Scientific Journal of Informatics
Vol. 11, No. 4, Nov 2024
p-ISSN 2407-7658 https://journal.unnes.ac.id/journals/sji/index e-ISSN 2460-0040

Improve Software Defect Prediction using Particle Swarm Optimization and Synthetic Minority Over-sampling Technique

Afif Amirullah[1*], Umi Laili Yuhana[2], Muhammad Alfian[3]
[1, 2, 3]Department of Informatic Engineering, Institut Teknologi Sepuluh Nopember, Indonesia

Abstract.
Purpose: Early detection of software defects is essential to prevent problems with software maintenance. Although much machine learning research has been used to predict software defects, most have not paid attention to the problems of data imbalance and feature correlation. This research focuses on overcoming the problems of imbalance dataset. It provides new insights into the impact of these two feature extraction techniques in improving the accuracy of software defect prediction.
Methods: This research compares three algorithms: Random Forest, Logistic Regression, and XGBoost, with the application of PSO for feature selection and SMOTE to overcome the problem of imbalanced data. Comparison of algorithm performance is measured using F1-Score, Precision, Recall, and Accuracy metrics to evaluate the effectiveness of each approach.
Result: This research demonstrates the potential of SMOTE and PSO techniques in enhancing the performance of software defect detection models, particularly in ensemble algorithms like Random Forest (RF) and XGBoost (XGB). The application of SMOTE and PSO resulted in a significant increase in RF accuracy to 87.63%, XGB to 85.40%, but a decrease in Logistic Regression (LR) accuracy to 72.98%. The F1-Score, Precision, and Recall metrics showed substantial improvements in RF and XGB, but not in LR due to the decrease in accuracy, highlighting the impact of the research findings.
Novelty: Based on the comparison results, it is proven that the SMOTE and PSO algorithms can improve the Random Forest and XGB models for predicting software defect.

Keywords: Software defect, Particle swarm optimization, Synthetic minority oversampling technique, Imbalanced dataset
Received December 2024 / **Revised** March 2025 / **Accepted** March 2025

INTRODUCTION

Software defect detection is critical because it can anticipate worse risks in the future [1]-[3]. Software defects not detected in the early stages of development can cause serious problems, such as financial losses, product quality deterioration, and complete system failure after implementation. By detecting software defects early, developers can identify and fix errors before the software is released to users, thereby reducing the risk and cost of repairs in the future. This timely detection also plays a vital role in ensuring the reliability and security of increasingly complex software systems as technology evolves [4].

Along with technology development, detecting software defects is now increasingly influential through machine learning techniques [3], [5], [6]. This technology allows the system to learn patterns from historical data, automatically identify potential software flaws, and provide more accurate predictions than traditional methods. In the context of software engineering, machine learning is not only able to reduce manual effort in detecting errors [7]-[9]. However, it can also improve the speed and accuracy of finding and handling software defects. Therefore, machine learning is a promising solution to optimize the software development process by reducing the risk of failure and improving the quality of the final product.

Many classification methods can be reliably used to predict various case studies, including the prediction of software defects. Among the methods often used are XGBoost, Random Forest, and Logistic Regression. XGBoost is well-known for its ability to handle complex data and provide accurate predictions through a boosting approach [10], [11]. The effectiveness of these methods instills confidence in their users [12],

*Corresponding author.
Email addresses: 6025241048@student.its.ac.id (Afif)
DOI: 10.15294/sji.v11i4.16808

[13]. With its ensemble learning approach, Random Forest can overcome the problem of overfitting and provide stable predictions [14]-[16]. Meanwhile, Logistic Regression, although more straightforward, remains a popular choice in predicting binary variables such as the presence or absence of software defects [17], [18]. These three methods have proven effective in various studies and provide reliable results in software defect detection, helping significantly improve the quality of software development.

In addition to reliable algorithms, a good prediction model also requires a quality dataset [19], [20]. A good dataset should be well-distributed and have relevant features and influence on the label without any risk of bias [21]. However, often in practice, the datasets used are imbalanced, where each class's data differs significantly [22]. This condition can affect the prediction model's performance, especially in detecting software defects. To solve this problem, sampling techniques, such as oversampling or undersampling, are needed to balance the data distribution. In addition, the feature selection process is also critical to ensure that only the genuinely influential features are used in the model, which can reduce model complexity and improve prediction accuracy [23], [24]. The combination of proper dataset selection and the use of powerful algorithms can result in a more effective model.

