Towards a New Hybrid Synthetic Minority Oversampling Technique for Imbalanced Problem in Software Defect Prediction

2024 5th International Conference on Data Analytics for Business and Industry (ICDABI) | 979-8-3503-6871-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDABI63787.2024.10800331

Yakub Kayode Saheed
School of IT& Computing
American University of Nigeria,
Adamawa State, Nigeria
0000-0002-0804-0707

Sulaiman Olaniyi Abdulsalam
Dept. of Computer Science
Kwara State University
Malete, Nigeria
sulaiman.abdulsalam@kwasu.edu.ng

Mohammed Babatunde Ibrahim
ICT Unit
Nigerian Extractive Industries Transparency Initiative
Abuja, Nigeria
mibrahim@neiti.gov.ng

Usman Ahmad Baba
Dept. of Computer Science
North-Eastern University
Gombe, Nigeria
usman.baba@neu.edu.ng

Abstract*—The software industry strives to improve software quality through continuous bug prediction, bug elimination, and module fault prediction. This issue has piqued researchers' interest because of its significant relevance in the software industry. Frequently, Software Defect Prediction (SDP) models contain significantly skewed data, making it difficult for classifiers to recognize defective occurrences. The machine learning (ML) community has put a lot of effort into solving the problem of learning from imbalanced SDP data, though less so in empirical software engineering. The over-sampling strategy is one of many recent solutions to this problem. This strategy balances the number of defective and non-defective cases by creating new defective instances. Unfortunately, these methods would result in non-diverse synthetic instances as well as a large number of unneeded noise instances, creating an imbalanced class problem. As a result, we propose a hybrid synthetic minority oversampling (HSMOTE) to address the problem of imbalance in SDP. In this paper, we introduce the hybrid Synthetic Minority Oversampling Technique (HSMOTE), a method that utilizes Extra Tree, Random Forest (RF), and Extreme Gradient Boosting (Xgboost) for classification. We develop and deploy the proposed method on the National Aeronautics and Space Administration (NASA) dataset, evaluating its performance on three datasets: JM1, KC1, and PC3. We compared the parameters for accuracy, precision, AUC, recall, F-measure, and Mathew Correlation Coefficient to those of the existing SDP. The findings from the simulations on the JM1 data showed that the proposed techniques work better than the current best models. The proposed SMOTE+RF technique surpasses the existing techniques with an accuracy of 93.69%, an AUC of 82.70%, and an F-measure of 32.98%. Similarly, the proposed SMOTE+Xgboost method outperforms the existing techniques with an accuracy of 93.432%, an AUC of 82.64%, and an F-measure of 34.13%, while SMOTE+ET achieved an accuracy of 93.43%, an AUC of 77.68%, and an F-measure of 31.90%.

Keywords—Synthetic Minority Oversampling Technique, Class Imbalance, Oversampling Method, Software Defect Prediction, Imbalance Data

I. INTRODUCTION

Software testing is an imperative stage in the lifecycle of software development for ensuring software quality[1]. By identifying possibly problematic software components before delivering the software products, software defect prediction (SDP) can assist quality assurance teams in allocating their limited testing resources more prudently. Thus, efficient defect prediction can help save money on testing and increase the quality of your program [2].

The purpose of SDP is to forecast defects before the deployment of software products, as finding bugs after the fact is a tiring and time-consuming task. Additionally, it has been established that SDP techniques increase software quality by assisting engineers in predicting the most likely problematic modules[3]. Because SDP is seen as a significant difficulty, many ML methods have been utilized to anticipate and identify defective modules [7]. SDP is used to differentiate problematic modules in current and future versions of a software product to increase the effectiveness of software testing. As a result, SDP techniques are extremely beneficial in directing more effort and resources to the testing and scrutinizing modules that are likely to be problematic [4]. By utilizing machine learning (ML) techniques [5], SDP systems may detect the greatest number of problematic modules[6]. When the balance of these training datasets is achieved, these machine learning approaches may achieve good prediction performance [7]. However, in software projects, there are a larger amount of non-defective examples than defective samples, resulting in a class-imbalanced difficulty in SDP. ML methods learned on these class-imbalanced data may exhibit a bias toward non-defective examples and may identify defective instances incorrectly [8]. As a result, the imbalanced class problem is regarded as a critical aspect influencing the efficiency of SDP, and an increasing number of scientists are determined to resolve it [9]. The process of building and bringing a high-quality, bug-free software product to end customers has become increasingly difficult in recent years. Software reliability and quality assurance helps ensure that end customers receive a high-quality software product [10].

