Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 10 No. 4 (2026) 1125 - 1132 e-ISSN: 2580-0760

Enhancing Software Defect Prediction Performance using NR-Clustering SMOTE to Address Class Imbalance

Hairani Hairani[1], Muhamad Masjun Efendi[2], Gede Yogi Pratama[3], Rahayun Amrullah Husaini[4], M. Khaerul Ihsan[5]
[1,3,4,5]Department of Computer Science, Faculty of Engineering, Universitas Bumigora, Mataram, Indonesia
[2]Informatics Management, Universitas Teknologi Mataram, Mataram, Indonesia

[*]*hairani@universitasbumigora.ac.id*

Abstract

Software defect detection is important to prevent system failures and increased maintenance costs. However, the complexity of modern software makes manual testing inefficient, so machine learning approaches are used. The main challenge of this approach is data imbalance, where defective cases are far fewer, causing the model to overlook the minority class and reducing detection capability, even though accuracy appears high. This study aims to address class imbalance in software defect detection by applying the NR-Clustering SMOTE method to improve machine learning performance—the classification methods using Random Forest. NR-Clustering SMOTE not only oversamples the minority class but also incorporates a noise-reduction mechanism to remove minority data that may degrade classification performance. The results show that NR-Clustering SMOTE improves the performance of Random Forest compared with the original data, SMOTE, and NR-Modified SMOTE across all evaluation metrics, namely accuracy, recall, and F1-score. These findings indicate that integrating noise reduction and SMOTE-based data balancing using Manhattan distance within each cluster produces a more representative data distribution, thereby improving the model's ability to classify software defect cases more accurately. Therefore, this study confirms that NR-Clustering SMOTE effectively improves Random Forest performance for software defect detection compared with existing approaches.

Keywords: noise reduction; NR-Clustering SMOTE; NR-Modified SMOTE; software defect

How to Cite: H. Hairani, Muhamad Masjun Efendi, Gede Yogi Pratama, Rahayun Amrullah Husaini, and M.Khaerul Ihsan, "Enhancing Software Defect Prediction Performance using NR-Clustering SMOTE to Address Class Imbalance", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 10, no. 4, pp. 1125–1132, Aug. 2026, doi: 10.29207/resti.v10i7558.
Permalink/DOI: https://doi.org/10.29207/resti.v10i7558

Received: February 6, 2026
Accepted: June 21, 2026
Available Online: August 21, 2026

This is an open-access article under the CC BY 4.0 License
Published by Ikatan Ahli Informatika Indonesia

1. Introduction

Software defect detection is crucial in the software development cycle because defects that are not detected early can cause system failures [1], increase maintenance costs [2], and reduce software quality and reliability [3]. As the complexity of modern software systems increases, manual testing becomes less efficient and difficult to perform optimally [4]. Therefore, an automated approach is needed to help developers identify potential software defects quickly and accurately [5].

Machine learning approaches have been widely applied in software defect detection due to their ability to learn historical patterns from software development data [6], [7]. However, machine learning implementation often faces the problem of data imbalance, where the number of defective modules is far smaller than that of non-defective modules [8]. This condition causes models to focus on the majority class and ignore the minority class, resulting in low defect detection capability, even though accuracy appears high [9]. The inability of machine learning models to detect software defects effectively can directly increase the risk of system failure and waste resources during development [10]. Therefore, handling data imbalance is a key priority for improving the effectiveness of machine learning–based software defect detection [11].

Previous studies in software defect detection have used various classification methods, ranging from deep learning approaches such as LSTM and 1D-CNN to conventional methods such as Logistic Regression, k-NN, Naive Bayes, and Random Forest, and to ensemble and hybrid methods such as XGBoost, AdaBoost, and Stacked Ensemble. Some studies also applied data preprocessing techniques, including

feature selection, dimensionality reduction, and under-sampling, to improve classification performance. However, most of these studies did not explicitly address class imbalance. Although some research has adopted data-balancing techniques such as Neighborhood-based Under-Sampling and SMOTE, these approaches still have limitations. These limitations include excessive removal of the majority class, which may eliminate important information, and the absence of a noise-reduction mechanism for minority-class data generated by oversampling. This situation indicates a research gap in more comprehensive class imbalance handling, particularly in considering the quality of minority-class data, which underpins the method proposed in this study.

Furthermore, the study [12] developed the Modified SMOTE method to handle class imbalance in a healthcare dataset by first removing minority data considered noise using the K-Nearest Neighbor (KNN) method. Then, data balancing was performed using SMOTE oversampling with the Manhattan distance. Study [13] proposed the NR-Clustering SMOTE method to address data imbalance in a healthcare dataset by removing minority data identified as noise using KNN. After that, the data were clustered using the K-Means method, and Manhattan-distance–based SMOTE oversampling was applied within each cluster.

