iConclusions https://iconclusions.com/ Conclusions in Engineering

A Novel Resampling Technique for Imbalanced Classification in Software Defect Prediction by a re-sampling method with filtering

Kamal Bashir[1, a)] and Mohamed Mosadag[1]

Department of Information Technology, College of Computer Science and Information Technology, Karary University, Omdurman 12304, Sudan

ABSTRACT One of the greatest difficulties that most algorithms that learn classifiers face is imbalanced data. A class imbalance problem, however, is not inherently detrimental, some contemporary studies claim, nor is the performance decline solely attributable to this issue, but rather other aspects of the data distribution, such as the presence of noise and borderline cases around the class heads. In order to rectify the issue of data imbalance, the author proposes a new hybrid preprocessing technique in this study. This technique handles class imbalance, the presence of noise, and borderline samples in software defect data. To solve this problem of data imbalance and noisy samples, a combination of hyperparameter optimization based on Rough Set Theory (RST), Iterative-Partitioning Filter (IPF), and Synthetic Minority Oversampling Technique (SMOTE) oversampling is used. The strategy begins with the application of SMOTE to synthesize artificial examples through a process of linear interpolation. The first step of the algorithm is to use SMOTE followed by linear interpolation of two defect-prone k-nearest neighbors to generate synthetic examples. Then, majority-class examples with a lower approximation of these originals and newly generated minority-class examples are removed using RST. Then IPF is applied to erase the data. We analyze the efficacy of our algorithm by conducting experiments where the learning algorithm is the C4.5 classifier. Then, statistical tests show the superiority of our proposed method over state-of-the-art sampling methods.

Received: 12 October 2024 **Accepted:** 09 November 2024
DOI: https://doi.org/10.71107/1v443128

I. INTRODUCTION

By quickly identifying Defect Prone (DP) modules through metric-based classification, Software Defect Prediction (SDP) aims to increase software quality[1]. SDP makes an effort to forecast the quantity of defects as well as the defect-proneness of system components prior to their deployment in order to reduce maintenance costs for a high-quality product delivery. In order to lower costs and support software developers in their efficient resource management, an efficient method for predicting DP software modules is required for suitable testing. Additionally, this may greatly enhance the software's quality and encourage user trust and dependability throughout the supply chain. Binary prediction models are mostly used to predict a module as either defective or not in order to detect the DP software module[1]. The quality of the training data, where class imbalance is a significant problem, is partially related to these learners' capacity to classify a module appropriately. Nevertheless, actual data, such as SD, are frequently disproportionately skewed, with a small proportion of unusual or fascinating DP examples and a large proportion of conventional (NDP) ones[2]. Meanwhile, Standard machine learning techniques, mean- while, assume that the training database is fairly partitioned into a number of classes. Therefore, class imbalanced data set with about equal encompassment training class increases the performance of classifiers[3].

As it happens in imbalance learning, if it happens, the normal machine learning classifiers mostly go to favor the samples from the majority class and simply neglect the ones from the minority class. Imbalances in class data cause problems in numerous applications, such as risk analysis, fraud detection, violation prevention, and medical studies. For instance, about our SDP, a software development team needs to implement a clas-

[a)] Electronic mail: kamalbashir1@yahoo.com

sification model predicting the likelihood of defective modules in a subsequent release of the program. DP contains only about 2% of historical data; that is, only a small portion of its totality is of this type. A defect prediction module can elaborate outputs with up to 98% accuracy if it predicts each module to be NDP. This output was generated because the target DPs were not sufficiently defined in the training data, which the classification model could not identify. If a classifier can guess the minority class appropriately, the industrialists and actions will be capable of setting cheap strategies[4]. This challenge is due to the classifying models inability to determine the target DPs because their representation in the training data is too low. If a classifier is able to effectively predict the minority class, such an achievement will be beneficial to industry stakeholders and firms in implementing policies that would cut down costs. Over the last 10 years, the issue of using imbalanced data to develop classifiers and the challenges that arise have attracted much interest from researchers. Solutions to this problem have been proposed[3–15]. A very good discussion on this may be found at[16]. In order to deal with the data imbalance issue, the Synthetic Minority Over-Sampling Technique (SMOTE) is often employed. Recent studies show that class imbalance is not the only issue and that data distribution issues can also cause low performance. In particular, the problem of the presence of noisy and borderline examples is of great importance. Borderline examples are the ones found in the regions that are near the class separation line, while noisy examples are those located inside the region of other classes away from theirs. However, some features that are intrinsic to SMOTE can aggravate this problem rather than provide relief, and the extension of SMOTE as it exists currently is not well adapted to mitigate these.