According to[11], a software product's quality is closely connected with the existence or absence of defects. In a computer program, a software fault manifests as an error

Authorized licensed use limited to: The University of Texas Rio Grande Valley. Downloaded on January 10,2025 at 13:59:04 UTC from IEEE Xplore. Restrictions apply.

scenario produced by improper requirements, improper program design logic, or a lack of coding and testing abilities, among other factors [6–8]. A software defective unit prevents the software after performing as intended, increasing design and support costs and contributing to customer discontent [12]. Owing to the high imbalance, high dimensionality, and linearly interrelated datasets, defect prediction is a subject that is exceedingly complex and challenging. In this research, we addressed the problem of extreme imbalance. When a class is badly unbalanced, the proper modules significantly outweigh the defective ones. This is natural, since defect prediction focuses on flaws [13]. Sadly, class imbalance causes the vast majority of classifiers to overlook flaws that contribute to poor performance. Ensemble classifiers, data-level techniques, and algorithm-level approaches [14] are the three ways to solve the imbalanced class problem. The data-level strategy (Sampling methods) balances class distribution by either decreasing observations from the majority class or boosting samples from the minority class. The algorithm is congruent with cost-sensitive learning (CSL), which asserts that misreading minority samples is more costly than mishandling majority class. Its purpose is to reduce costs [15]. Over the previous two decades, several SDP approaches have been established and applied, including the NN [13], CNN[14], SVM[15], NB[16], RF[17], Dt[18], Logistic regression [19],FL[20], Artificial Immune Systems[21], and Association Rule Mining[21]. The imbalance class problem [25] ubiquitous in software defect datasets hampers the performance of SDP models [24] substantially. In general, there are more non-defective cases than defective examples. While the accuracy of prediction models is typically good when majority class occurrences predominate, the majority of minority class occurrences are mistakenly predicted as majority class samples. Nevertheless, software practitioners are particularly concerned with issues involving minority groups. If forecasting models are skewed toward examples from the majority rather than the minority, they are less useful[22]. To address the issue of imbalance, this research employed a mixed approach.

Due to the vast variance in the dispersion of the classes in the datasets, the problem is regarded as incredibly difficult and tough from a machine learning standpoint. Where the proportion of normal cases greatly outweighs the fraction of defective ones. In such situations, most traditional and fundamental classification algorithms favor accurately classifying the large class and ignoring the smaller class, despite the fact that the fault-prone class is usually significant, as in the SDP problem. As a result, the classifier's performance will suffer. To overcome this challenge, in this research the following are our contributions:

- We performed normalization via the minimum-maximum normalization technique.
- We design a SMOTE to address the issue of imbalanced class problem.
- We proposed an extreme randomized tree, random forest, and extreme gradient boosting for the classification of SDP.
- We evaluate the performance of the models in terms of accuracy, AUC, recall, precision, F1-measure and MCC.
- We compared our proposed models with the existing models, and found that our models performed better.

The remaining sections of the paper are organized as follows. In Section 2, the associated work was described. Section 3 then presents the suggested strategy, and Section 4 presents the results of the experiments and comparative analyses. Section 5 provides the conclusion, which is the last section.

II. LITERATURE REVIEW

A. Related work

The authors [23] proposed a ML framework known as SDP. Pre-processing and re-sampling strategies are employed to prepare the SFP datasets for utilization in ML methodologies. The authors [24] provide the WACIL, an efficient synthetic over-sampling method designed to address the imbalance problem in SDP. The authors [25] have established that unbalanced data has a detrimental impact on the results of fault prediction algorithms. The authors [26] and authors [27] have investigated defect prediction using imbalanced data. They improved the role of defect prediction models by utilizing cost-effective training, sampling approaches, and ensemble approaches. The authors [28] argued that changing samples should be employed to address the issue of imbalanced data. The authors [29] surveyed defect prediction research and discovered widespread use of technical indicators in this domain. The authors [29] has published a review that examined both machine learning and statistical methodologies for fault prediction. As per her survey, the majority of research employed method-specific indicators. Additionally, machine learning techniques outperformed statistical strategies for constructing SDP models in the majority of cases. As a result, our study assesses machine learning strategies for generating defect prediction models. The following two sections summarize studies that used NASA sets of data from 2011 to 2021 and did not employ any cost-effective method for precisely addressing the imbalanced class issue in Table 1, as well as those that did employ a technique for directly addressing the imbalanced data problem in Table 2. summarizes authors that have examined sampling approaches on datasets other than NASA dataset.

Table 1. The studies that used NAA sets

References	Classification methods	Number of NASA dataset
[27]	RF, LR, 15 different variants of NB	8
[30]	DT, PDT, J48, GP, LR, NB, RF, 1Rule, Bagging	7
[31]	Simulated Annealing-PNN, Exhaustive Search-PNN, MLP, CART, C4.5	1
[32]	Cat boost, LGBM), XgBoost, bagged logistic regression, boosted cat boost,	3

Authorized licensed use limited to: The University of Texas Rio Grande Valley. Downloaded on January 10,2025 at 13:59:04 UTC from IEEE Xplore. Restrictions apply.

	boosted XgBoost, and boosted LGBM	

B. SDP Research that Does not Address the Issue of Imbalanced Problem

As mentioned previously, Table 1 outlines studies that employed NASA data but did not address the imbalance problem. While several of the methods [33], [34] in Table 1 employed adequate performance metrics for dealing with class imbalance, they do not solve the problem directly through the use of cost-sensitive machine learning or sampling.