Several previous studies have examined software defect detection using machine learning approaches and addressed data imbalance using SMOTE. However, these studies generally focused only on balancing the data using SMOTE, without addressing the noise in the minority class introduced by oversampling. Such noise can degrade machine learning performance, creating a research gap that this study addresses (see Table 1). In contrast to previous studies that introduced NR-Clustering SMOTE to address data imbalance in the healthcare sector, this study validates its robustness for software defect prediction. This method not only oversamples the minority class but also incorporates a noise-reduction mechanism to remove minority data that may degrade classification performance. In addition, the data balancing process is performed in a cluster-based manner to produce more representative synthetic data and reduce class overlap. This study aims to overcome class imbalance in software defect detection by applying NR-Clustering SMOTE to improve machine learning performance. With NR-Clustering SMOTE, the performance of machine learning methods, particularly in detecting defective software classes, is expected to improve significantly and consistently. This study uses machine learning methods, including Random Forest with default parameters, for software defect classification.

Table 1. Comparison of the Proposed Method's Performance with Several Previous Studies

Researchers	Classification Method	Preprocessing Technique	Limitation
Albattah & Alzahran [6]	LSTM	-	
Zain et al. [14]	1D-CNN1	-	
Shahzad et al. [15]	XGBoost	-	
Sourav et al. [16]	Logistic Regression	-	
Pradhan & Muduli [17]	IMPSO-ELM	PCA- LDA	
Ramadhani et al. [18]	k-nearest neighbor (kNN) method	-	
Alkhasawneh [19]	RBF	-	Does not resolve the issue of class imbalance
Priyadarshini [20]	Priority-Based Fuzzy SVM	-	
Ali et al. [21]	Voting ensemble (IECGA)	feature selection	
Wintana et al. [22]	C45 with AdaBoost	-	
Arasteh et al. [23]	KNN	BCOOA	
Babatunde et al. [24]	NB with Dagging	-	
Patel & Gupta [25]	XGBoost	-	
Iqbal et al. [26]	Random Forest	-	
Mehmood et al. [27]	Logistic Regression	Feature Selection	
Meng et al. [28]	Tri_SSDPM	-	
Mohil et al. [29]	Stacked Ensemble	-	
Goyal [30]	KNN	Neighborhood-based Under-Sampling (N-US)	Only deleting the majority class data results in the loss of a lot of important information
Febrian et al. [31]	KNN	PSO SMOTE CS CFS CMFS	Balancing data with SMOTE, but not resolving the resulting minority class data noise
Mustaqeem & Siddiqui [32]	SVM	SMOTE	

2. Methods

Figure 1 shows the research workflow, which starts with data collection, followed by data preprocessing, 10-fold cross-validation data splitting, and the construction of classification models using Random Forest, and ends with a performance evaluation stage based on accuracy, recall, and F1-score.

Software defect data were collected using the CM1 and PC1 datasets from Kaggle. The CM1 dataset consists of 498 instances and 22 attributes with an imbalance ratio of 10.16, while PC1 has 1109 instances and 22 attributes with an imbalance ratio of 13.40. Class imbalance is evident in CM1, with 49 minority and 449 majority data instances, and in PC1, with 77 minority and 1032 majority data points, which can affect classification performance. During data

preprocessing, class imbalance was addressed using two approaches: NR-Modified SMOTE and NR-Clustering SMOTE.

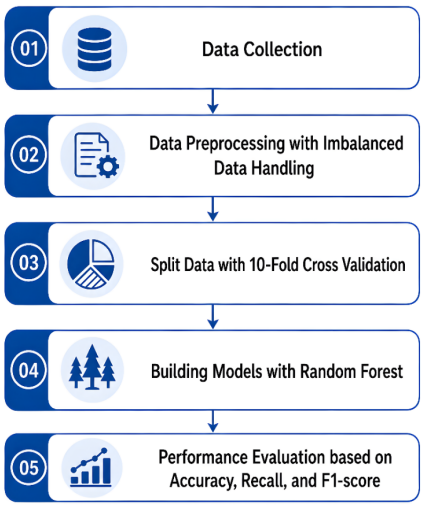


Figure 1. Research Flow

The NR-Modified SMOTE approach was carried out in two steps: removing noisy minority data located near the majority class using the k-NN method, and applying SMOTE oversampling using the Manhattan distance metric [12]. Meanwhile, NR-Clustering SMOTE involved three main steps: removing noisy minority data using k-NN, clustering the data with the K-Means algorithm to form decision boundaries, and applying Manhattan distance–based SMOTE oversampling within each cluster [13].

The processed dataset was evaluated using 10-fold cross-validation, with 9 subsets for training and 1 for testing, performed in an alternating manner to ensure reliable evaluation and minimize bias [33]. During model development, the Random Forest algorithm is used to build a classification model.

Random Forest works by taking bootstrap samples from a dataset through a sampling-with-replacement process to form several training datasets. Each dataset is used to build a decision tree, with features selected randomly at each split. Each tree then generates a prediction for the given data. These predictions are then aggregated using majority voting, where the class receiving the most votes becomes the Random Forest's final prediction. In this study, the Random Forest model was trained on data preprocessed using the NR-Clustering SMOTE method.

