of several multi-clusters and even majority instances mix with minority instance clusters. Restricting our technique to identifying mini clusters and working in local patches will improve the method and reduce the *pf*. We leave this for future studies. The metrics considered for our study are a potential threat to our experimental results. By using a single set or type of metrics, we cannot generalize our results to other types of defect metrics. Nonetheless, static code metrics are known to perform very well and have been used in several empirical studies on defect prediction because they are easy to collect and can be collected from any software once the source codes are available [57]. We also acknowledge that the sampling and splitting approach could affect the classification results. We randomly split the datasets into training (2/3) and testing (1/3)

TABLE 5
Performance in Terms of Wins, Losses, and Wins-Losses Aggregated from the Two Performance Measures on 20 Datasets per Each *Pfp* Value

Pfp 30					*Pfp* 40					*Pfp* 50				
Model	Sampling	Wins	Losses	Wins-Losses	Model	Sampling	Wins	Losses	Wins-Losses	Model	Sampling	Wins	Losses	Wins-Losses
RF	MAHAKIL	20	5	15	RF	MAHAKIL	27	11	16	KNN	MAHAKIL	34	11	23
RF	NONE	26	12	14	C45	MAHAKIL	24	11	13	RF	MAHAKIL	32	11	21
NNET	MAHAKIL	24	11	13	KNN	MAHAKIL	26	14	12	NNET	MAHAKIL	29	9	20
RF	ADASYN	19	9	10	NNET	MAHAKIL	23	11	12	C45	MAHAKIL	29	13	16
C45	MAHAKIL	16	7	9	RF	ADASYN	24	14	10	SVM	MAHAKIL	23	12	11
RF	BORDERLINE	19	11	8	RF	BORDERLINE	26	18	8	RF	SMOTE	21	15	6
NNET	ADASYN	19	12	7	SVM	ROS	24	16	8	NNET	ADASYN	19	14	5
RF	ROS	19	12	7	SVM	MAHAKIL	20	14	6	RF	BORDERLINE	27	22	5
SVM	MAHAKIL	11	4	7	NNET	ROS	16	13	3	C45	NONE	28	27	1
SVM	ADASYN	18	13	5	RF	SMOTE	19	16	3	NNET	BORDERLINE	18	17	1
KNN	MAHAKIL	12	8	4	C45	BORDERLINE	19	17	2	RF	ADASYN	24	23	1
SVM	ROS	12	8	4	RF	NONE	28	26	2	RF	NONE	28	27	1
C45	NONE	27	25	2	RF	ROS	26	24	2	KNN	NONE	28	28	0
NNET	ROS	13	12	1	C45	NONE	28	27	1	SVM	NONE	27	27	0
C45	BORDERLINE	10	10	0	KNN	NONE	28	28	0	NNET	NONE	26	27	-1
KNN	NONE	28	28	0	SVM	NONE	27	27	0	RF	ROS	24	26	-2
RF	SMOTE	12	12	0	C45	SMOTE	13	16	-3	NNET	ROS	16	19	-3
SVM	NONE	27	27	0	NNET	NONE	25	28	-3	SVM	SMOTE	17	20	-3
C45	ADASYN	9	11	-2	KNN	SMOTE	14	18	-4	SVM	ROS	16	20	-4
NNET	SMOTE	17	19	-2	NNET	ADASYN	9	13	-4	C45	SMOTE	14	19	-5
C45	ROS	8	12	-4	NNET	BORDERLINE	12	16	-4	KNN	ADASYN	18	23	-5
NNET	NONE	19	24	-5	SVM	ADASYN	12	17	-5	KNN	BORDERLINE	17	24	-7
SVM	BORDERLINE	7	13	-6	C45	ADASYN	15	21	-6	C45	ADASYN	18	26	-8
NNET	BORDERLINE	6	14	-8	NNET	SMOTE	11	19	-8	C45	BORDERLINE	18	26	-8
KNN	ADASYN	6	15	-9	SVM	BORDERLINE	10	18	-8	SVM	ADASYN	11	20	-9
C45	SMOTE	8	18	-10	C45	ROS	16	25	-9	SVM	BORDERLINE	13	22	-9
KNN	BORDERLINE	5	15	-10	SVM	SMOTE	16	25	-9	KNN	ROS	14	24	-10
SVM	SMOTE	12	22	-10	KNN	ADASYN	15	25	-10	KNN	SMOTE	11	22	-11
KNN	ROS	5	21	-16	KNN	ROS	10	22	-12	NNET	SMOTE	13	24	-11
KNN	SMOTE	4	28	-24	KNN	BORDERLINE	10	23	-13	C45	ROS	15	30	-15