Many studies have explored using machine learning models to predict software defects. However, most studies do not specifically highlight the problems of data imbalance and feature selection, although both are important factors in improving prediction accuracy [5], [25]. Data imbalance can make the model more likely to be biased toward the majority class, thereby reducing the model's ability to recognize the minority class. In addition, using irrelevant features can decrease model performance by increasing complexity without significantly contributing to the prediction results. Therefore, strategies to deal with data imbalance and select relevant features are necessary to produce more reliable and accurate prediction models [19], [22], [23].

This study proposes using the SMOTE (Synthetic Minority Over-sampling Technique) algorithm to handle imbalanced data and PSO (Particle Swarm Optimization) for feature selection and combine with XBoost, Random Forest, and Logistic regression for the classifier algorithm to answer the research gap. Then, the model evaluation will use a confusion matrix to determine the model's precision, recall, F-1 Score, and accuracy. Combining these two techniques is expected to improve the performance of software defect prediction models, making the results more accurate and reliable.

Literature review

Several studies have examined the use of machine learning algorithms in detecting software defects [7], [25]-[27]. Software defect prediction (SDP) is an important element in ensuring software quality is maintained, with various machine learning-based approaches continuing to be developed. One widely used approach is Random Forest (RF), where using Bayesian Optimization to tune hyperparameters has increased accuracy by up to 85%, making it more effective than conventional RF [26]. Boosting algorithm-based approaches are also increasingly popular, such as Extreme Gradient Boosting (XGBoost), which, when optimized using metaheuristics such as Particle Swarm Optimization (PSO), can achieve up to 90% accuracy [27]. Additionally, the integration of Shapley Additive Explanations (SHAP) analysis provides clarity about feature contributions, which is critical in making data-driven decisions. Logistic Regression (LR) remains relevant for a simpler approach, especially when combined with the Gradient Descent algorithm for cost computing. Research shows that LR with this approach achieves an accuracy of around 78%, making it an efficient solution for low-complexity datasets [25]. Equally important, research utilizing novel structural feature engineering in object-based software shows that this approach can improve cross-project prediction accuracy by up to 82%, highlighting the importance of quality features in improving cross-domain model generalization and enlightening the audience about this crucial aspect [7].

One effective approach to improve model performance is using Particle Swarm Optimization (PSO) for parameter optimization. This technique has been proven to improve model accuracy through adaptive exploration of global and local solutions, as shown in a study that modified PSO with smart particles to solve inverse problems in electromagnetic devices [19]. Using multi-swarm PSO operating in two stages also provides optimal results, especially in solving global optimization problems without constraints [23]. The combination of PSO with fuzzy hybrid techniques in the medical field has proven this approach's superiority so that it can be adapted for software defect prediction with higher accuracy [28].

In addition, to handle data imbalances that often occur in SDP datasets, the Synthetic Minority Oversampling Technique (SMOTE) is a solution that is often applied. This method can be improved with techniques such as Improved Random SMOTE, which shows an excellent ability to handle data imbalance by improving the representation of minority data [29]. A recent Bayesian Optimization-based approach, BO-SMOTE, also offers significant improvements in oversampling by optimally tuning parameters, improving evaluation metrics such as accuracy, precision, and sensitivity on imbalanced datasets [22]. The combination of PSO and SMOTE has the potential to provide a comprehensive solution to improve the performance of software defect prediction models, especially in increasing accuracy in complex and imbalanced data scenarios.

METHODS

This research proposed a new method to improve model performance to detect software defects by using feature selection and imbalanced data handling. The research method shown in figure 1 combines the SMOTE and PSO algorithms for data preprocessing. Then, for data classification, the XGBoost algorithm, Logistic Regression, and Randomized Search CV method are used to find the best parameter of every classification algorithm. This research proposed a new method to improve model performance and detect software defects using feature selection and imbalanced data handling. Ultimately, the model evaluation will compare and measure each model before and after using SMOTE and PSO.