The final stage was to evaluate the classification models using several metrics, namely accuracy, recall, and F1-score. These metrics were used to provide a thorough assessment of model performance, particularly in detecting the minority class with greater accuracy. Accuracy measures the proportion of correct predictions across all data, but it is less representative for imbalanced datasets because it tends to be influenced by the majority class [34]. Recall measures the model's ability to correctly detect the minority class, making it an important metric for class imbalance problems [35]. The F1-score is the harmonic mean of precision and recall, balancing prediction accuracy and completeness, and is therefore effective for evaluating imbalanced datasets [36], [37]. Accuracy, Recall, and F1-score are calculated based on Equations 1 to 3.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

3. Results and Discussions

Software defect data were collected using the CM1 and PC1 datasets from Kaggle. The CM1 dataset consists of 498 instances and 22 attributes with an imbalance ratio of 10.16, while PC1 has 1109 instances and 22 attributes with an imbalance ratio of 13.40. Class imbalance is evident in CM1, with 49 minority and 449 majority data instances, and in PC1, with 77 minority and 1032 majority data points, which can affect classification performance [38]. Therefore, data imbalance in software defect detection needs to be addressed through an oversampling approach. Table 2 presents the class distribution in the dataset before and after oversampling. In the original data, the class distribution is highly imbalanced: 449 instances in the False class and 49 in the True class in the CM1 Dataset. After applying the SMOTE method, the data distribution becomes balanced, with each class containing 449 instances of the CM1 Dataset. Meanwhile, the PC1 dataset has a highly unbalanced class distribution, with 1032 instances in the False class and 77 in the True class. After applying the SMOTE, NR-Modified SMOTE, and NR-Clustering SMOTE methods, the data distribution becomes balanced, with each class containing 1032 instances.

Table 2. Number of Classes in the Dataset After Oversampling

Data Sampling	CM1		PC1	
	False	True	False	True
Data Original	449	49	1032	77
SMOTE	449	449		
NR-Modified SMOTE			1032	1032
NR-Clustering SMOTE	449	447		

Furthermore, the application of NR-Modified SMOTE and NR-Clustering SMOTE produces a more balanced class distribution than the original data, indicating the effectiveness of the proposed approach in addressing class imbalance in the dataset.

Based on the confusion matrix results for software defect detection shown in Table 3, using the original imbalanced data yields suboptimal classification performance, especially in detecting the minority class. Random Forest tends to be biased toward the majority class, as indicated by very low True Positive values. The application of SMOTE improves the models' ability to recognize the minority class. Increases in

True Positive values show this improvement and decreases in False Negative values across all classification methods, especially in Random Forest.

Table 3. Performance Comparison with Previous Research on Software Defects

Data Sampling	Datasets	False	True
Data Original	CM1	447	2
		48	1
	PC1	1016	16
		60	17
SMOTE	CM1	403	46
		13	436
	PCI	967	65
		23	1009
NR-Modified SMOTE	CM1	433	16
		1	448
	PC1	1012	20
		5	1027
NR-Clustering SMOTE	CM1	434	15
		0	447
	PC1	1016	16
		4	1028

The NR-Modified SMOTE approach produces more balanced performance than standard SMOTE. The noise reduction (NR) stage using the k-NN method effectively reduces potentially disruptive minority-class instances, thereby improving the quality of the oversampled data. This condition leads to an increase in True Positives and a decrease in classification errors across all methods, with Random Forest showing the most stable and accurate performance in detecting software defects. Meanwhile, the NR-Clustering SMOTE approach provides the most optimal and consistent results across all testing scenarios. The combination of noise filtering, data clustering using the K-Means algorithm, and Manhattan distance–based SMOTE creates clearer decision boundaries and reduces class overlap. This approach achieves excellent minority-class detection, particularly with Random Forest, which achieves a zero false-negative rate. Therefore, it can be concluded that NR-Clustering SMOTE is the most effective method for addressing data imbalance in software defect prediction.

Class imbalance in the original data leads to suboptimal classification performance, especially in minority-class detection. SMOTE improves minority-class detection, but can still produce noise and class overlap [39], [40]. The NR-Modified SMOTE approach provides more stable improvements through the noise reduction stage. In contrast, NR-Clustering SMOTE shows the best and most consistent performance, particularly with Random Forest, making it the most effective approach for handling data imbalance in software defect prediction.

Based on the performance evaluation results on the software defect dataset presented in Table 4, using the original imbalanced data yields suboptimal classification performance, particularly in recall and F1-score. Although Random Forests show relatively high accuracy, their ability to detect the minority class is very low. This condition indicates that the dominance of the majority class mainly influences the high accuracy on the original data and does not reflect the model's true ability to identify software defects. After applying SMOTE, classification performance improves significantly for most methods, particularly Random Forest, which achieves high values for accuracy, recall, and F1-score.

The NR-Modified SMOTE approach provides more stable and consistent improvements across all classification methods. Applying noise reduction before oversampling improves data quality, as reflected in higher recall, and F1-score values. In this scenario, Random Forest achieves very high, almost-perfect performance across all evaluation metrics compared to standard SMOTE.

Furthermore, NR-Clustering SMOTE delivers the best and most consistent performance across all testing scenarios. The integration of noise filtering, K-Means clustering, and Manhattan distance–based SMOTE within each cluster creates clearer decision boundaries and reduces class overlap [13]. This approach produces optimal accuracy, recall, and F1-score values, particularly for Random Forest, which achieves the highest performance [41]. Therefore, the results confirm that NR-Clustering SMOTE is the most effective method for improving classification performance in software defect prediction with imbalanced data.