Higher wins(%), wins-losses(%) and lower losses(%) indicate higher performance of the predictor. The predictors are ordered by wins-losses.

sets. Most empirical studies use this procedure for predictive data modelling [38], [58], [59] making it an accepted approach. It would be interesting to examine the effects of our proposed oversampling method on cross-version datasets. However, we leave this for future studies. Because there are no universally agreed upon performance measures for assessing the prediction performance on imbalanced data, we used two different measures: Recall (*pd*) and *pf*. Recall (*pd*) and *pf* enabled us to know the performance of the predictors on the minority class. The assessment results for the performance of the prediction models using the AUC and G-mean measures are provided in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2017.2731766. Other measures not used for our study are left for future studies.

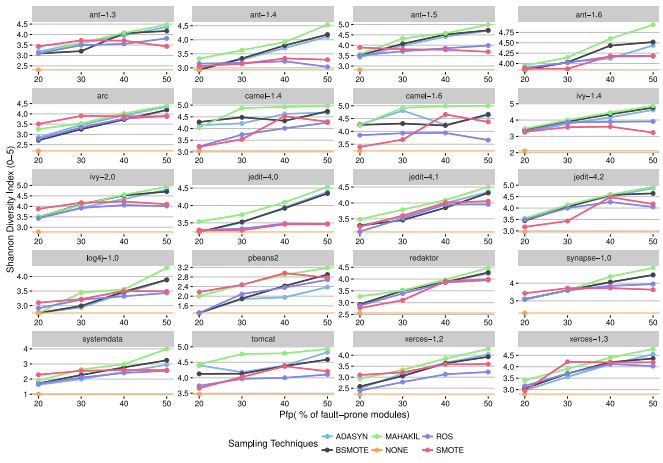

Fig. 7. Average diversity value at each *Pfp* for all 20 projects.

7.4.2 Internal Validity

Our approach is dependent on the use of the Mahalanobis distance metric. The advantages of the Mahalanobis distance notwithstanding its use comes with a high price. For very large data records, the computation of the covariance matrices can be difficult to determine accurately, especially if there are a large number of features. More memory and time resources are required because these resources grow quadratically rather than linearly with the number of features. These problems may be insignificant when only a few features are needed. This challenge also affects methods based on the euclidean distance measure and a more advanced and time-efficient technique is required to handle the computation of covariance matrices for large dimensional features. Moreover, covariance matrices cannot be inverted for datasets with fewer minority samples than the number of features (singular matrix). We adopted multicollinearity techniques to eliminate metrics with very high correlations. A generalized inverse method can be incorporated to compute an inverse-like matrix that will ignore the non-informative metrics and the impact of such a method on our algorithm is left for future studies. The hyper-parameters configuration set could introduce some bias in our results. However, we used 10-fold cross validation at each run and repeated the overall experiments five times, which enabled us to report the average results across all run times. An important known validity in empirical experiments is the

quality of the data, which is difficult to obtain and verify. We do acknowledge that noise or outliers inherent in datasets extracted from most open-source projects could have significant effects on prediction performance as reported by Gray et al. [3]. As such, applications of data cleansing techniques such as noise detection and elimination measures [60], [61] are left for future studies.