As shown in Fig. 1, various classifications and differentiations have to be made in terms of safe, borderline, and noisy cases to avoid the term confusion. The regions with mostly similar class labels are the dwelling places of the safe examples. When we talk about noisy examples, it involves one class engaging in safe space within the other class. As noted in[17], it might be best to consider them as examples of what class label noise may affect. Finally, the marginal instances lie in the region close to the classification borders in between the minority and majority classes, or when the shape of the boundary is brittle. Even those cases can be shifted to the wrong side of the judgment boundary by the least attribute noise, therefore making them more difficult[17]. To the best of our knowledge, learning imbalance is not a problem in and of itself, and other factors pertaining to the data's

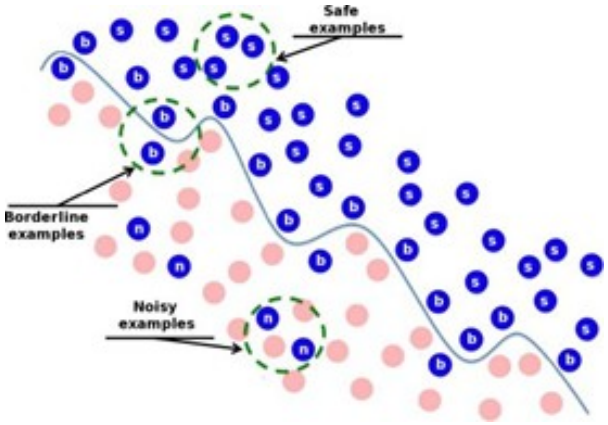

FIG. 1: The safe (s), borderline (b) and noisy (n) examples. The line represent the decision boundary among the two classes

class distribution also have an impact on the decline in performance. Among these is the presence of borderline and noisy examples. Although a great deal of research has been done to investigate, treat, and mitigate each issue of class imbalance, little has been done to examine datasets that share these challenges. Therefore, our goal is to develop a new preprocessing technique that may solve these issues by utilizing the SMOTE, Rough Set Instance Selection (RST), and Iterative-Partitioning Filter (IPF).

In this study, we introduce a new oversampling methodology: SMOTE generates synthetic examples. RST any synthetic instance that does not fall into the lower approximation of the minority class as noisy or outliers in the boundary region and thus not useful for classification. Fixedly, we utilize the IPF to wash the noise out of the whole set. This process, referred to as SMOTE-RSTNF, will take care of two major challenges: They analyzed the misleading effects from (1) the imbalanced distribution of classes, which is addressed by SMOTE oversampling, and (2) the data quality, where the post-processing removes noisy, borderline, and irrelevant data examples. The rest of the work is divided with a view. Section II of the study is the Review of Literature. Section III describes the preprocessing technique of this chapter violently. The experiment is explained in detail in Section V of this paper. The final section of this research is Section IV, where the findings of the study will be discussed and analyzed. Section VI, on the other hand, will provide a conclusion and suggestions for future research

II. LITERATURE REVIEW

Scholars have noted that searching for a practical solution for misclassification is almost always a challenge because it is more likely to occur in the overlap area or near decision boundaries, as this work confirmed [18]. For instance, Napierala et al.[18] showed in a number of experiments that the number of borderline samples has a direct influence on the classifier's degradation in an imbalanced scenario. Two distinct techniques have been taken by the literature in an attempt to address the aforementioned issue.

A. Modifications of SMOTE

change-direction techniques that are associated with SMOTE modifications. These direct SMOTEs generate positive examples toward specified areas of the data space as well as taking into account particular data features. This category includes the following methods: ADMOS[19], ADASYN[20], Borderline-SMOTE[21] and Safe level SMOTE[22]. These techniques are intended to generate positive examples only within the region of positive class or around regions where the density of positive examples is high.

B. Extensions of SMOTE

According to the results of the research, the results of further data pretreatment methods combined with SMOTE include filtering. Usually, noise filters are used in ordinary classification problems to screen out the possibly noisy samples and make the classification boundaries more clear and definite when defining the training sets[17]. In the empirical analysis of the performance of filters to identify the behavior of data balance methods for the training of computer ML data[23]. The usefulness of integrating filters with oversampling approaches is established. Some of the generals of SMOTE are SMOTE-RSB[24], SMOTE- FRST[25], SMOTE-Tomek Links (TL)[26], SMOTE- ENN [Edited Nearest Neighbor Rule (ENN)][27], and SMOTE-IPF[17] in which a form of filtering takes place after the SMOTE operation. Tumar et al.[28] proposed an SDP model in which BMFO is combined with an ADASYN to overcome data imbalance problems. When used on PMB Poland's PROMISE dataset, the suggested strategy improves the results of numerous classifiers.

Rathore et al.[29] created three independent generative oversampling techniques: Other models that we compare our results against include conditional generative adversarial networks, also known as CTGAN, vanilla GAN, and Wasserstein GAN with Gradient Penalty, known as WGANGP. This experiment is carried out on PROMISE, JIRA, and Eclipse datasets. If these sampling strategies are used with baseline models in fault assessment datasets, the baseline model results are significantly improved. For intra-release and cross-release SDP, non-linear and linear Bayesian regression have been carried out by Singh and Rathore[30]. Therefore, integrated with SMOTE data sampling methods, Random Forest (RF), Support Vector Support Vector Machine (SVM), Linear Regression (LR), Linear Bayesian Regression (LBR), and Non-linear Bayesian Regression (NLBr). The study demonstrated that, in an independent software product dataset of 46 products, the Bayesian non-linear model outperforms the linear regression model algorithms. Elahi et. al.[31], carried out a study whereby a number of ensemble methods used in SDP were analyzed. For classification, applied are Logistic Regression LR, Naive Bayes NB, binomial producer multinomial NB, Decision Tree DT, and K-nearest neighbor KNN. This experiment is performed on four datasets from the PROMISE repository. This paper uses F-measure as the performance measure. The findings about model averaging outperform voting and stacking ensemble approaches were derived directly from the data. He et al. proposed SHSE[32]. In fact, it can be concluded that CSS is a mixture between different sampling, feature subspace, and ensemble learning. The problems of data imbalance are solved with the help of the Subspace Hybrid Sampling approach. When conducting experiments on 27 datasets, SHSE performed better than any other algorithms used in software failure number prediction. DT performs the best when implemented together with SHSE.