Applying SMOTE shows a very significant performance improvement compared to the original data. On the CM1 dataset with Random Forest, recall increases from 1.22% to 97.19% (an increase of approximately 95.97%) with a standard deviation of 0.26, and F1-score rises from 0% to 93.68% with a standard deviation of 0.17. NR-Modified SMOTE provides more consistent improvements than SMOTE. For Random Forest, accuracy increases from 89.79% to 98.09% (approximately 8.31%) with a standard deviation of 0.10, recall rises from 1.22% to 99.78% with a standard deviation of 0.00, and F1-score reaches 99.13% with a standard deviation of 0.10. NR-Clustering SMOTE performs best across all scenarios. Compared to the original data, Random Forest achieves an accuracy increase of approximately 8.47% (from 89.79% to 98.26%) with a standard deviation of 0.11 and a recall increase of nearly 99% (from 1.22% to 99.96%) with a standard deviation of 0.09.

On the PC1 dataset with Random Forest, recall increases from 22.07% to 97.79% (an increase of approximately 75.72%) with a standard deviation of 0.25, and F1-score rises from 30.90% to 95.84% with a standard deviation of 0.23. NR-Modified SMOTE provides more consistent improvements than SMOTE. For Random Forest, accuracy increases from 93.15% to 98.81% (approximately 5.66%) with a standard deviation of 0.06, recall rises from 22.07% to 99.56% with a standard deviation of 0.09, and the F1-score increased by 67.92% with a standard deviation of 0.06.

NR-Clustering SMOTE performs best across all scenarios. Compared to the original data, Random Forest achieves an accuracy increase of approximately 5.88% (from 93.15% to 99.03%) with a standard deviation of 0.10 and a recall increase of nearly 77.58% (from 22.07% to 99.65%) with a standard deviation of 0.08. The high performance achieved in this study is due to applying the Noise Reduction (NR) approach to remove noise from the minority class and using the K-Means algorithm to cluster the data more effectively. Furthermore, oversampling was performed using SMOTE based on the Manhattan distance within each cluster, resulting in a more representative data distribution and clearer decision boundaries between the minority and majority classes.

Lower standard deviations in the accuracy, recall, and F1-score scores across folds indicate that the proposed method has better performance consistency [42], [43]. This indicates that the model performs more stably and reliably across different test datasets.

Table 4. Performance Results of the Proposed Method with the Random Forest Method on the Defect Software Dataset

Data Balancing Method	Methods	Accuracy (± std)	Recall (± std)	F1-score (± std)
Data Original	CM1	89.79%	1.22%	0.00%
	PC1	93.15%	22.07%	30.90%
SMOTE	CM1	93.44% (±0.18)	97.19% (±0.26)	93.68% (±0.17)
	PC1	95.75% (±0.23)	97.79% (±0.25)	95.84% (±0.23)
NR-Modified SMOTE	CM1	98.10% (±0.10)	99.78% (±0.00)	99.13% (±0.10)
	PC1	98.81% (±0.06)	99.56% (±0.09)	98.82% (±0.06)
NR-Clustering SMOTE	CM1	98.26% (±0.11)	99.96% (±0.09)	98.28% (±0.11)
	PC1	99.03% (±0.10)	99.65% (±0.08)	99.03% (±0.10)

In this section, the t-test is used for hypothesis testing. The t-test used in this study is the paired t-test. The t-test is conducted based on the performance results of the proposed methods, namely NR-Modified SMOTE and NR-Clustering SMOTE, compared to the SMOTE method. Table 5 presents the results of statistical tests using paired t-tests to compare the performance of several imbalance data handling methods, namely NR-Modified SMOTE, NR-Clustering SMOTE, and SMOTE on two different datasets (CM1 and PC1). Tests were conducted on four main evaluation metrics, namely accuracy, recall, and F1-score, with a sample size (n) of 10 and degrees of freedom (df) of 9. The t-table value used was 1.83 as the significance limit.

The results show that the NR-Modified SMOTE and NR-Clustering SMOTE methods significantly outperform the standard SMOTE method across all evaluation metrics and datasets. This is evidenced by t-count values consistently exceeding the critical value (t-table = 1.83), and significance values (Sig) below 0.05. Specifically, the CM1 dataset yielded very high t-values, particularly for the accuracy, and F1-score metrics, indicating strong performance improvements, with the NR-Modified SMOTE method performing best, followed by NR-Clustering SMOTE. A similar pattern was also observed for the PC1 dataset. However, the magnitude of the improvement was relatively smaller than that of CM1, indicating that dataset characteristics also influence the level of performance improvement.

Furthermore, a direct comparison between NR-Clustering SMOTE and NR-Modified SMOTE yielded t-values that were slightly smaller, although they remained statistically significant. This indicates that despite the differences between the two methods, their performance is likely comparable. For example, on the CM1 dataset, the t-value for the recall metric is very low (t = 0.00; Sig = 0.008), indicating that both methods have nearly identical ability to detect the minority class. Overall, these results suggest that the choice between NR-Modified and NR-Clustering depends more on implementation considerations than significant performance differences. Overall, Table 5 confirms that the NR-Clustering SMOTE-based method improves the performance of the Random Forest model compared to NR-Modified SMOTE and conventional SMOTE.