C. SDP Research that Solved the Issue of Imbalanced Class Problem

Table 2 summarizes studies that used specific strategies to solve the problem of class imbalance while utilizing NASA data between 2010 and 2017. Zheng [35] employed cost-sensitive boosting to address the datasets' uneven nature. Seliya et al. [36]suggested an approximately balanced bagging approach for creating defect prediction models when the datasets are uneven. Seliya and Khoshgoftaar [37] analyzed the performance of ML approaches in the presence of imbalanced datasets using a cost-sensitive approach. Rodriguez et al. [38] addressed the uneven structure of training data by utilizing subgroup identification methodologies. The authors [6] correlated balancing strategies and presented a changing variant of Adaboost to deal with data that is out of balance. Rodriguez et al. [39] conducted a study in which they examined cost-effective sample approaches, hybrid strategies, and ensembles for dealing with unbalanced datasets. Their findings indicated that strategies for dealing with unbalanced datasets improved prediction model performance. The researchers [40] presented a two-stage expense learning (TSCS) method to address the unbalanced issue in software fault prediction. The suggested method integrated cost data in two stages, namely the feature selection stage and the classification phase. Their findings established that TSCS approaches outperform single-phase cost-effective learning strategies. The authors [41] classified data into two groups, unbalanced and balanced, to develop accurate defect prediction methods. The authors [42] handled unbalanced data via a cost-effective neural network. The authors [43] presented two strategies for resolving the unbalanced data problem: cost-sensitive DT and cost-effective voting. Additionally, they used SL-SM and SMOTE oversampling approaches to minimize the cost of SDP utilizing decision forests. The paper [44] also dealt with the issue of dealing with unbalanced data through the use of kernel and ensemble learning. Cheng et al.[45] advocated using semi-supervised learning in conjunction with mission dictionary learning to address challenges associated with imbalanced data and deficiently labeled samples.

Table 2. SDP Research that Addresses the Issue of Imbalance Data

References	Classification methods	Number of NASA Datasets
[35]	Cost-sensitive Neural Boosting Classifiers	4
[36]	NB, C4.5, Balanced Bagging Algorithm	7
[38] [6]	CN2, ED, Subgroup discovery	7
[39]	NB, Adaboost, RF, Under-sampling Random, Threshold moving, Random Under-sampling	10
[40]	Two-stage Cost-sensitive NN, Cost-sensitive learning.	7

D. Studies that Employed Sampling Techniques to Deal with Imbalanced Data

Besides the studies discussed in both Sections B and C, there were more significant research papers that solved the topic of balancing data through sampling. The authors [46] conducted an experimental study to determine the influence of sampling strategies on machine-learning-based defect forecasting models. They noticed that while sampling strategies increased the prediction finding of logistic and linear models, they did not necessarily boost the effectiveness of neural network and categorization tree models. Tan et al. [47] tested four sampling strategies for improving online change categorization performance, one of which was SMOTE. Their findings indicated that resampling enhanced classification performance. The authors [48] evaluated the resampling and SMOTE techniques on five NASA data. Their findings bolstered the case for using SMOTE to construct defect prediction models. The authors [49] tested three machine learning strategies for creating defect prediction models using SMOTE on the Eclipse dataset. He discovered that applying SMOTE was advantageous for improving SDP.

III. MATERIAL AND METHODS

The data, learning algorithms, and evaluation standards used in this study are covered in this section. We choose Python programming languages, data sets, and levels of imbalance that varied. The algorithms selected for learning include a variety of additional methods for class imbalance learning. The Figure 1 showed the architecture of the suggested model.

Authorized licensed use limited to: The University of Texas Rio Grande Valley. Downloaded on January 10,2025 at 13:59:04 UTC from IEEE Xplore. Restrictions apply.

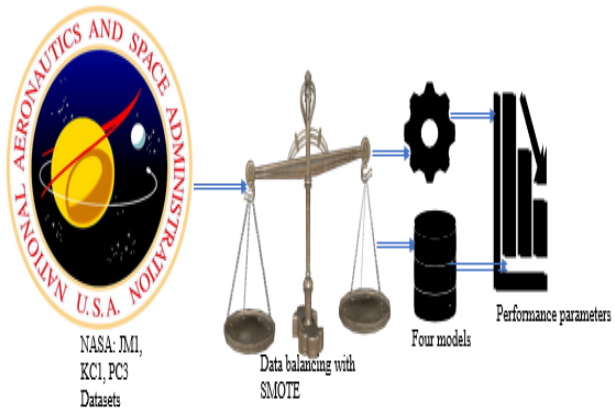


Figure 1. Architecture of the model

A. *Synthetic Minority Oversampling Technique*

In order to address the challenges of imbalance, academics devised the SMOTE strategy, which uses an oversampling method to create synthetic minority class instances. SMOTE chooses neighboring instances of the minority class and utilizes them to create synthetic samples[50].

B. *Extreme Randomized Tree*

This is a subtype of ensemble model in which the classification findings of numerous de-correlated decision trees (DTs) accrued within a "forest" are combined. It is similar to the random forest in operation but differs mostly in the way the DTs within the forest are formed [54]. That DT is mostly based on data collected during training in Extra Trees Forest. Thus, at each testing node, each tree is randomly assigned k-features from the feature set, from which each DT must select the most significant characteristics to separate the statistics according to particular numerical parameters.

C. *Random Forest*

Random Forest (RF) is a technique for data mining that is often employed. As a classifier integration strategy, RF has the advantages of being rapid to classify, easy to train, and suitable for feature selection based on variable relevance. Breiman introduced random forests in 2001 as a more recent evolution of decision tree learning [52], [53]. This information is used as a validation set in order to acquire a credible estimation of the classification error. Random forest in [54] is a promising candidate for predicting software quality, particularly for large-scale systems, because it is consistently more accurate than existing classification algorithms.