To address data imbalance issue in SDP, Goyal[33] proposed an innovative sampling technique known as neighborhood-based undersampling (N-US). ANN, DT, KNN, SVM, and NB classifier models are used in the modeling process. The study also makes use of the PROMISE dataset. For the measurement of the model's performance accuracy, AUC and ROC are being used. As it has been evidenced, when acting in accordance with the N-US approach, the classifiers' accuracy increases. Similar, Pandey et al.[34] also used NASA and the PROMISE repository for SDP. To address the problem of data imbalance, the feature selection procedure SMOTE combined with Kernel Principal Component Analysis (K-PCA) is further used on the dataset in order to exclude features that are irrelevant to the circumstance. Compared with conventional methods of fault prediction, using rebooted NB, LR, Multi-Layer Perceptron Neural Network (MLP), SVM, as well as conventional K-PCA and SMOTE methods that are incorporated with Extreme Learning Machine and PCA-ELM

have technically higher ROC indices in this study. The recommended technique provides more objective results as compared to other reliable methods. To overcome the data imbalance problem, Yedida and Menzies[35] have put forward a new oversampling technique called fuzzy sampling. An SDP model is developed by means of a deep belief network. The experiment is implemented on the PROMISE dataset. It uses AUC, RUC, and false alarm rates to assess the performance of the methodologies. Thus, the authors find that oversampling is necessary before applying deep learning for SDP.

Pandey et al.[36] have done experiments on raw NASA datasets to detect software faults. It is seen that the dataset is highly imbalanced, and thus the SMOTE technique is applied to them. For the balanced dataset, the SqueezeNet and Bottleneck DL models are used. Tantithamthavorn et al.[37] applied four class imbalancing procedures: sampling techniques including oversampling, undersampling, SMOTE, and Random Oversampling Examples (ROSE) associated with NB, AVNNet, xGBTree, C5.0, RF, LR, and GBM classification algorithms. The study revealed that to optimize the parameters of SMOTE, AUC might be improved. In case of defect prediction, Nitin et al.[38] used four ensemble techniques: random forest, bagging, random subspace, boosting, and SMOTE for handling imbalance data. The ensemble techniques employ the use of DT, LR, and KNN as base learners. In the experiment, fifteen datasets from the Eclipse and PROMISE repositories are used. A model has been proposed by Balaram and Vasundra[39] also known as BOA combined with E-RF linked to ADASYN. PROMISE dataset is used in the study. BOA is used to address the overfitting issue, and to address the class imbalance issue, ADASYN is employed. The evaluations in specificity, AUC, and sensitivity claimed that this proposed E-RF-ADASYN is slightly higher than KNN and DT classifiers. Table I shows some existing SDP models that used balancing techniques

III. THE PROPOSED METHOD

In this section, we introduce the SMOTE-RSTNF technique for improving the SMOTE algorithm, in which every step removes noisy and borderline instances that can deteriorate learning performances. In the suggested algorithm, new synthetic minority class instances are first added to the training set using SMOTE. Synthetic instances, or majority class instances, are then removed using the IPF filter, which eliminates noisy examples from both the dataset and those produced by SMOTE. All of this is done by using RST and the lower approximation of a subset.

A. Synthetic Minority Over-Sampling Technique (SMOTE)

A vast majority of the under-sampling and minority over-sampling methods have been discussed elaborately in this literature on data sampling methods. This work employed the SMOTE[2], an algorithm that generates new synthetic instances in minority classes. They do so not in data space, but in feature space where they create the synthetic instances. The SMOTE instances are given by $S = S + u \times (X' - X)$ with $0 \leq u \leq 1\$$, where $(X \; and \; X')$ are two similar samples belonging the minority class. X' is randomly selected from K neighborhoods of X in the minority class. The construction of the more recent examples expands the fullness and generality of the minority while reducing the rarity of the occurrence.