Table 5. Paired t-Test Results Across Methods and Datasets

Dataset	Comparison Method	Accuracy (t)	Recall (t)	F1-score (t)	Sig
CM1	NR-Modified SMOTE vs	76.46	31.18	76.14	0.000
PC1	SMOTE	39.38	19.70	39.44	0.000
CM1	NR-Clustering SMOTE vs	61.63	28.98	60.76	0.000
PC1	SMOTE	41.70	25.61	41.91	0.000
CM1	NR-Clustering SMOTE vs	3.09	0.00	3.10	0.008
PC1	NR-Modified SMOTE	5.53	2.23	5.12	0.000

Table 6 presents a performance comparison between the proposed method and several previous studies in the context of software defect detection. The comparison shows that various machine learning approaches have been used, ranging from conventional methods such as Logistic Regression, KNN, and Naive Bayes to deep learning and ensemble learning approaches such as LSTM, 1D-CNN, XGBoost, and Stacked Ensemble. Although some previous studies reported relatively high accuracy, most did not explicitly address class imbalance and noise in the minority class, which can affect overall model performance. The proposed method, namely Random Forest with NR-Clustering SMOTE, demonstrates better performance than most previous studies,

achieving 98.26% accuracy and 98.28% F1-score of the CM1 dataset.

Meanwhile, Random Forest with NR-Clustering SMOTE outperformed most previous studies on the PC1 dataset, achieving 99.03% accuracy and 99.03% F1-score. This performance improvement indicates that integrating noise reduction and SMOTE-based data balancing using Manhattan distance within each cluster produces a more representative data distribution, thereby improving the model's ability to classify software defect cases more accurately. Therefore, these results confirm the effectiveness of NR-Clustering SMOTE in improving classification performance for software defect detection compared with existing approaches [13].

Table 6. Comparison of the Proposed Method's Performance with Several Previous Studies

Researchers	Methods	Accuracy	F1-score
Albattah & Alzahran [6]	LSTM	87.00%	87.00%
Zain et al. [14]	1D-CNN1	-	97.56%
Shahzad et al. [15]	XGBoost	95.00%	95.00%
Sourav et al. [16]	Logistic Regression	93.80%	-
Pradhan & Muduli [17]	PCA- LDA+ IMPSO-ELM	96.96%	-
Goyal [30]	KNN with Neighborhood-based Under-Sampling (N-US)	95.60%	
Alkhasawneh [19]	RBF	93.99%	95.00%
Priyadarshini [20]	Priority-Based Fuzzy SVM	96.00%	-
Ali et al. [21]	feature selection-based voting ensemble (IECGA)	85.86%	
Febrian et al. [31]	KNN with PSO SMOTE CS CFS CMFS	-	-
Arasteh et al. [23]	KNN with BCOOA	98.20%	-
Babatunde et al. [24]	NB with Dagging	70.80%	80.60%
Patel & Gupta [25]	XGBoost	98.00%	-
Iqbal et al. [26]	Random Forest	89.79%	-
Mehmood et al. [27]	Feature Selection with Logistic Regression	90.56%	-
Meng et al. [28]	Tri_SSDPM	88.5%	87.00%
Mustaqeem & Siddiqui [32]	SVM with SMOTE	98.00%	87.00%
Ramadhani et al. [18]	k-nearest neighbor (kNN) method	-	-
Wintana et al. [22]	C45 with AdaBoost	94.18%	93.70%
Mohil et al. [29]	Stacked Ensemble	88.00%	82.00%
Proposed Method	CM1	98.26%	98.28%
	PC1	99.03%	99.03%

Although some previous studies have demonstrated competitive accuracy, most do not report all evaluation metrics, such as F1-score, thereby limiting the comprehensiveness and depth of their performance analysis. In contrast, the proposed method achieves high, consistent performance across all three metrics, demonstrating not only strong classification performance but also a good balance between precision and recall, as reflected in the F1-score values.

Compared with the study [32], which used SVM with SMOTE and achieved 98.00% accuracy and 87.00% F1-score, the proposed method shows superior performance, especially in terms of the balance between precision and recall. Although the accuracy values obtained are relatively comparable, the significant difference in F1-score indicates that the proposed approach is more effective in handling class imbalance and reducing bias towards the majority class. This shows that the proposed method not only improves overall classification performance but also produces more stable, more representative predictions across classes than the SMOTE-based SVM approach. This superiority can be explained by the proposed method (NR-Clustering SMOTE), which integrates noise filtering, K-Means-based clustering, and Manhattan-distance-based SMOTE oversampling within each cluster, thereby forming clearer decision boundaries and reducing class overlap [13]. In addition, noise removal in the minority class at the initial stage can reduce the formation of noisy synthetic data, producing a more representative, cleaner minority class distribution that is less noisy and less prone to overlap during oversampling.