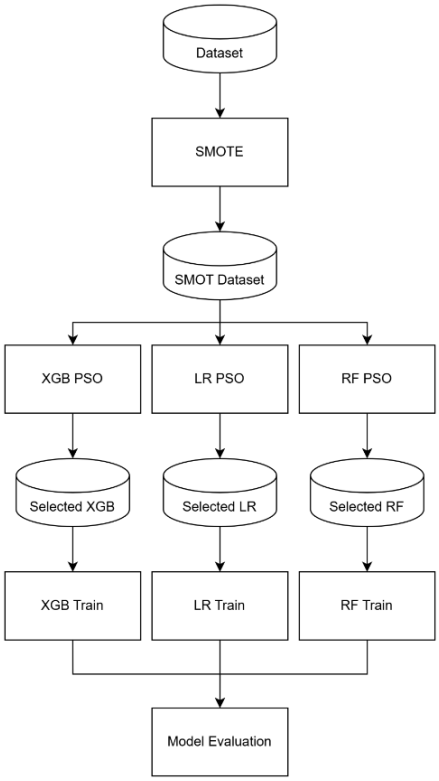

Figure 1. Research design

Dataset and feature

This study used a dataset from Kaggle, Software Defect Prediction with Binary Class which used in various research previously [30]-[33]. The dataset contains 22 features and *defects* columns as label. Features in the dataset include *loc*, which provides information about the number of lines of code, and *v(g)*, which represents the program's cyclomatic complexity or logical complexity. *ev(g)* describes how well the code is organized in the program structure, while *iv(g)* reflects the complexity of the program's flow. The *n* feature refers to the total number of operators and operants in the program, and *l* represents the total length of the program. *d* indicates difficulty understanding the program and represents the intelligence required to write or solve problems in the program. *e* represents the effort required to understand or write the program, and *b* represents the total number of bugs in the program. The *t* feature refers to the time it takes to write a program, whereas *IOcode* indicates an executable line of code.

Table 1. Dataset feature

No.	Columns	Dtype
1	loc	float64
2	v(g)	float64
3	ev(g)	float64
4	iv(g)	float64
5	n	float64
6	v	float64
7	l	float64
8	d	float64
9	i	float64
10	e	float64
11	b	float64
12	t	float64
13	lOCode	int64
14	lOComment	int64
15	lOBlank	int64
16	locCodeAndComment	int64
17	uniq_Op	object
18	uniq_Opnd	object
19	total_Op	object
20	total_Opnd	object
21	branchCount	object
22	defects	bool

Synthetic minority oversampling technique

The first data preprocessing process performed in this experiment is the SMOTE process. SMOTE (Synthetic Minority Over-sampling Technique) is a method used to handle the problem of class imbalance in datasets that often occurs in classification problems, especially when the minority class is much less than the majority class [34]. This imbalance can lead to model bias, where the model is more likely to predict the majority class than the minority class. SMOTE addresses this issue by generating synthetic samples for the minority class instead of simply duplicating the data. Synthetic samples are created by selecting the nearest neighbor of each minority class data point and creating new samples between those points through interpolation.

Particle swarm optimization

Particle Swarm Optimization (PSO) is a population-based optimization method inspired by the social behavior of animals, such as birds and fish. In machine learning, PSO is beneficial for optimizing model hyperparameters, which are essential in model performance and accuracy [35], [36]. The PSO process involves a group of "particles" representing a potential solution in the search space. Each particle has an updated position and velocity based on two main components: the best experience ever achieved by the particle itself (best) and the best experience known to the group (best). The advantage of PSO lies in its ability to simultaneously explore and exploit solutions, which allows it to find optimal solutions in complex multidimensional spaces. PSO is also recognized as a relatively easy algorithm to implement, as it requires only a few parameters to be tuned compared to other optimization algorithms.

Training model

The research method to be used starts with the feature extraction step on the prepared dataset. After the feature extraction process is complete, the next step is to train the model using three machine learning algorithms: XGBoost, Random Forest (RF), and Logistic Regression (LogReg). Hyperparameter tuning uses the RandomizedSearchCV method to improve the model's performance. This method allows for the best hyperparameter search within a sizeable random search space, thereby reducing computation time compared to a grid search across the board. Each algorithm will be tested with different parameters to find the best combination. In this tuning process, three cross-validation (CVs) are carried out using 30 iterations per CV to ensure the stability and generalization of the model against previously unseen data.

Model evaluation

After completing data training, the final step is to perform model evaluation, which will be conducted using the confusion matrix method. The confusion matrix consists of True Positives (TP), which represent correct optimistic predictions; True Negatives (TN), which represent correct pessimistic predictions; False Positives (FP), which are incorrect optimistic predictions (also known as false alarms); and False Negatives (FN), which are incorrect pessimistic predictions. Key metrics such as accuracy (01), precision (02), recall

(03), and F1-score (04) can be calculated from the confusion matrix to assess overall model performance, identify errors, and improve prediction quality [37], [38].