B. Rough Set Analysis

Data is represented as an information system, or$S = (X, A)$, in rough set analysis[40], where $X = \{x_1, \ldots, x_n\}$ and $A = \{a_1, \ldots, a_m\}$ are the non-empty and finite sets of objects and attributes, respectively. There is a mapping $a : X \to V_a$ for every $a \in A$, where V_a is the value set of attribute a. The B-indiscernibility relation R_B is defined with regard to any subset$B \subseteq A$.

$$R_B = \{(x, y) \in X^2 : a(x) = a(y), \forall a \in B\} \tag{1}$$

R_B is an equivalence relation, which can create a partition of the universal$X$, denoted as X/R_B. $[x]_{R_B} = \{y \in X : (x, y) \in R_B\}$ is the equivalence class of x and$X/R_B = \{[x]_{R_B} : x \in X\}$. Given$U \subseteq X$, the lower and upper approximation w.r.t. R_B are determined by

$$R_B \downarrow U = \{x \in X | [x]_{R_B} \subseteq U\} \tag{2}$$

$$R_B \uparrow U = \{x \in X | [x]_{R_B} \cap U \neq \emptyset\} \tag{3}$$

In the context of classification, a decision system $(X, A \cup \{d\})$ is a unique type of information system, where the designated attribute $d (d \notin A)$ is called the decision attribute. In $X/R_d = \{[x]_{R_d} : x \in X\}$, the decision classes with respect to d are given. Given $B \subseteq A$, the objects from X for which the values of B allow for unambiguous prediction of the decision class are included in the B-positive areaPOS_B:

$$POS_B = \bigcup_{x \in X} R_B \downarrow [x]_{R_d} \tag{4}$$

In fact, if$x \in POS_B$, then any object that shares the same values with x for any attribute in B will likewise be

a member of the same decision class as x. The following number (degree of reliance of d on B) represents the predictive ability w.r.t. d of the qualities in B:

$$\gamma_B = \frac{|POS_B|}{|X|} \tag{5}$$

$(X, A \cup \{d\})$ is called consistent if $\gamma_A = 1$. If a subset B of A meets these requirements, it's referred to as a decision reduct. (1) $POS_B = POS_A$, meaning that B maintains $A's$ ability to make decisions. (2) It is not reducible further, that is, $POS_{B'} = POS_A$ does not exist for any suitable subset $B^{'}$ of B . We refer to B as a choice superreduct if the latter requirement is lifted, that is, if B is not necessarily minimum.

C. Iterative Partitioning Filtering Based Noise Filtering

IPF arising due to the excellent work by[41] The method of IPF eliminates a number of noisy cases in a number of iterations prior to reaching a certain threshold. The iterative method stops when, for a number of consecutive iterations k, the number of mistakenly noisy examples in each of them is less than the percentage p of the initial training data set. Initially, the approach starts with a set of noisy instances A = 0. The basic steps of each iteration are as follows:

- Divide the given training dataset denoted as E into further n equally small subsets.
- Create a classifier by the C4.5 method on each of these n subgroups and use them for the evaluation of the entire current training data set E.
- Overlaid with B in A, the noisy cases are identified in E through the voting system (consensus or majority).
- Remove the noisy examples: $E = E \setminus A$.

Two voting techniques can be employed to identify noisy examples: consensus and majority. The former removes an example if it is misclassified by all the classifiers, while the latter eliminates an example if it is misclassified by more than half the classifiers. The parameter setting for the implementation of IPF used in this study has also been found to estimate the degree of balance and the level of noisy and borderline samples on imbalanced datasets once preprocessed with SMOTE. In fact, the majority approach is applied to recognize the noisy examples; the number of partitions with random examples is n = 9, k = 3 iterations for the stop criterion, and p = 1% of deleted examples. Scholarly investigations into the effect of those parameters in the findings are used to determine this parameter configuration.

IV. EXPERIMENTAL DESIGN

In this work, the data is divided into ten mutual choice subsets (the folds) of nearly the same measure as the training datasets are divided randomly. All the nine folders are chosen in the 10 fold procedure to train both of the models and then testing it on the separate ninth fold. This continues until all of the folds have been utilized per their ability for testing or training alternatively. Empirical data is generated and analyzed by using computation programs WEKA version 3.8.1, MATLAB R2016a, the KEEL software tool, and the R statistical program. in this work, the C4.5 learning algorithms are applied with the use of WEKA tools with the default settings. The AUC statistic determines the accuracy of picture painted at how well the constructed models in terms of categorization. Datasets: All the datasets in the experiment except for JDT and PDE that are mined from[42], were derived from public software project data repositories[43]. Attached is Table II that gives features of information. The characteristics of the data are presented in Table II.

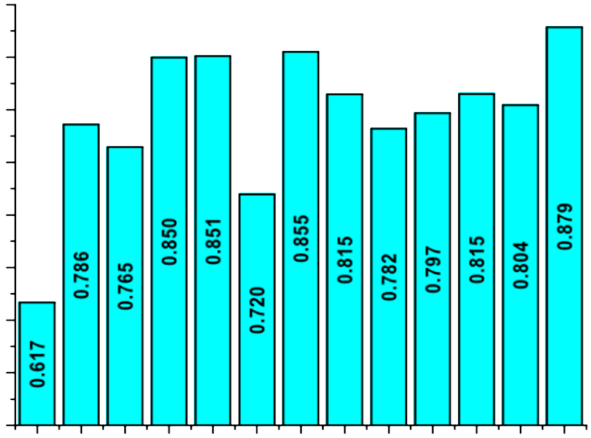

FIG. 2: Average AUC over all datasets for the C4.5 classifier.