7.4.3 Statistical Conclusion Validity

In this study, we used Brunner's statistical test for the win-tie-loss analysis and Cliff's method with Hochberg's method for the effect size computations. These statistics are recommended by Kitchenham et al. [62] as a more robust and reliable test than other statistical tests for evaluating the performance of prediction models. The impact of this threat on our study is thus limited.

7.4.4 External Validity

Our study used a limited number of datasets. The projects used were all from one source, the PROMISE repository and have been extensively used in several defect prediction studies. However, our results may not be generalizable to all software projects, especially proprietary projects, because we used only open source projects. Nevertheless, we provided a detailed description of the experimental setup, which will be useful for replicating our experiment using commercial software systems. We also used only five classification techniques spanning the machine learning, data mining and statistical based families. The effects of our proposed technique on other classification models that were not examined in this study are not known and extra evaluation of other models are required to generalize our results. Lastly, we compared our proposed technique to four other sampling techniques. We intend to compare our technique with other simple and more complex techniques in a future study.

8 Future Works

As a novel approach to facilitating learning from class imbalanced datasets, MAHAKIL aims to improve the performance of the prediction models by generating new and diverse synthetic data and strengthening the decision boundary of the minority class. This approach outperformed the other oversampling approaches considered with respect to the *pd* and *pf* performance measures. Our approach shows great potential and can benefit class imbalance learning in several domains. We present a summary of the likely future research areas regarding our approach.

In the domain of cross project or company prediction, where historical datasets are not readily available for training by new projects and thus acquires projects of similar characteristics and domains for training classifiers, MAHAKIL could prove to be of great value when appropriately applied with similar projects to generate synthetic data, which is close to the new project under consideration. This could be applied after using state of the art techniques to filter and select the appropriate datasets for the new project. This technique could also be used to generate both majority and minority classes of datasets for cross project prediction. It would be interesting to investigate this area further. To significantly reduce the *pf*, our approach could be modified to identify multi-clusters and work in local patches. The MAHAKIL algorithm could thus be applied to work within each cluster of minority instances found after applying clustering algorithms. In this study, the performance of MAHAKIL was assessed by comparing it with ROS, SMOTE, Borderline-SMOTE and ADASYN and five single or base classifiers. However, ensemble classifiers have been shown to perform better than single classifiers and it would be interesting to assess the performance of ensemble classifiers integrated with our algorithm. Moreover, several aspects of the theory of inheritance were not considered in the proposed method, such as fitness evaluation, mutation and recombination. We intend to consider these procedures in a future study to improve our method. Lastly, the internal structure of how MAHAKIL works could be explained using linear algebra from an analytical perspective. We intend to reposition and explain the MAHAKIL approach using convex hull algorithms in a future study.

9 Related Works

9.1 Software Defect Prediction

Defect predictors have been used to improve the quality of software and reduce the overhead cost of producing software. Several machine learning, data mining and statistical models have been proposed and evaluated for software defect prediction. Using basic discriminant analysis, Munson and Khoshgoftaar [63] observed that there is a strong useful relationship between software faults and software complexity metrics during development and suggested that it was possible to use predictive models to help determine program faults. Afterwards, several conventional methods of constructing fault-prone module prediction models have been proposed [35], [64], [65], [66], [67]. For example, logistic regression analysis has been used in many studies to determine the given value of a set of explanatory variables (set of metrics values measured from the module) as demonstrated by Basili et al. [64].

Modeling methods such as linear discriminant analysis [66], classification tree [68], neural network [35], support vector machine [36], Bayesian classifier [69], k-neighbor classifier [70], have also been used. Recent studies [38], [71] have recommended the Random Forest model because it is fast to train, less complex and more robust in relation to the parameter settings.

The main goal of existing research has been to increase the prediction performance. Several methods have been used to achieve this, such as considering the modeling methods, using robust feature selection and preprocessing techniques and proposing new defect metrics. Modification of the class distribution using data re-sampling approaches is another method of improving prediction performance. This approach is widely gaining recognition because it is independent of the prediction model and its effects are clearly observable. In this study, we consider rectifying the imbalanced nature of the defect datasets to improve the prediction performance.