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

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1-Score = 2 \text{ X } \frac{PRECISION \text{ X } RECALL}{PRECISION+RECALL} \quad (4)$$

Model evaluation using the confusion matrix is one of the most commonly used methods in machine learning research, and the reliability of its measurements is widely accepted [39], [40].

RESULTS AND DISCUSSIONS

This research using dataset Software Defect Prediction with Binary Class from Kaggle and Defect column that contain defect status used as label in this dataset. Results of this research will present an analysis of the imbalance handling, feature selection process and the evaluation outcomes for each algorithm.

Imbalanced handling

The software defect dataset used has a total value of 101,763 with an unbalanced class distribution, with 78,699 data in the majority class and 23,064 data in the minority class. This distribution imbalance can negatively affect the prediction model's performance, especially in predicting minority classes, which are often more susceptible to being ignored by machine learning algorithms. The distribution before implementing SMOTE is shown in Figure 2.

Figure 2. Data distribution before SMOTE

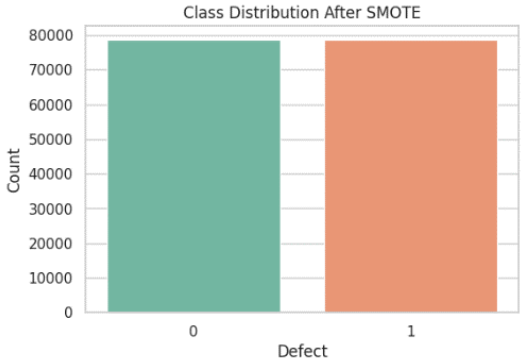

Figure 3. Data distribution after SMOTE

The balancing method used in this research is oversampling, where the data of the minority class is increased until it reaches an amount equivalent to the majority class. This oversampling technique works by randomly replicating samples from the minority class to equal the total amount of data in both classes. In this case, after oversampling, the composition of the minority data was adjusted to be equal to the majority class, which had 78,699 data for each class. Figure 3 showed after the implementation of SMOTE. The distribution of data after using SMOTE shows improvement, with a more balanced class distribution. This method allows for a more balanced data distribution, allowing machine learning models to learn more fairly and be less biased towards the majority class.

Feature selection

In this research, the feature extraction stage using the Particle Swarm Optimization (PSO) algorithm plays a vital role in the data preprocessing process. PSO was chosen for its ability to optimize feature selection quickly and efficiently. This algorithm simulates the social behavior of a group of particles moving in the search space to find the best solution, which, in this case, is the most relevant feature. This process is done by combining PSO with several classification algorithms, such as Extreme Gradient Boosting (XGB),

Logistic Regression (LR), and Random Forest (RF). Each combination of these algorithms has different feature selection results, depending on how each algorithm utilizes the features available in the dataset to build an accurate and practical model.

Table 2. Feature comparison

XGB_PSO	LR_PSO	RF_PSO
loc	loc	loc
v(g)	v(g)	ev(g)
ev(g)	l	iv(g)
iv(g)	d	n
n	i	v
l	lOCode	l
d	lOComment	d
b	uniq_Op	i
lOComment	total_Op	b
locCodeAndComment	branchCount	lOCode
uniq_Op		lOComment
uniq_Opnd		locCodeAndComment
total_Op		uniq_Op
		uniq_Opnd
		branchCount

Overall, using PSO as a feature selection method proved effective in reducing the dimensionality of the data and selecting the most relevant features for each classification algorithm. Each algorithm has a different approach to utilizing features, depending on the characteristics of the model. XGB, which prioritizes data truncation as a decision tree, tends to select features that are more directly related to code complexity. As a simpler model, LR focuses on features that provide linear information and can be translated into superficial relationships between features and targets. With its ensemble power, RF can handle more varied features and utilize code complexity more comprehensively. Thus, PSO reduces computation time and significantly improves model performance.

Evaluation

The application of the SMOTE and PSO techniques in this research clearly shows how these two techniques can improve model performance in detecting software defects. SMOTE overcomes the problem of class imbalance by adding samples to the minority class so that the model obtains more balanced information and is not biased towards the majority class. This improves evaluation metrics such as Precision, Recall, and F1 Score. On the other hand, PSO functions as a feature selection technique that selects the most relevant features, reduces redundancy and prevents overfitting. Using SMOTE and PSO significantly improves ensemble algorithms such as Random Forest (RF) and XGBoost (XGB), especially in terms of accuracy, Precision, Recall, and F1 Score.