V. RESULTS AND DISCUSSION

To perform an evaluation of the results obtained in this paper, several statistical tests are run to compare our method with the one without preprocessing and also compare it with eleven other preprocessing techniques selected from the literature.

The outcomes of the experimental investigation for the test partitions are summarized in Table V, where the best approach is highlighted for each data set and the one overall is in the first row. Because the proposed

TABLE II: Characteristics of datasets.

Defect Ratio	NFP	FP	#Attribute	#Modules	Datasets
3.48	579	166	21	745	ant-1.7
7.67	207	27	21	234	arc
1.97	268	136	62	404	EQ
3.65	6110	1672	22	7782	jm1
9.79	627	64	62	691	LC
6.16	1288	209	62	1497	PDE

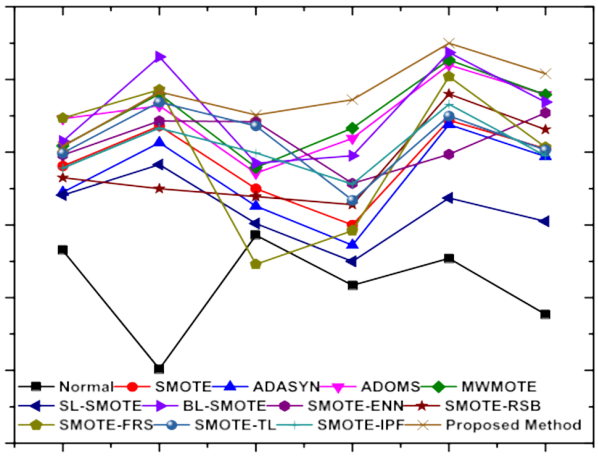

FIG. 3: The AUC classification results over all datasets for the C4.5 classifier.

TABLE V: Average ranks obtained by each method in the Friedman tes.

Algorithm	Ranking
Proposed method	2
BL-SMOTE	3.1667
MWMOTE	3.5833
ADOMS	4.0833
SMOTE-FRS	6.3333
SMOTE-TL	6.3333
SMOTE-ENN	6.1667
SMOTE-IPF	7.5
SMOTE	8.3333
SMOTE-RSB	8.5
ADASYN	10.5
SL-SMOTE	11.6667
Normal	12.8333

method outperforms the other methods in AUC measures in four of the six software defect datasets, we can understand how efficient it is. Moreover, the improvements reached by BL-SMOTE and MWMOTE against the purely SMOTE present how essential the cleaning phase in the oversampling process is in order to enhance the behavior at the moment of the classification. As shown in Figs. 3 and 2, all the preprocessing techniques in the end surpass the result using the original data sets as expected. The algorithm ratings for each of the original data sets and the ones selected for this analysis are indicated in Table III. As the reader can see, our idea wins first place four times, third place once, and fifth place once.

The results will be tested with an additional multiple comparison to compare methods and distinct preprocessing procedures. Table IV provides the comparison of the performance of the different methods of our proposal and compared methods, which indicates our method has the best ranking compared, while the two last positions correspond to the Borderline-ADASYN Rank and SL-SMOTE Rank. For the current study, a p-value of nearly zero must be found in order to detect a statistical difference between the methods. A multiple comparison test will be used to compare the results attained and identify the most effective preprocessing method. As also shown in Table III, our proposal obtains the best ranking, with Borderline-ADASYN and SL-SMOTE ranking last. In this investigation, therefore, it is necessary to use a p-value close to zero in order to detect any significant difference between the approaches.

A. Statistical Test

The Holm process results for comparing our proposal with the other ones are presented in Table VI. Thirdly, the z-value obtained for each of the equations helps in identifying the order of the algorithms. Since the test rejects all cases, the equivalent p-values for each comparison can be done with normal distribution. This can then be compared to the other related in the corresponding row of the table, either to reject the hypothesis of equal behavior in favor of the best ranking algorithm. This work demonstrated consistently that our strategy is better than every other strategy that we analyzed statistically.

TABLE I: Summary of SDP Models Handling Class Imbalance Issues

Author	Dataset	ML Algorithm	Sampling Techniques
Tumar et al., 2019[28]	PROMISE	KNN, DT, LDA NB, RF, Dynamic Adaboost	Adaptive synthetic sampling
Elahi et. Al, 2020[31]	PROMISE	ANN, DT, KNN, SVM, NB	EasyEnsemble (Ensemble random under-sampling)
Goyal, 2021[33]	NASA	NB, RF, LR, DT, KNN	Neighbourhood based under-sampling (N-US)
Rathore et al., 2022[29]	ECLIPSE, JIRA, PROMISE	NB, RF, LR, DT, KNN	GAN based oversampling, Wasserstein GAN with Gradient Penalty (WGANGP), Conditional GAN (CTGAN)
Singh and Rathore, 2022[30]	PROMISE, AEEEM, JIRA AEEEM, MetricsRepo	RF, SVM, LR, LBR, NLBR DT Regressor K-PCA-ELM, SVM, LR, Naive Bayes, MLP Deep belief network	SMOTE
Tong et al., 2022[32]			Random under sampling, SMOTER
Pandey et al., 2020[34]	PROMISE, NASA		SMOTE
Yedida and Menzies, 2021[35]	PROMISE		Fuzzy
Pandey et al., 2022[36]	NASA	SqueezeNet, BottleNeck RF, LR, NB, GBM, C5.0, Boosting, NNixGBTree, AVNNet	SMOTE
Tantithamtha et al., 2020[37]	NASA, PROMISE	SVM, NB	Over sampling, SMOTE, Under sampling, ROSE random under-sampling)
Nitin et al., 2021[38]	PROMISE, ECLIPSE	Ensemble Learning	SMOTE
A. Balaram and Vasundra, 2022[39]	PROMISE	Ensemble RF	Adaptive synthetic sampling