D. *Extreme Gradient Boosting*

Extreme gradient boosting is a GB ensemble model that utilizes the partial derivatives of the loss function to more correctly and efficiently select the best base classifier[51]. Whereas GB fits a new base classifier using gradients, XGB uses second-order gradients[52].

IV. EXPERIMENTAL ANALYSIS

We explained the experimental dataset, and performance metrics adopted in this section.

A. *Dataset*

All three SDP data sets given in Table 3 are from real-world projects and are freely available through the public PROMISE repository [53] to ensure the reproducibility and verifiability of our predictive models and to facilitate comparisons to other publications. These data were compiled from real NASA software projects written in C/C++ for managing ground data storage (KC1 data), and scientific information processing (PC3 data) and (JM1 data). Table 3 has a thorough overview of the datasets.

Table 3. Overview of the Experimental Datasets

Project	Description	Language	Defects	Non-Defects	%Defects	%Non-Defects
JM1	Instrument on a NASA spacecraft	C	2106	8779	19.35	80.65
KC1	Storage management system for collecting and transmitting ground data	C++	326	1783	15.45	84.54
PC3	Metadata for earth-orbiting satellite flight software	C	160	1403	10.23	89.7

B. *Performance Metrics for Imbalance Datasets*

Metrics play a critical role in resolving class imbalance issues. Although accuracy is an essential metric for evaluating classification outcomes, it is inadequate for analyzing imbalanced datasets, as it prioritizes the majority class and fails to examine the classifier's performance on the minority class[54]. When dealing with unbalanced situations, performance evaluation measures must take class distribution into account [55]. Metrics including the count of correct predictions, true positives, false positives, and false negatives can be employed to elucidate the classification performance for each class separately. Due to the essential requirement for high-quality performance in imbalanced scenarios, these metrics are included into broader assessments like AUC [56] and F-measure [57].

V. RESULTS AND DISCUSSION

All algorithms utilized in our research are trained and evaluated by a ten-fold cross-validation procedure. The ML algorithms are evaluated based on the performance measures specified in the previous section. SMOTE generated synthetic instances of the minority class to rectify the dataset's imbalance. For each occurrence involving a minority group. We applied the three proposed hybrid models SMOTE+ET, SMOTE+RF, and SMOTE+Xgboost on the JM1, KC1, and PC3 datasets. The classification accuracy, AUC, recall, precision, F1-score, and MCC have been utilized to evaluate the suggested method's performance. The

Authorized licensed use limited to: The University of Texas Rio Grande Valley. Downloaded on January 10,2025 at 13:59:04 UTC from IEEE Xplore. Restrictions apply.

experimental findings of the hybrid SMOTE were given in Tables 4, 5, and Table 6 for the JM1, KC1, and PC3 datasets.

Table 4. Proposed hybrid model's evaluation result on the JM1 dataset

Algorithms	Accuracy	AUC	Recall	Precision	F-measure	MCC
SMOTE+Extra tree	80.84	72.28	18.45	52.26	27.24	22.49
SMOTE+RF	81.39	73.89	18.72	56.47	28.03	24.54
SMOTE+Xgboost	80.65	72.46	19.86	51.01	28.53	22.77

Based on the experimental results shown in Table 4, the proposed SMOTE+RF obtained the best results when compared to the SMOTE+Extra tree and SMOTE+Xgboost in terms of the AUC on the JM1 dataset. The SMOTE+xgboost model outperformed the SMOTE+Extra tree and SMOTE+RF in terms of the F-measure on the same JM1 data. Hence, we, therefore, plot the ROC of the SMOTE+RF that gave the best-proposed model on the JM1 dataset in Figure 2.

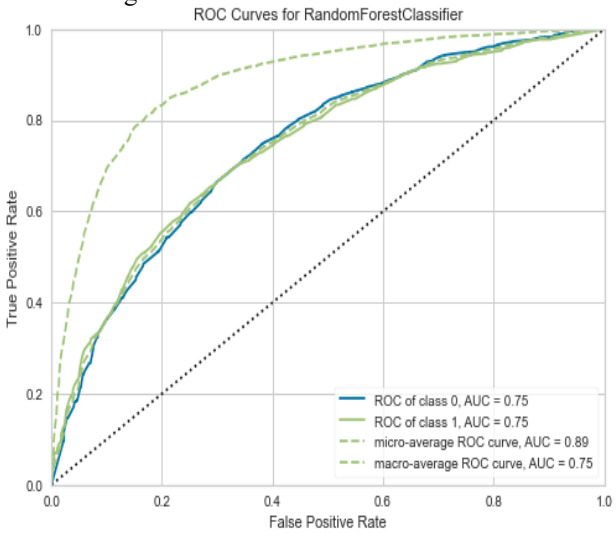


Fig.2. ROC of the proposed SMOTE+RF model on the Jm1 data

As shown in Figure 2, our proposed SMOTE-RF gave an AUC of 0.75 for the non-defect (class 0) and 0.75 for the defect class. The micro-average ROC curve attained for the AUC is 0.89, and the macro-average ROC curve obtained for the AUC is 0.75.