9.2 Approaches for Tackling Class Imbalanced Defect Datasets Learning

The problem of class imbalance makes it challenging to conduct experiments with datasets in the field of software defect prediction because the models trained on such skewed defect datasets tend to be more biased towards the non fault-prone modules. These models inadvertently disregard the fault-prone modules [7] and most often predict fault-prone modules as non fault-prone. However, several significant studies have sought to address this prevalent issue. The approaches of the majority of these works can be categorized as follows: (a) re-sampling the original datasets by manipulating the class distribution of the training data [14], [72] (b) assigning specific cost to the classes so as to reduce the misclassification cost [73], [74], [75] and (c) adoption of ensemble methods [18], [76]. Although no method has proven to be universally better across all of the domains, the application of sampling methods has been shown to be more successful and dominant in recent studies. We focused more on sampling techniques because they are easy to apply and can be applied to any prediction model (i.e., independent of the prediction model being used). Hence, we provided a brief overview of various studies on sampling methods. Further information on the categories other than sampling techniques can be found in [1], [77].

The data sampling techniques focus on balancing the majority and minority class distribution samples by either increasing the minority class, which is known as over-sampling, or by reducing the majority class, which is known as under-sampling, or by effectively combining both methods. The simplest and most common form of under-sampling is Random Under-Sampling (RUS) [78] in which the majority data samples are randomly removed through deletion. Similarly, Random Over-Sampling (ROS) randomly duplicates the minority samples so as to increase the minority samples. However, ROS adds no further or new information to the classifier because the datasets consist of duplicates and therefore lead to over-fitting [79]. An improved method proposed by Chawla et al. [14] known as Synthetic Minority Over-sampling TEchnique (SMOTE), increases the minority class samples by generating new synthetic samples that contribute vital information to the dataset. This method creates new instances along a line segment that joins each instance and some defined k minority class nearest neighbors instances. Several synthetic instances are thus created but depending on the degree of oversampling required, the number of synthetic samples are randomly selected.

Several other modifications of the SMOTE algorithm have been proposed [15], [16], [17], [80]. Han et al. [16] observed that data instances on the borderline of the classification boundary were the most likely to be misclassified and proposed the Borderline-SMOTE method, which generates synthetic data along the line separating the minority and majority class samples in a bid to strengthen the minority samples found on the decision border. Another form of the SMOTE is the Adaptive synthetic sampling approach (ADASYN) proposed by He et al. [15]. They consider the classification boundary and apply a weighted distribution method that assigns weights depending on the learning attributes of the minority class samples thus differentiating the harder-to-classify minority class samples from the easier-to-learn minority class samples. Other sampling methods use clustering algorithms to partition the dataset before under or oversampling the dataset. The majority weighted minority oversampling technique (Mwmote) proposed by Barua et al. [17] partitions the datasets using clustering before using the euclidean distance similarity measure to find very close class samples and synthetically generate samples based on the weights assigned to the minority class samples. Several studies have preferred oversampling to under-sampling [40], [78], [79]. Because undersampling removes the class samples, information crucial for building an effective predictive model is lost. Thus, undersampling is not considered in our study. Although our approach is also synthetic based, it differs from the others in terms of generating diverse data instances. In contrast to the other approaches, which consider very similar parent instances with regards to their similarity distance measure values, our approach generates new data instances from two dissimilar parents

10 Conclusion

With most real world defect datasets highly imbalanced, resampling approaches have been adopted to alleviate the class imbalance learning issues. The majority of resampling approaches are synthetic based, where synthetic minority data instances are generated to balance the distribution between the minority and majority class samples. Although these synthetic methods have been shown to increase the prediction performance, they sometimes generate erroneous or duplicated data instances, which leads to high false positives. They also tend to generate less diverse data points within the minority class. Exploiting these challenges, we proposed a novel over-sampling approach that systematically generates new minority samples based on their Mahalanobis distance. Motivated by the chromosomal theory of inheritance, MAHAKIL uses features from two parent instances to produce a new synthetic instance that has the characteristics of both parent instances and, thus, ensures that the synthetic data fall within the decision boundary of any classification algorithm. Our approach generates new samples by finding the average between two instances merged together based on the Mahalanobis distance disparity.