TABLE III: AUC Classification Results Over All Datasets for the C4.5 Classifier.

Dataset	Normal	SMOTE	ADASYN	ADOMS	MWMOTE	SL-SMOTE	BL-SMOTE	SMOTE-ENN	SMOTE-RSB	SMOTE-FRS	SMOTE-TL	SMOTE-IPF	Proposed Method
ant-1.7	0.665	0.781	0.745	0.846	0.809	0.741	0.815	0.796	0.765	**0.847**	0.799	0.778	0.808
arc	0.502	0.836	0.813	0.864	0.88	0.783	**0.931**	0.843	0.75	0.856	0.869	0.833	0.883
EQ	0.686	0.75	0.726	0.771	0.779	0.702	0.785	0.842	0.739	0.646	0.836	0.799	**0.851**
jm1	0.617	0.7	0.672	0.819	0.833	0.695	0.795	0.757	0.728	0.692	0.734	0.756	**0.872**
LC	0.654	0.844	0.838	0.92	0.927	0.737	0.937	0.797	0.808	0.904	0.835	0.866	**0.95**
PDE	0.577	0.805	0.794	0.879	0.879	0.705	0.869	0.854	0.831	0.807	0.803	0.85	**0.908**
Average	0.617	0.786	0.765	0.85	0.851	0.72	0.855	0.815	0.782	0.797	0.815	0.805	**0.879**
Winner	0	0	0	0	0	0	1	0	0	1	0	0	4

TABLE IV: Performance ranking for the preprocessing algorithms.

Dataset	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th
ant-1.7	SMOTE-FRS	ADOMS	BL-SMOTE	MWMOTE	**Proposed Method**	SMOTE-TL	SMOTE-ENN	SMOTE	SMOTE-IPF	SMOTE-RSB	ADASYN	SL-SMOTE	Normal
arc	BL-SMOTE	SMOTE-FRS	**Proposed Method**	MWMOTE	SMOTE-TL	ADOMS	SMOTE-ENN	SMOTE	SMOTE-IPF	ADASYN	SL-SMOTE	SMOTE-RSB	Normal
EQ	**Proposed Method**	SMOTE-ENN	SMOTE-TL	SMOTE-IPF	BL-SMOTE	MWMOTE	ADOMS	SMOTE	SMOTE-RSB	ADASYN	SL-SMOTE	Normal	SMOTE-FRS
jm1	**Proposed Method**	MWMOTE	ADOMS	BL-SMOTE	SMOTE-ENN	SMOTE-IPF	SMOTE-TL	SMOTE-RSB	SMOTE	SMOTE-FRS	ADASYN	SL-SMOTE	Normal
LC	**Proposed Method**	BL-SMOTE	MWMOTE	ADOMS	SMOTE-FRS	SMOTE-RSB	SMOTE-IPF	SMOTE-TL	SMOTE	ADASYN	SMOTE-ENN	SL-SMOTE	Normal
PDE	**Proposed Method**	ADOMS	MWMOTE	BL-SMOTE	SMOTE-ENN	SMOTE-RSB	SMOTE-FRS	SMOTE	SMOTE-TL	SMOTE-IPF	ADASYN	SL-SMOTE	Normal

B. SMOTE-RSTNF: suitability of the approach, strengths and weaknesses

The noise examples' iterative removal is precisely the feature that makes IPF + RST fit to unbalanced datasets preprocessed with SMOTE containing some

TABLE VI: Post Hoc comparison for A= 0:05, our proposed technique is the control method.

i	algorithm	z = (R0 − Ri)/SE	p	Holm
12	Normal	4.818121	0.000001	0.004167
11	SL-SMOTE	4.299246	0.000017	0.004545
10	ADASYN	3.780372	0.000157	0.005
9	SMOTE-RSB	2.890872	0.003842	0.005556
8	SMOTE	2.816747	0.004851	0.00625
7	SMOTE-IPF	2.446123	0.01444	0.007143
6	SMOTE-FRS	1.927248	0.053949	0.008333
5	SMOTE-TL	1.927248	0.053949	0.01
4	SMOTE-ENN	1.853123	0.063865	0.0125
3	ADOMS	0.926562	0.354154	0.016667
2	MWMOTE	0.704187	0.481316	0.025
1	BL-SMOTE	0.518875	0.603848	0.05
Holm's procedure rejects those hypotheses which the p-value ≤ 0:007143				

noise and borderline examples. This fact indicates more efficient noise filtering since instances that were discarded in the first stage do not interfere with the detection process at the next stage. Furthermore, in this case, the ensemble nature of IPF with RST allows it to pool together predictions made by different classifiers, leading to a better estimation of hard-to-classify noisy samples than pooling data from a single classifier.