This study still has several limitations that could be addressed in further research. First, the NR-Clustering SMOTE approach has not been specifically tested on more complex multi-class data schemes, so its generalization ability under these conditions still requires further validation. Second, this study does not provide a comprehensive analysis of the proposed method's computational time, nor has it tested its performance on a large-scale dataset, so its efficiency and scalability cannot be fully evaluated. Third, the method evaluation is still limited to certain classification models and has not been directly compared with other algorithms such as XGBoost, LightGBM, and SVM. Hence, the robustness and generalization ability of the proposed method still need further study across various scenarios.

4. Conclusions

Based on experimental results on the software defect dataset, it can be concluded that class imbalance in the original data significantly reduces the ability of

machine learning models to detect software defects, particularly in recall, and F1-score. Applying data-balancing techniques has been shown to improve classification performance significantly. SMOTE improves Random Forest performance by increasing accuracy by 3.65%, recall by 95.97%, and F1-score by 93.68% for the CM1 dataset. In contrast, on the PC1 dataset, it increases accuracy by 2.60%, recall by 75.72%, and F1-score by 64.94%. Meanwhile, NR-Modified SMOTE achieves more stable performance by integrating a noise-reduction mechanism, with improvements of 8.31% in accuracy, 98.56% in recall, and 99.13% in F1-score on the CM1 dataset. In comparison, on the PC1 dataset, there is an increase of 5.66% in accuracy, 77.49% in recall and 67.92% in F1-score. The proposed method, NR-Clustering SMOTE, shows the highest performance improvement for Random Forest across all evaluation metrics compared to the original data, SMOTE, and NR-Modified SMOTE, with increases of 8.47% in accuracy, 98.74% in recall, and 98.28% in F1-score on the CM1 dataset, while on the PC1 dataset there was an increase in accuracy of 5.88%, recall of 77.58%, and F1-score of 68.13%.

These findings confirm that integrating cluster-based data balancing, the Manhattan distance, and noise-reduction mechanisms produces a more representative data distribution and effectively improves software defect detection performance. Future research is recommended to apply the NR-Clustering SMOTE approach to imbalanced data problems under a multi-class scheme. Furthermore, further research is needed to conduct a more comprehensive analysis of the proposed method's computational time and to test it on large-scale datasets to evaluate its efficiency and scalability. This will allow for an optimal balance between classification performance and computational time. Furthermore, future research can extend this study by evaluating the NR-Clustering SMOTE and NR-Modified SMOTE methods using various classification algorithms, such as XGBoost, LightGBM, and SVM, to further validate the robustness and generalization capabilities of the proposed approach.

References

[1] X. Dong, Y. Liang, S. Miyamoto, and S. Yamaguchi, "Ensemble learning based software defect prediction," *J. Eng. Res.*, vol. 11, no. 4, pp. 377–391, 2023, doi: https://doi.org/10.1016/j.jer.2023.10.038.

[2] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, Data Validation Methods, Approaches, and Tools," *Eng. Appl. Artif. Intell.*, vol. 111, p. 104773, 2022, doi: https://doi.org/10.1016/j.engappai.2022.104773.

[3] Q. Zhi, W. Pu, J. Ren, and Z. Zhou, "A Defect Detection Method for the Primary Stage of Software Development," *Comput. Mater. Contin.*, vol. 74, no. 3, pp. 5141–5155, 2022, doi: https://doi.org/10.32604/cmc.2023.035846.

[4] S. Rehan, B. Al-Bander, and A. Al-Said Ahmad, "Harnessing Large Language Models for Automated Software Testing: A Leap Towards Scalable Test Case Generation," *Electronics*, vol. 14, no. 7, pp. 1–25, Apr. 2025, doi: 10.3390/electronics14071463.

[5] P. Dhavakumar and S. Vengadeswaran, "Software defect prediction using graph sample and aggregate-attention network optimized with nomadic people optimizer for enhancing the software reliability," *Comput. Stand. Interfaces*, vol. 95, p. 104033, 2026, doi: https://doi.org/10.1016/j.csi.2025.104033.

[6] W. Albattah and M. Alzahrani, "Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach," *AI*, vol. 5, no. 4, pp. 1743–1758, Sep. 2024, doi: 10.3390/ai5040086.

[7] G. Cauvery, D. Suresh, G. Aswini, P. Jayanthi, and K. Kalaiselvi, "Software Defect Prediction Using Machine Learning Techniques," in *Advances in Artificial and Human Intelligence in the Modern Era*, 2023, pp. 180–195. doi: 10.4018/979-8-3693-1301-5.ch010.

[8] Y. Gao, Y. Zhu, and Y. Zhao, "Dealing with imbalanced data for interpretable defect prediction," *Inf. Softw. Technol.*, vol. 151, p. 107016, 2022, doi: https://doi.org/10.1016/j.infsof.2022.107016.

[9] A. A. Khan, O. Chaudhari, and R. Chandra, "A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation," *Expert Syst. Appl.*, vol. 244, no. November 2023, p. 122778, 2024, doi: 10.1016/j.eswa.2023.122778.

[10] H. Z. Zaidi *et al.*, "Machine Learning Approaches for Software Defect Prediction," *Appl. Comput. Intell. Soft Comput.*, vol. 2025, no. 1, pp. 1–29, Jan. 2025, doi: 10.1155/acis/7933078.