We empirically evaluated our approach by comparing it to four other over-sampling approaches (ROS, SMOTE, Borderline-SMOTE and ADASYN) using five classification models (C4.5, NNET, KNN, RF, SVM) on 20 imbalanced datasets. In total, over 200 different combinations of experiments were conducted and MAHAKIL consistently performed better than the other four approaches with regards to the performance measures. With respect to the individual prediction models, MAHAKIL was significantly better than all four approaches regarding the *pf* measure, with the RF and KNN models outperforming all of the models for all of the performance measures considered.

The performance of MAHAKIL was superior to the other methods because the data generated by the MAHAKIL algorithm inherited features from the parent data instances, which helped in correctly classifying most of the minority classes used during testing. We demonstrated in a more

practical scenario that MAHAKIL works irrespective of the sample size of the data. In future studies, we intend to modify MAHAKIL to work in local patches for multi-cluster datasets. We will also consider comparing the performance of MAHAKIL to that of the ensemble models and other complex sampling techniques.

Acknowledgments

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 11208017 and 11214116), the research funds of City University of Hong Kong (No. 7004683 and 7004474) and JSPS KAKENHI 17K00102.

References

[1] 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.

[2] 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.

[3] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "The misuse of the NASA metrics data program data sets for automated software defect prediction," in *Proc. 15th Annu. Conf. Eval. Assessment Softw. Eng.*, 2011, pp. 96–103.

[4] K. E. Bennin, J. Keung, A. Monden, Y. Kamei, and N. Ubayashi, "Investigating the effects of balanced training and testing datasets on effort-aware fault prediction models," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf.*, 2016, pp. 154–163.

[5] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Trans. Syst. Man Cybern. Part C Appl. Reviews*, vol. 42, no. 6, pp. 1806–1817, Nov. 2012.

[6] N. E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Trans. Softw. Eng.*, vol. 26, no. 8, pp. 797–814, Aug. 2000.

[7] G. M. Weiss and F. Provost, "The effect of class distribution on classifier learning: An empirical study," Rutgers Univ, 2001.

[8] K. Yoon and S. Kwek, "A data reduction approach for resolving the imbalanced data issue in functional genomics," *Neural Comput. Appl.*, vol. 16, no. 3, pp. 295–306, 2007.

[9] F. Provost, "Machine learning from imbalanced data sets 101," in *Proc. AAAI' Workshop Imbalanced Data Sets*, 2000, pp. 1–3.

[10] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, Nov./Dec. 2012.

[11] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *J. Syst. Softw.*, vol. 83, no. 1, pp. 2–17, 2010.

[12] 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.*, 2007, pp. 196–204.

[13] L. Pelayo and S. Dick, "Applying novel resampling strategies to software defect prediction," in *Proc. Annu. Meet. North Amer. Fuzzy Inform. Process. Soc.*, 2007, pp. 69–72.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artificial Intell. Res.*, 2002, pp. 321–357.

[15] H. He, et al., "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. IEEE World Congr. Comput. Intell.*, 2008, pp. 1322–1328.

[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.*, 2005, pp. 878–887.

[17] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowledge and Data Eng.*, vol. 26, no. 2, pp. 405–425, Feb. 2014.

[18] G. Y. Wong, F. H. Leung, and S.-H. Ling, "A novel evolutionary preprocessing method based on over-sampling and under-sampling for imbalanced datasets," in *Proc. IECON 2013–39th Annu. Conf. IEEE Ind. Electron. Soc.*, 2013, pp. 2354–2359.

[19] 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.

[20] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "The significant effects of data sampling approaches on software defect prioritization and classification," in *Proc. 11th Int. Symp. Empirical Softw. Eng. Meas.*.