Table 3. Evaluation score

Model	Accuracy	Precision	Recall	F1-Score
XGB	81.18%	0.625	0.384	0.476
RF	80.93%	0.616	0.377	0.373
LR	78.63%	0.537	0.286	0.373
XGB_SMOTE_PSO	85.39%	**0.908**	0.789	**0.873**
RF_SMOTE_PSO	**87.63%**	0.901	**0.847**	**0.873**
LR_SMOTE_PSO	72.98%	0.772	0.655	0.709

Comparison of each model showed in Table 3. Model comparison. Without the application of SMOTE and PSO techniques, the evaluation results of the Logistic Regression (LR), Random Forest (RF), and XGBoost (XGB) models show significant variations in performance. The highest accuracy was achieved by the XGBoost model with a value of 81.18%, followed by Random Forest at 80.93% and Logistic Regression at 78.63%. Based on the F1 Score, XGBoost is also superior, with a value of 0.476, followed by Random Forest and Logistic Regression, which have the same value, 0.373. Regarding Precision, XGBoost has the

highest value of 0.625, while Random Forest is in second place with 0.616 and Logistic Regression with 0.537. However, in the Recall metric, XGBoost obtained the highest value of 0.384, followed by Random Forest with a value of 0.377 and Logistic Regression with the lowest value of 0.286. These results show that ensemble models such as XGBoost and Random Forest, without SMOTE and PSO, tend to perform better than Logistic Regression, especially in dealing with data synchronization and identifying relevant features.

By applying the SMOTE and PSO techniques, the performance of the three models shows a significant improvement, especially the Random Forest (RF) and XGBoost (XGB) models. The highest accuracy was achieved by Random Forest with a value of 87.63%, followed by XGBoost at 85.39%, while Logistic Regression (LR) experienced a decrease in accuracy to 72.98%. The F1 Score also shows a significant increase in Random Forest and XGBoost, amounting to 0.873 and 0.844, respectively, while Logistic Regression has an F1 Score of 0.709. The highest precision was achieved by XGBoost with a value of 0.908, Random Forest at 0.901 and Logistic Regression at 0.772. Regarding Recall, Random Forest is superior with a value of 0.847, followed by XGBoost at 0.789 and Logistic Regression at 0.655.

Table 4. Related work comparison

Study	Highest Accuracy
This study	**87.63%**
Shrimankar et al. 2022 [33]	85.08%
Shailee et al. 2024 [32]	82.00%
[41]	80.44%
Khan et al. 2021 [42]	82.02%

Then, compared to the other related work, this proposed method's highest accuracy model achieved higher than another model in previous research. Again, it proves that balancing the dataset and selecting the related feature using SMOTE and PSO are essential to improve model performance. These findings reinforce the critical role of data preprocessing and feature selection in machine learning pipelines for software defect prediction. This study outperformed the benchmarks established by prior works by adopting advanced techniques such as SMOTE and PSO. It set a precedent for future research leveraging these methods to address similar challenges.