[11] A. Hakami, "Strategies for overcoming data scarcity, imbalance, and feature selection challenges in machine learning models for predictive maintenance," *Sci. Rep.*, vol. 14, no. 1, pp. 1–14, 2024, doi: 10.1038/s41598-024-59958-9.

[12] T. Widiyaningtyas, H. Hairani, D. D. Prasetya, U. Pujianto, and W. Caesarendra, "A Modified SMOTE with Noise Filtering and Manhattan Distance Metric Approach to Address Imbalanced Health Datasets," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 4, pp. 25452–25459, Aug. 2025, doi: 10.48084/etasr.11925.

[13] H. Hairani, T. Widiyaningtyas, D. D. Prasetya, and A. Aminuddin, "Addressing Imbalance in Health Datasets: A New Method NR-Clustering SMOTE and Distance Metric Modification," *Comput. Mater. Contin.*, vol. 82, no. 2, pp. 2931–2949, 2025, doi: 10.32604/cmc.2024.060837.

[14] K. Pin, J. Ho Chang, and Y. Nam, "Software Defect Prediction Harnessing on Multi 1-Dimensional Convolutional Neural Network Structure," *Comput. Mater. Contin.*, vol. 71, no. 1, pp. 1521–1546, 2022, doi: 10.32604/cmc.2022.022085.

[15] T. Shahzad, S. Khan, T. Mazhar, W. Ahmad, K. Ouahada, and H. Hamam, "Predicting Software Perfection Through Advanced Models to Uncover and Prevent Defects," *IET Softw.*, vol. 2025, no. 1, pp. 1–17, Jan. 2025, doi: 10.1049/sfw2/8832164.

[16] S. Parija, D. Muduli, M. Shameem, and S. K. Sharma, "Enhancing Software Defect Prediction: A Stacking Ensemble Framework with Advanced Feature Engineering," in *Proceedings of the 2025 29th International Conference on Evaluation and Assessment in Software Engineering Companion*, in EASE Companion '25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 28–34. doi: 10.1145/3727967.3756848.

[17] D. Pradhan and D. Muduli, "Improved prediction of software defect: A particle swarm optimization based ELM approach," *e-Prime - Adv. Electr. Eng. Electron. Energy*, vol. 13, no. July, pp. 1–21, Sep. 2025, doi: 10.1016/j.prime.2025.101081.

[18] A. A. P. Ramadhani, R. A. Nugroho, M. R. Faisal, F. Abadi, and R. Herteno, "The impact of software metrics in NASA metric data program dataset modules for software defect prediction," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 22, no. 4, pp. 846–853, 2024, doi: 10.12928/TELKOMNIKA.v22i4.25787.

[19] M. S. Alkhasawneh, "Software Defect Prediction through Neural Network and Feature Selections," *Appl. Comput. Intell. Soft Comput.*, vol. 2022, pp. 1–16, Sep. 2022, doi: 10.1155/2022/2581832.