[21] X. Hang and H. Dai, "Applying both positive and negative selection to supervised learning for anomaly detection," in *Proc. 7th Annu. Conf. Genetic Evol. Comput.*, 2005, pp. 345–352.

[22] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with precision: A response to "Comments on 'data mining static code attributes to learn defect predictors'"," *IEEE Trans. Softw. Eng.*, vol. 33, no. 9, pp. 637–640, Sep. 2007.

[23] W. S. Sutton, "The chromosomes in heredity," *Biological Bulletin*, vol. 4, no. 5, pp. 231–250, 1903.

[24] T. Aida, "On the inheritance of color in a fresh-water fish, aplocheilus latipes temmick and schlegel, with special reference to sex-linked inheritance," *Genetics*, vol. 6, no. 6, 1921, Art. no. 554.

[25] T. Gjedrem, "Genetic variation in quantitative traits and selective breeding in fish and shellfish," *Aquaculture*, vol. 33, no. 1–4, pp. 51–72, 1983.

[26] Z. Liu and J. Cordes, "DNA marker technologies and their applications in aquaculture genetics," *Aquaculture*, vol. 238, no. 1, pp. 1–37, 2004.

[27] Z. Jiu-ran, "New method of elite inbred line breeding in corn [j]," *J. Maize Sci.*, vol. 2, p. 009, 2005.

[28] P. C. Mahalanobis, "On the generalized distance in statistics," *Proc. Nat. Inst. Sci. (Calcutta)*, vol. 2, pp. 49–55, 1936.

[29] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proc. 6th Int. Conf. Predictive Models Softw. Eng.*, 2010, Art. no. 9.

[30] M. Jureczko and D. Spinellis, "Using object-oriented design metrics to predict software defects," *Models Methods System Dependability. Oficyna Wydawnicza Politechniki Wrocławskiej*, pp. 69–81, 2010.

[31] J. Sayyad Shirabad and T. Menzies, "The PROMISE repository of software engineering databases," School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[32] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Automated Softw. Eng.*, vol. 19, no. 2, pp. 167–199, 2012.

[33] A. Liu, J. Ghosh, and C. E. Martin, "Generative oversampling for mining imbalanced datasets," in *Proc. Int. Conf. Data Mining*, 2007, pp. 66–72.

[34] Y. Ma, L. Guo, and B. Cukic, "A statistical framework for the prediction of fault-proneness," *Advances Mach. Learning Appl. Softw. Eng., Idea Group Inc*, pp. 237–265, 2006.

[35] T. M. Khoshgoftaar, A. S. Pandya, and D. L. Lanning, "Application of neural networks for predicting program faults," *Ann. Softw. Eng.*, vol. 1, no. 1, pp. 141–154, 1995.

[36] F. Xing, P. Guo, and M. R. Lyu, "A novel method for early software quality prediction based on support vector machine," in *Proc. 16th IEEE Int. Symp. Softw. Rel. Eng.*, 2005, Art. no. 10.

[37] J. R. Quinlan, *C4. 5: Programs for Machine Learning*. Amsterdam, Netherlands: Elsevier, 2014.

[38] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, Jul./Aug. 2008.

[39] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[40] A. A. Shanab, T. M. Khoshgoftaar, R. Wald, and A. Napolitano, "Impact of noise and data sampling on stability of feature ranking techniques for biological datasets," in *Proc. IEEE 13th Int. Conf. Inform. Reuse Integr.*, 2012, pp. 415–422.

[41] R. C. Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Found. Statistical Comput., 2012.