IPF + RST might be effective in terms of removing noise and borderline samples because such characteristics seem to be their main strength factors rather than another lower-ranked noise filter's versatility. Most of the noise filters empowered with SMOTE, such as ENN or TL, define noise around two nearest neighbor instances but not around two different classes. Even though they have not received attention in the literature, this issue could be a source of embarrassment: since SMOTE plots the position of a new positive example relative to a nearest neighbor, these synthetic examples, which are faulty, are likely to escape detection by noise filters, which are based on superfluous nearest neighbors. While they will see some exceptions that have been classified as noise, noise identification methods such as IPF and RST that are based on more complex criteria enable sample sizes that have similar characteristics to be grouped. This solves the problem mentioned earlier and helps in quite easily spotting the outliers. Among all its parameters, the ones for IPF's selection can be termed as one of its major weaknesses, as there are many parameters and their values dictate most of the filter's performance. By way of our numerous experiments, then, we may draw some conclusions as to the effects of the different parameters on the performance results. Since there are enough noisy as well as borderline examples with regard to the number of safe examples, we have confirmed that the majority method is superior to the consensus scheme. The consensus scheme has been noted to be quite conservative with regard to deleting examples, and it does not permit deletion to the extent that it can significantly improve the performance.

VI. CONCLUSION

When it comes to learning from imprecise data, the presence of noise and outliers is an art and still an active area of research. In this work, we proposed SMOTE-RSTNF, a combined preprocessing technique for unbalanced multifaceted data. With the former, we employed the RST to get rid of the original majority samples and synthetic samples that were not part of the lower approximation of the class after applying SMOTE to generate the samples. The IPF is then applied to clean up all of the data. Employing real-world software datasets, the suggested methodologies were employed to develop classifiers that could identify malfunctioning modules. The techniques were tested using the C4.5 classifier, and the outcome was stored using the AUC performance metric. The outcome of the experiment proves that the use of our proposed approach performed better than the existing methods reported in the current literature. The signed-rank test conducted by the author revealed that the results for the Friedmans scheme preferred the strategies suggested came out to be statistically significant for the independent experimental investigations. In our further work, we plan to address other problems associated with data, like class overlapping, which is not considered in our work, and taking it into consideration may enhance the performance more. Also, we plan to apply boosting to bring greater improvements to implementing the

suggested method for SDP.

DECLARATION OF COMPETING INTEREST

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

REFERENCES

[1]S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," IEEE Transactions on Software Engineering **34**, 485–496 (2008).

[2]N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research **16**, 321–357 (2002).

[3]K. Bashir, T. Li, C. W. Yohannese, and Y. Mahama, "Enhancing software defect prediction using supervised-learning based framework," in *Proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)* (IEEE, 2017) pp. 1–6.

[4]K. Bashir, T. Li, C. W. Yohannese, and M. J. Yahaya, "Smotefris-inffc: Handling the challenge of borderline and noisy examples in imbalanced learning for software defect prediction," Journal of Intelligent & Fuzzy Systems **38**, 917–933 (2020).

[5]K. Bashir, T. Li, and C. W. Yohannese, "An empirical study for enhanced software defect prediction using a learning-based framework," International Journal of Computational Intelligence Systems **12**, 282–298 (2018).

[6]C. W. Yohannese and T. Li, "A combined-learning based framework for improved software fault prediction," International Journal of Computational Intelligence Systems **10**, 647–662 (2017).

[7]M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42**, 463–484 (2011).

[8]T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction," in *Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, Vol. 1 (IEEE, 2010) pp. 137–144.

[9]D. Van Nguyen, K. Ogawa, K.-i. Matsumoto, and M. Hashimoto, "Editing training sets from imbalanced data using fuzzy-rough sets," in *Artificial Intelligence Applications and Innovations: 11th IFIP WG 12.5 International Conference, AIAI 2015, Bayonne, France, September 14–17, 2015, Proceedings* (Springer, 2015) pp. 115–129.

[10]C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Rusboost: A hybrid approach to alleviating class imbalance," IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans **40**, 185–197 (2009).

[11]C. W. Yohannese, T. Li, M. Simfukwe, and F. Khurshid, "Ensembles based combined learning for improved software fault prediction: A comparative study," in *Proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)* (IEEE, 2017) pp. 1–6.

[12]S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," IEEE Transactions on Reliability **62**, 434–443 (2013).

[13]C. W. Yohannese, T. Li, and K. Bashir, "A three-stage based ensemble learning for improved software fault prediction: An empirical comparative study," International Journal of Computational Intelligence Systems **11**, 1229–1247 (2018).

[14]K. Bashir, T. Li, C. W. Yohannese, M. Yahaya, and T. Ali, "A novel preprocessing approach for imbalanced learning in software defect prediction," in *Data Science and Knowledge Engineering for Sensing Decision Support: Proceedings of the 13th International FLINS Conference (FLINS 2018)* (World Scientific, 2018) pp. 500–508.