[20] A. Priyadarshini, "Software Defect Prediction across Multiple Datasets using Classification Techniques," *J. Inf. Syst. Eng. Manag.*, vol. 10, no. 41s, pp. 158–167, May 2025, doi: 10.52783/jisem.v10i41s.7807.
[21] M. Ali, T. Mazhar, A. Al-Rasheed, T. Shahzad, Y. Yasin Ghadi, and M. Amir Khan, "Enhancing software defect prediction: a framework with improved feature selection and ensemble machine learning," *PeerJ Comput. Sci.*, vol. 10, pp. 1–37, Feb. 2024, doi: 10.7717/peerj-cs.1860.
[22] D. Wintana, D. Ismunandar, and E. H. Juningsih, "Improving Software Defect Prediction Performance Using C4 . 5 Based Ensemble Learning with AdaBoost and Bagging Techniques," *J-INTECH (Journal Inf. Technol.*, no. 10, pp. 411–421, 2025, doi: https://doi.org/10.32664/k7fyc413.
[23] B. Arasteh, K. Arasteh, A. Ghaffari, and R. Ghanbarzadeh, "A new binary chaos-based metaheuristic algorithm for software defect prediction," *Cluster Comput.*, vol. 27, no. 7, pp. 10093–10123, Oct. 2024, doi: 10.1007/s10586-024-04486-4.
[24] B. M. Shankar, S. A. Sivakumar, D. Dhabliya, P. A. Sundari, M. Asmitha, and S. M. G. Shree, "Software Defect Prediction Using Dagging Meta-Learner-Based Classifier," in *2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, IEEE, Oct. 2023, pp. 682–686. doi: 10.1109/I-SMAC58438.2023.10290523.
[25] K. Patel and S. Gupta, "Prediction of Software Defects for Quality Assurance and Improvement Based on Machine Learning Methods," in *2024 2nd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, Nov. 2024, pp. 400–406. doi: 10.1109/ICAICCIT64383.2024.10912382.
[26] 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.
[27] I. Mehmood *et al.*, "A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning," *IEEE Access*, vol. 11, no. June, pp. 63579–63597, 2023, doi: 10.1109/ACCESS.2023.3287326.
[28] F. Meng, W. Cheng, and J. Wang, "Semi-supervised Software Defect Prediction Model Based on Tri-training," *KSII Trans. Internet Inf. Syst.*, vol. 15, no. 11, pp. 4028–4042, 2021, doi: 10.3837/TIIS.2021.11.009.
[29] A. S. Mohil, A. Mohil, and M. Mohil, "Stacked Ensemble Learning for Software Defect Prediction : Model Integration and Cross-Dataset Validation," *Int. J. Environ. Sci.*, vol. 10, no. 4, pp. 392–402, 2024, doi: https://theaspd.com/index.php/ijes/article/view/11187.
[30] S. Goyal, "Handling Class-Imbalance with KNN (Neighbourhood) Under-Sampling for Software Defect Prediction," *Artif. Intell. Rev.*, vol. 55, no. 3, pp. 2023–2064, Mar. 2022, doi: 10.1007/s10462-021-10044-w.
[31] M. M. Febrian, S. W. Saputro, T. H. Saragih, F. Abadi, and R. Herteno, "Hybrid Feature Selection and Balancing Data Approach for Improved Software Defect Prediction," *Indones. J. Electron. Electromed. Eng. Med. Informatics*, vol. 7, no. 2, pp. 232–244, Apr. 2025, doi: 10.35882/ijeeemi.v7i2.67.
[32] M. Mustaqeem and T. Siddiqui, "A Hybrid Software Defects Prediction Model for Imbalance Datasets Using Machine Learning Techniques: (S-SVM Model)," *J. Auton. Intell.*, vol. 6, no. 1, pp. 1–19, Jun. 2023, doi: 10.32629/jai.v6i1.559.
[33] V. Teodorescu and L. Obreja Brașoveanu, "Assessing the Validity of k-Fold Cross-Validation for Model Selection: Evidence from Bankruptcy Prediction Using Random Forest and XGBoost," *Computation*, vol. 13, no. 5, pp. 1–22, 2025, doi: 10.3390/computation13050127.
[34] P. Thölke *et al.*, "Class imbalance should not throw you off balance: Choosing the right classifiers and performance metrics for brain decoding with imbalanced data," *Neuroimage*, vol. 277, no. August, pp. 1–14, 2023, doi: https://doi.org/10.1016/j.neuroimage.2023.120253.
[35] T. Prexawanprasut and T. Banditwattanawong, "Improving Minority Class Recall through a Novel Cluster-Based Oversampling Technique," *Informatics*, vol. 11, no. 2, pp. 1–24, May 2024, doi: 10.3390/informatics11020035.
[36] M. Imani, M. Joudaki, A. Bagheri, and H. R. Arabnia, "Why ROC-AUC Alone Is Insufficient for Highly Imbalanced Data: In-Depth Evaluation of MCC, F2-Score, H-Measure, and AUC-Based Metrics for Rare-Event Binary Classification," *Technologies*, vol. 14, no. 1, pp. 1–41, 2025, doi: https://doi.org/10.3390/technologies14010054.
[37] A. A. Khan, O. Chaudhari, and R. Chandra, "A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation," *Expert Syst. Appl.*, vol. 244, no. May, pp. 1–29, 2024, doi: https://doi.org/10.1016/j.eswa.2023.122778.
[38] Z. Sun, J. Zhang, X. Zhu, and D. Xu, "How Far Have We Progressed in the Sampling Methods for Imbalanced Data Classification ? An Empirical Study," *Electronics*, vol. 12, no. 20, pp. 1–21, 2023, doi: https://doi.org/10.3390/electronics12204232.
[39] G. Husain *et al.*, "SMOTE vs. SMOTEENN: A Study on the Performance of Resampling Algorithms for Addressing Class Imbalance in Regression Models," *Algorithms*, vol. 18, no. 1, pp. 1–16, 2025, doi: 10.3390/a18010037.
[40] Y. Li, Y. Yang, P. Song, L. Duan, and R. Ren, "An improved SMOTE algorithm for enhanced imbalanced data classification by expanding sample generation space," *Sci. Rep.*, vol. 15, no. July, pp. 1–21, 2025, doi: https://doi.org/10.1038/s41598-025-09506-w.
[41] D. Priyanto, H. Hairani, K. Marzuki, and M. Innuddin, "Optimization of Random Forest for Health Data Classification Using PCA and K-Means SMOTE-ENN," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 5, pp. 27646–27652, 2025, doi: 10.48084/etasr.12976.
[42] F. Kazerouni, A. Bayani, F. Asadi, L. Saeidi, N. Parvizi, and Z. Mansoori, "Type2 diabetes mellitus prediction using data mining algorithms based on the long-noncoding RNAs expression: A comparison of four data mining approaches," *BMC Bioinformatics*, vol. 21, no. 1, pp. 1–13, 2020, doi: 10.1186/s12859-020-03719-8.
[43] D. Z. Abidin, L. Afuan, A. N. Toscany, and N. Nurhadi, "A Comprehensive Benchmarking Pipeline for Transformer-Based Sentiment Analysis using Cross-Validated Metrics," *J. Tek. Inform.*, vol. 6, no. 4, pp. 1797–1810, 2025, doi: 10.52436/1.jutif.2025.6.4.4894.