[42] M. Kuhn, J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, and Z. Mayer, "caret: Classification and regression training. r package version 6.0-24," 2014, https://CRAN.R-project.org/package=caret

[43] L. Torgo, *Data Mining with R, Learning with Case Studies*. London, U.K.: Chapman and Hall/CRC, 2010. [Online]. Available: http://www.dcc.fc.up.pt/ ltorgo/DataMiningWithR
[44] M. K. Buckland and F. C. Gey, "The relationship between recall and precision," *J. Amer. Soc. Inform. Sci.*, vol. 45, no. 1, pp. 12–19, 1994.
[45] M. V. Joshi, V. Kumar, and R. C. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements," in *Proc. IEEE Int. Conf. Data Mining*, 2001, pp. 257–264.
[46] F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction," in *Proc. 10th IEEE Work. Conf. Mining Softw. Repositories*, 2013, pp. 409–418.
[47] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy, "Active learning and effort estimation: Finding the essential content of software effort estimation data," *IEEE Trans. Softw. Eng.*, vol. 39, no. 8, pp. 1040–1053, Aug. 2013.
[48] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 425–438, Mar./Apr. 2012.
[49] E. Brunner, U. Munzel, and M. L. Puri, "The multivariate nonparametric Behrens-Fisher problem," *J. Statistical Planning Inference*, vol. 108, no. 1, pp. 37–53, 2002.
[50] B. Kitchenham, "Robust statistical methods: Why, what and how: Keynote," in *Proc. 19th Int. Conf. Eval. Assessment Softw. Eng.*, 2015, Art. no. 1.
[51] H. C. Kraemer and D. J. Kupfer, "Size of treatment effects and their importance to clinical research and practice," *Biological Psychiatry*, vol. 59, no. 11, pp. 990–996, 2006.
[52] L. Breiman, "Random forests," *Mach. Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[53] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.
[54] L. Jost, "Entropy and diversity," *Oikos*, vol. 113, no. 2, pp. 363–375, 2006.
[55] J. Siegmund, N. Siegmund, and S. Apel, "Views on internal and external validity in empirical software engineering," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, 2015, pp. 9–19.
[56] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012.
[57] 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. 4, pp. 375–407, 2010.
[58] X. Tan, X. Peng, S. Pan, and W. Zhao, "Assessing software quality by program clustering and defect prediction," in *Proc. 18th Work. Conf. Reverse Eng.*, 2011, pp. 244–248.
[59] K. E. Bennin, K. Toda, Y. Kamei, J. Keung, A. Monden, and N. Ubayashi, "Empirical evaluation of cross-release effort-aware defect prediction models," in *Proc. IEEE Int. Conf. Softw. Quality Rel. Secur.*, 2016, pp. 214–221.
[60] W. Tang and T. M. Khoshgoftaar, "Noise identification with the k-means algorithm," in *Proc. 16th IEEE Int. Conf. Tools Artificial Intell.*, 2004, pp. 373–378.
[61] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in *Proc. 33rd Int. Conf. Softw. Eng.*, 2011, pp. 481–490.
[62] B. Kitchenham, et al., "Robust statistical methods for empirical software engineering," *Empirical Softw. Eng.*, pp. 1–52, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10664-016-9437-5
[63] J. C. Munson and T. M. Khoshgoftaar, "The detection of fault-prone programs," *IEEE Trans. Softw. Eng.*, vol. 18, no. 5, pp. 423–433, May 1992.
[64] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Trans. Softw. Eng.*, vol. 22, no. 10, pp. 751–761, Oct. 1996.
[65] L. C. Briand, V. Brasili, and C. J. Hetmanski, "Developing interpretable models with optimized set reduction for identifying high-risk software components," *IEEE Trans. Softw. Eng.*, vol. 19, no. 11, pp. 1028–1044, Nov. 1993.
[66] N. Ohlsson and H. Alberg, "Predicting fault-prone software modules in telephone switches," *IEEE Trans. Softw. Eng.*, vol. 22, no. 12, pp. 886–894, Dec. 1996.
[67] Q. Song, M. Shepperd, M. Cartwright, and C. Mair, "Software defect association mining and defect correction effort prediction," *IEEE Trans. Softw. Eng.*, vol. 32, no. 2, pp. 69–82, Feb. 2006.
[68] T. M. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," *Empirical Softw. Eng.*, vol. 9, no. 3, pp. 229–257, 2004.
[69] G. J. Pai and J. B. Dugan, "Empirical analysis of software fault content and fault proneness using Bayesian methods," *IEEE Trans. Softw. Eng.*, vol. 33, no. 10, pp. 675–686, Oct. 2007.
[70] O. Vandecruys, D. Martens, B. Baesens, C. Mues, M. De Backer, and R. Haesen, "Mining software repositories for comprehensible software fault prediction models," *J. Syst. softw.*, vol. 81, no. 5, pp. 823–839, 2008.
[71] A. Monden, et al., "Assessing the cost effectiveness of fault prediction in acceptance testing," *IEEE Trans. Softw. Eng.*, vol. 39, no. 10, pp. 1345–1357, Oct. 2013.
[72] M. Kubat, et al., "Addressing the curse of imbalanced training sets: one-sided selection," in *Proc. 14th Int. Conf. Mach. Learning*, 1997, pp. 179–186.
[73] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 1999, pp. 155–164.
[74] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk, "Reducing misclassification costs," in *Proc. 11th Int. Conf. Mach. Learning*, 1994, pp. 217–225.
[75] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, 2007.
[76] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Inform. Softw. Technol.*, vol. 58, pp. 388–402, 2015.
[77] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[78] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, 2002.
[79] V. García, J. S. Sánchez, and R. A. Mollineda, "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance," *Knowledge-Based Syst.*, vol. 25, no. 1, pp. 13–21, 2012.
[80] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Proc. Eur. Conf. Principles Data Mining Knowl. Discovery*, 2003, pp. 107–119.