Table 5. Proposed hybrid model's evaluation result on the KC1 dataset

Algorithms	Accuracy	AUC	Recall	Precision	F-measure	MCC
SMOTE+Extra tree	86.04	85.47	32.87	58.70	41.78	36.67
SMOTE+RF	86.11	84.67	32.41	61.04	41.81	37.28
SMOTE+Xgboost	85.84	81.96	38.10	56.70	45.05	38.60

Based on the experimental results shown in Table 5, the proposed SMOTE+Extra Tree attained the greatest results when compared to the SMOTE+RF and SMOTE+Xgboost in terms of the AUC on the KC1 dataset. The SMOTE+RF model outperformed the SMOTE+Extra tree and SMOTE+Xgboost in terms of the F-measure on the same KC1 data. Hence, we, therefore, plot the ROC of the SMOTE+extra tree that gave the best-proposed model on the KC1 dataset in Figure 3.

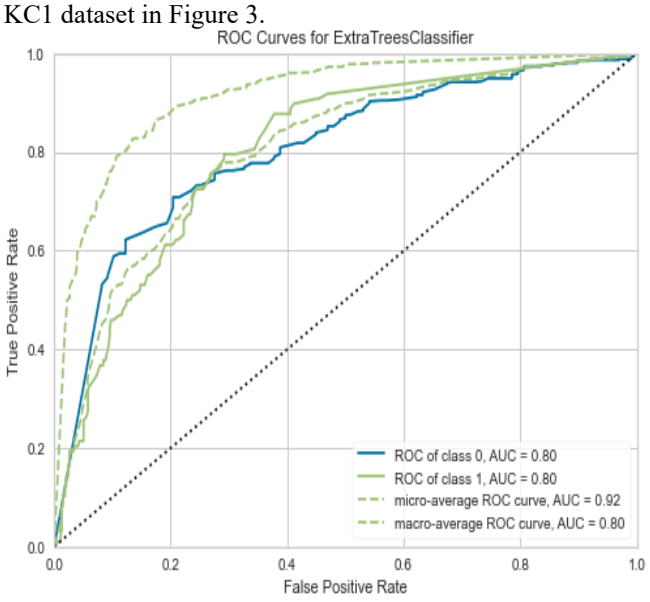


Fig.3.ROC of the proposed SMOTE+Extra tree model on the Jm1 data

As shown in Figure 3, our proposed SMOTE-Extra tree gave an AUC of 0.80 for the non-defect (class 0) and 0.80 for the defect class. The micro-average ROC curve achieved an AUC of 0.92, whereas the macro-average receiver operating curve yielded an AUC of 0.80.

Table 6. Proposed hybrid model's evaluation result on the PC3 dataset

Algorithms	Accuracy	AUC	Recall	Precision	F1	MCC
SMOTE+Extra tree	93.43	77.68	22.67	60.00	31.90	33.41
SMOTE+RF	93.69	82.70	23.00	61.67	32.98	34.74
SMOTE+Xgboost	93.43	82.64	26.33	58.33	34.13	34.96

As per experimental results summarized in Table 6, the proposed SMOTE+RF achieved the highest results when compared to the SMOTE+Extra tree and SMOTE+Xgboost in terms of the AUC and precision on the PC3 dataset. The SMOTE+Xgboost model outperformed the SMOTE+Extra tree in terms of the AUC on the same PC3 data. Hence, we, therefore, plot the ROC of the SMOTE+Xgboost model that gave the best-proposed model on the PC3 dataset in Figure 4.

Authorized licensed use limited to: The University of Texas Rio Grande Valley. Downloaded on January 10,2025 at 13:59:04 UTC from IEEE Xplore. Restrictions apply.

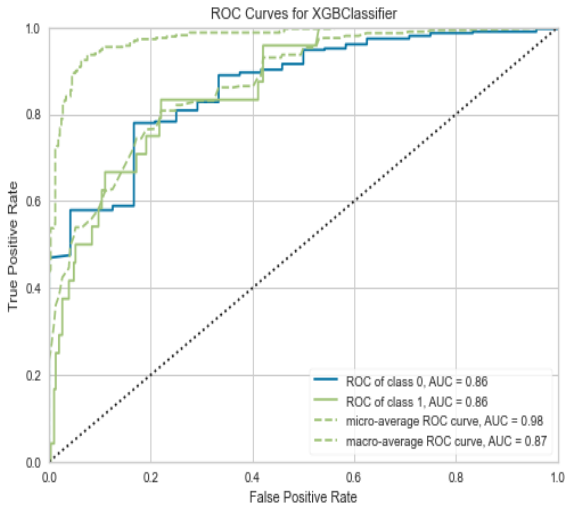


Fig.4.ROC of the proposed SMOTE+Xgboost model on the PC3 data

As shown in Figure 4, our proposed SMOTE+Xgboost gave an AUC of 0.86 for the non-defect (class 0) and 0.86 for the defect class. The micro-average ROC curve attained for the AUC is 0.98, and the macro-average ROC curve obtained for the AUC is 0.87.

A. Comparison with the Existing Techniques

To validate the proposed HSMOTE's effect, we compare it to the recent state-of-the-art class unbalanced approaches, which will be discussed in detail in Table 7. As observed in Table 7, our proposed hybrid models outperformed the existing study in terms of AUC, accuracy, and F-measure. The existing study [10] did not consider the AUC and F-measure. These two metrics are known to be the most accepted performance measures in imbalanced data problems.