[15]K. Bashir, S. Pirasteh, H. Abdelrhman, M. Mosadag, and A. Mohammed, "An enhanced feature selection approach for breast cancer prediction using a hybrid framework," Journal of Karary University for Engineering Science (2024), dOI not available.

[16]H. He and E. A. Garcia, "Learning from imbalanced data," IEEE Transactions on Knowledge and Data Engineering **21**, 1263–1284 (2009).

[17]J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," Information Sciences **291**, 184–203 (2015).

[18]K. Napierała, J. Stefanowski, and S. Wilk, "Learning from imbalanced data in presence of noisy and borderline examples," in *Rough Sets and Current Trends in Computing: 7th International Conference, RSCTC 2010, Warsaw, Poland, June 28-30, 2010. Proceedings* (Springer, 2010) pp. 158–167.

[19]S. Tang and S.-P. Chen, "The generation mechanism of synthetic minority class examples," in *2008 International Conference on Information Technology and Applications in Biomedicine* (IEEE, 2008) pp. 444–447.

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

[21]H. Han, W. Wang, and B. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *International Conference on Intelligent Computing* (Springer, 2005) pp. 878–887.

[22]C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Advances in Knowledge Discovery and Data Mining* (Springer, 2009) pp. 475–482.

[23]V. García, J. Sánchez, and R. A. Mollineda, "An empirical study of the behavior of classifiers on imbalanced and overlapped data sets," *Progress in Pattern Recognition, Image Analysis and Applications*, , 397–406 (2007).

[24]E. Ramentol, Y. Caballero, R. Bello, and F. Herrera, "Smote-rsb*: A hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory," Knowledge and Information Systems **33**, 245–265 (2012).

[25]E. Ramentol, N. Verbiest, R. Bello, Y. Caballero, C. Cornelis, and F. Herrera, "Smote-frst: A new resampling method using fuzzy rough set theory," in *Uncertainty Modeling in Knowledge Engineering and Decision Making* (World Scientific, 2012) pp. 800–805.

[26]G. E. Batista, A. L. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: A case study," WIT Transactions on Information and Communication Technologies **29**, 10–18 (2003).

[27]G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," ACM SIGKDD Explorations Newsletter **6**, 20–29 (2004).

[28]I. Tumar, Y. Hassouneh, and H. Turabieh, "Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction," IEEE Access **8**, 8041–8055 (2020).

[29]S. S. Rathore, S. Chouhan, D. Jain, and A. Vachhani, "Generative oversampling methods for handling imbalanced data in software fault prediction," IEEE Transactions on Reliability **71**, 747–762 (2022).

[30]R. Singh and S. S. Rathore, "Linear and non-linear bayesian regression methods for software fault prediction," International Journal of System Assurance Engineering and Management **13**, 1864–1884 (2022).

[31]E. Elahi, S. Kanwal, and A. N. Asif, "A new ensemble approach for software fault prediction," in *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)* (IEEE, 2020) pp. 407–412.

[32]H. Tong, W. Lu, W. Xing, B. Liu, and S. Wang, "Shse: A subspace hybrid sampling ensemble method for software defect number prediction," Information Software Technology **142**, 106747 (2022).

[33]S. Goyal, "Handling class-imbalance with knn (neighbourhood) under-sampling for software defect prediction," Artificial Intelligence Review **55**, 2023–2064 (2022).

[34]S. K. Pandey, D. Rathee, and A. K. Tripathi, "Software defect prediction using k-pca and various kernel-based extreme learning machine: an empirical study," IET Software **14**, 768–782 (2020).

[35]R. Yedida and T. Menzies, "On the value of oversampling for deep learning in software defect prediction," IEEE Transactions on Software Engineering **48**, 3103–3116 (2021).

[36]S. K. Pandey, A. Haldar, and A. K. Tripathi, "Is deep learning good enough for software defect prediction?" Innovations in Systems Software Engineering , 1–16 (2023).

[37]C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," IEEE Transactions on Software Engineering **46**, 1200–1219 (2018).

[38]Nitin, K. Kumar, and S. S. Rathore, "Analyzing ensemble methods for software fault prediction," in *Advances in Communication and Computational Technology: Select Proceedings of ICACCT 2019* (Springer, 2021) pp. 1253–1267.

[39]A. Balaram and S. Vasundra, "Prediction of software fault-prone classes using ensemble random forest with adaptive synthetic sampling algorithm," Automated Software Engineering **29**, 6 (2022).

[40]Z. Pawlak, "Rough sets," International Journal of Computer & Information Sciences **11**, 341–356 (1982).

[41]T. M. Khoshgoftaar and P. Rebours, "Improving software quality prediction by noise filtering techniques," Journal of Computer Science and Technology **22**, 387–396 (2007).

[42]T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, "The promise repository of empirical software engineering data," (2012), accessed: 2025-02-10.

[43]M. D'Ambros, M. Lanza, and R. J. E. S. E. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," Empirical Software Engineering **17**, 531–577 (2012).