Kwabena Ebo Bennin received the bachelor's (Hons) degree in computer science and statistics from University of Ghana, in 2011. He is an HKPFS fellow and is working toward the PhD degree in the Department of Computer Science, City University of Hong Kong. He is under the supervision of Dr. Jacky Keung. He was a visiting researcher with Okayama University, Japan working under the supervision of Professor Akito Monden. His research interests include improving the fundamentals and performances of software defect prediction models, software effort and cost estimation, data mining for software engineering datasets and techniques for bug localization. He is a student member of the IEEE.

Jacky Keung received the BSc (Hons) degree in computer science from the University of Sydney, and the PhD degree in software engineering from the University of New South Wales, Australia. He is assistant professor in the Department of Computer Science, City University of Hong Kong. His main research interests include software effort and cost estimation, empirical modeling and evaluation of complex systems, and intensive data mining for software engineering datasets. He has published papers in prestigious journals including the *IEEE Transactions on Software Engineering*, the *Empirical Software Engineering*, and many other leading journals and conferences. He is a member of the IEEE.

Passakorn Phannachitta received the BE (Hons) degree in computer engineering from Kasetsart University, Thailand, and the ME and PhD in information science from Nara Institute of Science and Technology (NAIST), Japan. He is a lecturer in the College of Arts, Media and Technology, Chiang Mai University, Thailand. His research interests include quantitative methods in empirical software engineering (ESE), software effort and cost estimation, and quality measurement and improvement of ESE datasets.

Akito Monden received the BE degree in electrical engineering from Nagoya University, in 1994, and the ME and DE degrees in information science from Nara Institute of Science and Technology (NAIST), in 1996 and 1998, respectively. He is a professor in the Graduate School of Natural Science and Technology at Okayama University, Japan. His research interests include software measurement and analytics, and software security and protection. He is a member of the IEEE, the ACM, the IEICE, the IPSJ, and the JSSST.

Solomon Mensah received the BSc degree in computer science and statistics from the University of Ghana, in 2011 and the MEng degree in computer engineering from the University of Ghana, in 2014. He is working toward the PhD degree in the Department of Computer Science, City University of Hong Kong under the supervision of Dr. Jacky Keung. His research interests include software effort estimation, moving windows, technical debt, and deep learning. He has experience in the application and development of statistical methods for Mathematical modeling, predictive modeling and software development.