Table 7. Comparison with the existing model utilizing the JM1 dataset

Authors	Approaches	Dataset	Accuracy	AUC	F-measure
[10]	SMOTE+Adaboost	JM1	77.87	x	X
Proposed	SMOTE+ET	JM1	80.84	72.28	27.24
Proposed	SMOTE+RF	JM1	81.39	73.89	28.03
Proposed	SMOTE+Xgboost	JM1	80.65	72.46	28.53

We conducted a comparative analysis of the performance of our proposed models against the existing study that employed the KC1 dataset for SDP, as presented in Table 8. Our proposed HSMOTE models outperformed the existing [10] model in terms of AUC, accuracy, and F-measure.

Table 8. Comparison with the existing model utilizing the KC1 dataset

Authors	Approaches	Dataset	Accuracy	AUC	F-measure
[10]	SMOTE+Adaboost	KC1	83.29	x	x
Proposed	SMOTE+ET	KC1	86.04	85.47	41.78
Proposed	SMOTE+RF	KC1	86.11	84.67	41.81
Proposed	SMOTE+Xgboost	KC1	85.84	81.96	45.05

The performance of our proposed models is given in Table 9, the existing models yielded an accuracy of 87.72%. The HSMOTE model we proposed achieved an accuracy of 93.43%, illustrating its superiority.The existing model [10]did not paid attention to the AUC, and F-measure, which are the main strength of our proposed HSMOTE models.

Table 9. Comparison with the existing model utilizing the PC3 dataset

Authors	Approaches	Dataset	Accuracy	AUC	F-measure
[10]	SMOTE+Adaboost	PC3	87.72	x	X
Proposed	SMOTE+ET	PC3	93.43	77.68	31.90
Proposed	SMOTE+RF	PC3	93.69	82.70	32.98
Proposed	SMOTE+Xgboost	PC3	93.43	82.64	34.13

VI. CONCLUSION AND FUTURE WORK

This research proposes a new hybrid technique for predicting software defects using ML models. Many strategies are being considered to further enhance the performance of software defect detection in this intriguing field of research. In this research, we solved the issue of imbalance in SDP by inventing a new hybrid SMOTE technique. To categorize our suggested HSMOTE model, we employed the Extra tree, RF, and Xgboost. On the JM1 dataset, SMOTE+Extra tree achieved an AUC of 72.28%, SMOTE+RF achieved an AUC of 73.89%, and SMOTE+Xgboost achieved an AUC of 72.46%. On the KC1 dataset, SMOTE+Extra tree achieved an AUC of 85.47%, SMOTE+RF achieved an AUC of 84.67%, and SMOTE+Xgboost revealed an AUC of 81%. Likewise, it performs better in terms of classification accuracy. In a future study, we intend to leverage Adaptive synthetic technique for resolving the imbalance issue in SDP and utilize the deep learning for classification of the NASA SDP.

REFERENCES

[1] Z. Xu *et al.*, "Software defect prediction based on kernel PCA and weighted extreme learning machine," *Inf. Softw. Technol.*, vol. 106, no. October, pp. 182–200, 2019, doi: 10.1016/j.infsof.2018.10.004.

[2] X. Yang, K. Tang, and X. Yao, "A learning-to-rank approach to software defect prediction," *IEEE Trans. Reliab.*, vol. 64, no. 1, pp. 234–246, 2015, doi: 10.1109/TR.2014.2370891.

[3] N. Kalaivani and R. Beena, "Overview of software defect prediction using machine learning algorithms," *Int. J. Pure Appl. Math.*, vol. 118, no. 20, pp. 3863–3873, 2018.

[4] D. Jianli and N. Shi, "Research on Fuzzy Extension Synthesis Metrics Algorithm for Software Quality," *J. Softw.*, vol. 6, no. 11, pp. 2099–2328, 2015.

Authorized licensed use limited to: The University of Texas Rio Grande Valley. Downloaded on January 10,2025 at 13:59:04 UTC from IEEE Xplore. Restrictions apply.

[5] Y. K. Saheed and M. A. Hambali, "Customer Churn Prediction in Telecom Sector with Machine Learning and Information Gain Filter Feature Selection Algorithms," in *2021 International Conference on Data Analytics for Business and Industry (ICDABI)*, 2021, pp. 208–213, doi: 10.1109/ICDABI53623.2021.9655792.

[6] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Trans. Reliab.*, vol. 62, no. 2, pp. 434–443, 2013, doi: 10.1109/TR.2013.2259203.

[7] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models," *IEEE Trans. Softw. Eng.*, vol. 5589, no. c, pp. 1–22, 2018, doi: 10.1109/TSE.2018.2876537.

[8] F. Provost, "Machine learning from imbalanced data sets 101," *Proc. AAAI'2000 Work. ...*, p. 3, 2000, [Online]. Available: https://www.aaai.org/Papers/Workshops/2000/WS-00-05/WS00-05-001.pdf%5Cnpapers://1c40c143-2a6e-4e94-8c17-c5bc9ae73d7e/Paper/p11435.

[9] 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. Rev.*, vol. 42, no. 6, pp. 1806–1817, 2012, doi: 10.1109/TSMCC.2012.2226152.

[10] H. Alsawalqah, H. Faris, I. A. B, and L. Alnemer, "Hybrid SMOTE-Ensemble Approach," vol. 1, doi: 10.1007/978-3-319-57141-6.

[11] 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, 2012, doi: 10.1109/TSE.2011.103.

[12] N. Fenton and J. Bieman, *Software metrics: a rigorous and practical approach*. CRC Press Taylor & Francis Group, 2014.

[13] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," *Cluster Comput.*, vol. 22, pp. 77–88, 2019, doi: 10.1007/s10586-018-1730-1.

[14] T. M. Usman, Y. K. Saheed, A. Nsang, A. Ajibesin, and S. Rakshit, "A systematic literature review of machine learning based risk prediction models for diabetic retinopathy progression," *Artif. Intell. Med.*, vol. 143, 2023, doi: 10.1016/j.artmed.2023.102617.

[15] X. Cai *et al.*, "An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search," *Concurr. Comput. Pract. Exp.*, vol. 32, no. 5, pp. 1–14, 2020, doi: 10.1002/cpe.5478.

[16] H. Ji, S. Huang, Y. Wu, Z. Hui, and C. Zheng, "A new weighted naive Bayes method based on information diffusion for software defect prediction," *Softw. Qual. J.*, vol. 27, no. 3, pp. 923–968, 2019, doi: 10.1007/s11219-018-9436-4.

[17] A. Iqbal *et al.*, "Performance analysis of machine learning techniques on software defect prediction using NASA datasets," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 5, pp. 300–308, 2019, doi: 10.14569/ijacsa.2019.0100538.

[18] A. Iqbal, S. Aftab, I. Ullah, M. Salman Bashir, and M. Anwaar Saeed, "A Feature Selection based Ensemble Classification Framework for Software Defect Prediction," *Int. J. Mod. Educ. Comput. Sci.*, vol. 11, no. 9, pp. 54–64, 2019, doi: 10.5815/ijmecs.2019.09.06.

[19] L. Li, S. Lessmann, and B. Baesens, "Evaluating Software Defect Prediction Performance: An Updated Benchmarking Study," *SSRN Electron. J.*, 2019, doi: 10.2139/ssrn.3312070.

[20] R. Özakıncı and A. Tarhan, "Early software defect prediction: A systematic map and review," *J. Syst. Softw.*, vol. 144, pp. 216–239, 2018, doi: 10.1016/j.jss.2018.06.025.

[21] X. Peng, "Research on Software Defect Prediction and Analysis Based on Machine Learning," *J. Phys. Conf. Ser.*, vol. 2173, no. 1, 2022, doi: 10.1088/1742-6596/2173/1/012043.

[22] Y. K. Saheed, T. T. Salau-Ibrahim, M. Abdulsalam, I. A. Adeniji, and B. F. Balogun, "Modified bi-directional long short-term memory and hyperparameter tuning of supervised machine learning models for cardiovascular heart disease prediction in mobile cloud environment," *Biomed. Signal Process. Control*, vol. 94, no. February, p. 106319, 2024, doi: 10.1016/j.bspc.2024.106319.

[23] M. Mafarja *et al.*, "Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning," *Appl. Intell.*, vol. 53, no. 15, pp. 18715–18757, 2023, doi: 10.1007/s10489-022-04427-x.

[24] M. B. Pravali Manchala, "Diversity based imbalance learning approach for software fault prediction using machine learning models," *Appl. Soft Comput.*, vol. 124, 2022, doi: 10.1016/j.asoc.2022.109069.

[25] Z. Mahmood, D. Bowes, and P. C. R. Lane, "What is the Impact of Imbalance on Software Defect Prediction Performance ?," 2015.

[26] G. M. Weiss, K. Mccarthy, and B. Zabar, "Cost-Sensitive Learning vs . Sampling : Which is Best for Handling Unbalanced Classes with Unequal Error Costs ?"

[27] M. Galar, A. Fern, E. Barrenechea, and H. Bustince, "Hybrid-Based Approaches," vol. 42, no. 4, pp. 463–484, 2012.

[28] M. Lin, K. Tang, S. Member, and X. Yao, "Dynamic Sampling Approach to Training Neural Networks for Multiclass Imbalance Classification," vol. 24, no. 4, pp. 647–660, 2013.

[29] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, 2009, doi: 10.1016/j.eswa.2008.10.027.

[30] Y. C. Liu and I. C. Soceity, "Evolutionary Optimization of Software Quality Modeling with Multiple Repositories," vol. 36, no. 6, pp. 852–864, 2010.

[31] P. C. Pendharkar, "Engineering Applications of Artificial Intelligence Exhaustive and heuristic search approaches for learning a software defect prediction model," *Eng. Appl. Artif. Intell.*, vol. 23, no. 1, pp. 34–40, 2010, doi: 10.1016/j.engappai.2009.10.001.

[32] Y. K. Saheed, O. Longe, U. A. Baba, S. Rakshit, and N. R. Vajjhala, "An Ensemble Learning Approach for Software Defect Prediction in Developing Quality Software Product," in *Advances in Computing and Data Sciences*, M. Singh, V. Tyagi, P. K. Gupta, J. Flusser, T. Ören, and V. R. Sonawane, Eds. Springer, Cham, 2021.

[33] K. Dejaeger, T. Verbraken, and B. Baesens, "Prediction Models Using Bayesian Network Classifiers," vol. 39, no. 2, pp. 237–257, 2013.

[34] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Inf. Sci. (Ny).*, vol. 264, pp. 260–278, 2014, doi: 10.1016/j.ins.2013.12.031.

[35] J. Zheng, "Cost-sensitive boosting neural networks for software defect prediction," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4537–4543, 2010, doi: 10.1016/j.eswa.2009.12.056.

[36] N. Seliya and J. Van Hulse, "Predicting Faults in High Assurance Software," 2010, doi: 10.1109/HASE.2010.29.

[37] N. Seliya and T. M. Khoshgoftaar, "The use of decision trees for cost-sensitive classification: An empirical study in software quality prediction," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 1, no. 5, pp. 448–459, 2011, doi: 10.1002/widm.38.

[38] D. Rodríguez, R. Ruiz, J. C. Riquelme, and J. S. Aguilar-Ruiz, "Searching for rules to detect defective modules: A subgroup

Authorized licensed use limited to: The University of Texas Rio Grande Valley. Downloaded on January 10,2025 at 13:59:04 UTC from IEEE Xplore. Restrictions apply.

discovery approach," *Inf. Sci. (Ny).*, vol. 191, pp. 14–30, 2012, doi: 10.1016/j.ins.2011.01.039.

[39] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary Comparison of Techniques for Dealing with Imbalance in Software Defect Prediction Categories and Subject Descriptors."

[40] M. Liu, L. Miao, and D. Zhang, "Two-stage cost-sensitive learning for software defect prediction," *IEEE Trans. Reliab.*, vol. 63, no. 2, pp. 676–686, 2014, doi: 10.1109/TR.2014.2316951.

[41] Y. Abdi, S. Parsa, and Y. Seyfari, "A hybrid one-class rule learning approach based on swarm intelligence for software fault prediction," *Innov. Syst. Softw. Eng.*, vol. 11, no. 4, pp. 289–301, 2015, doi: 10.1007/s11334-015-0258-2.

[42] Ö. F. Arar and K. Ayan, "Software defect prediction using cost-sensitive neural network," *Appl. Soft Comput. J.*, vol. 33, pp. 263–277, 2015, doi: 10.1016/j.asoc.2015.04.045.

[43] M. J. Siers and M. Z. Islam, "Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem," *Inf. Syst.*, vol. 51, pp. 62–71, 2015, doi: 10.1016/j.is.2015.02.006.

[44] T. Wang, Z. Zhang, X. Jing, and L. Zhang, "Multiple kernel ensemble learning for software defect prediction," *Autom. Softw. Eng.*, vol. 23, no. 4, pp. 569–590, 2016, doi: 10.1007/s10515-015-0179-1.

[45] M. Cheng, G. Wu, M. Yuan, and H. Wan, "Semi-supervised software defect prediction using task-driven dictionary learning," *Chinese J. Electron.*, vol. 25, no. 6, pp. 1089–1096, 2016.

[46] B. J. Williams and J. C. Carver, "Characterizing software architecture changes: An initial study," *Proc. - 1st Int. Symp. Empir. Softw. Eng. Meas. ESEM 2007*, pp. 410–419, 2007, doi: 10.1109/ESEM.2007.28.

[47] M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online Defect Prediction for Imbalanced Data," *Proc. - Int. Conf. Softw. Eng.*, vol. 2, pp. 99–108, 2015, doi: 10.1109/ICSE.2015.139.

[48] J. C. Riquelme, R. Ruiz, and D. Rodr, "Finding Defective Modules from Highly," *Engineering*, vol. 2, no. 1, pp. 67–74, 2008.

[49] R. Shatnawi, "Improving software fault-prediction for imbalanced data," *2012 Int. Conf. Innov. Inf. Technol. IIT 2012*, pp. 54–59, 2012, doi: 10.1109/INNOVATIONS.2012.6207774.

[50] Y. K. Saheed, S. Misra, and S. Chockalingam, "Autoencoder via DCNN and LSTM Models for Intrusion Detection in Industrial Control Systems of Critical Infrastructures," *2023 IEEE/ACM 4th Int. Work. Eng. Cybersecurity Crit. Syst. (EnCyCriS), Melbourne, Aust.*, pp. 9–16, 2023, doi: 10.1109/EnCyCriS59249.2023.00006.

[51] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," *PROMISE 2020 - Proc. 16th ACM Int. Conf. Predict. Model. Data Anal. Softw. Eng. Co-located with ESEC/FSE 2020*, pp. 1–10, 2020, doi: 10.1145/3416508.3417114.

[52] C. Tianqi and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *KDD*, vol. 42, no. 8, p. 665, 2016, doi: 10.1145/2939672.2939785.

[53] G. Boetticher, "The PROMISE repository of empirical software engineering data."

[54] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou, "On the Class Imbalance Problem *," pp. 192–201, 2008, doi: 10.1109/ICNC.2008.871.

[55] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data : Empirical results and current trends on using data intrinsic characteristics," 2013, doi: 10.1016/j.ins.2013.07.007.

[56] J. Huang and C. X. Ling, "Using AUC and Accuracy in Evaluating Learning Algorithms," vol. 17, no. 3, pp. 299–310, 2005.

[57] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*. New York: ACM press., 1999.

Authorized licensed use limited to: The University of Texas Rio Grande Valley. Downloaded on January 10,2025 at 13:59:04 UTC from IEEE Xplore. Restrictions apply.