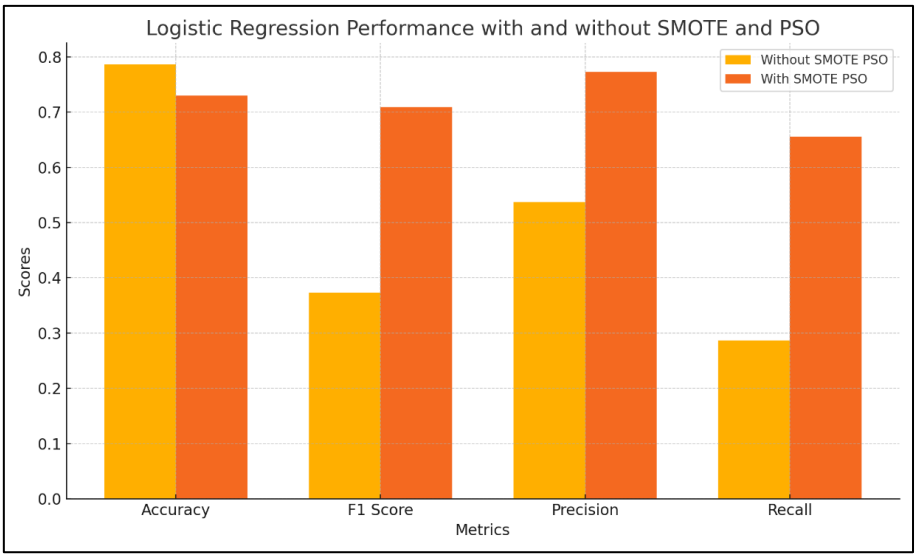

Figure 4. Logistic Regression comparison diagram

However, the results obtained from Logistic Regression (LR) in Figure. show a different phenomenon. When SMOTE and PSO are applied, LR performance especially in accuracy decreases, indicating that the algorithm does not always benefit from either technique in the context of software defect detection. Several factors may explain the decline in accuracy and other evaluation metrics in LR. First, Logistic Regression is an inherently simple algorithm based on the assumption of a linear relationship between a feature and a target [17], [25]. When SMOTE is used to generate synthetic samples in minority classes, the data distribution can become more complex, and the linear relationships assumed by LR may no longer be valid

or can be studied effectively. SMOTE can overfit models on synthetic samples, which often do not fully reflect the original patterns in the data, leading to poor performance.

CONCLUSION

This research presents a proposed method by adding PSO for feature selection and SMOTE to handle imbalanced datasets to improve the model's performance in predicting software defects and compare it with models that do not use. Results of the study proved the proposed method by handling imbalanced dataset using SMOTE and feature selection using PSO can improve the model of machine learning to predict software defect in several model. Before the application of SMOTE and PSO techniques, the XGBoost (XGB) model showed the highest accuracy of 81.18%, followed by Random Forest (RF) with 80.93%, and Logistic Regression (LR) at 78.63%. XGBoost also excels in the F1 Score with a score of 0.476, while RF and LR have the same value of 0.373. However, after the implementation of SMOTE and PSO, RF and XGB performance improved significantly, with RF achieving 87.63% accuracy and XGB 85.39%, while LR experienced a decrease in accuracy to 72.98%. Overall, the implementation of SMOTE and PSO significantly improved F1 Score, Precision, and Recall on the RF and XGB models, especially in handling unbalanced data and feature selection, But interestingly, different results were obtained in the LR model which showed a decrease after the use of SMOTE PSO. This indicates that although the SMOTE PSO method is improved in ensemble models, it cannot work optimally for simple algorithms such as logistic regression. As a further direction of research, it is important to explore more deeply how SMOTE and PSO techniques can be adapted to simpler models such as logistic regression to avoid performance degradation. Further research can also investigate combinations of other techniques to address data imbalances and feature selection that better suits the characteristics of each model.

REFERENCES

[1] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 356–370, 2011, doi: 10.1109/TSE.2010.90.

[2] F. Huang and L. Strigini, "HEDF: A Method for Early Forecasting Software Defects Based on Human Error Mechanisms," *IEEE Access*, vol. 11, pp. 3626–3652, 2023, doi: 10.1109/ACCESS.2023.3234490.

[3] K. Okumoto, "Early Software Defect Prediction: Right-Shifting Software Effort Data into a Defect Curve," in *Proceedings - 2022 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 43–48. doi: 10.1109/ISSREW55968.2022.00037.

[4] S. Huda *et al.*, "A Framework for Software Defect Prediction and Metric Selection," *IEEE Access*, vol. 6, pp. 2844–2858, Dec. 2017, doi: 10.1109/ACCESS.2017.2785445.

[5] Y. Wang, S. Huang, Y. Xu, and H. Zhang, "Research on Software Defects Prediction and Peformance Analysis based on Machine Learning," in *2024 7th International Conference on Computer Information Science and Application Technology, CISAT 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 774–777. doi: 10.1109/CISAT62382.2024.10695196.

[6] X. Dong, Y. Liang, S. Miyamoto, and S. Yamaguchi, "Ensemble learning based software defect prediction," *Journal of Engineering Research (Kuwait)*, vol. 11, no. 4, pp. 377–391, Dec. 2023, doi: 10.1016/j.jer.2023.10.038.

[7] M. Singh and J. K. Chhabra, "Machine learning based improved cross-project software defect prediction using new structural features in object oriented software," *Appl Soft Comput*, vol. 165, Nov. 2024, doi: 10.1016/j.asoc.2024.112082.

[8] C. M. Liapis, A. Karanikola, and S. Kotsiantis, "Data-efficient software defect prediction: A comparative analysis of active learning-enhanced models and voting ensembles," *Inf Sci (N Y)*, vol. 676, Aug. 2024, doi: 10.1016/j.ins.2024.120786.

[9] A. Mishra and A. Sharma, "Deep learning based continuous integration and continuous delivery software defect prediction with effective optimization strategy," *Knowl Based Syst*, vol. 296, Jul. 2024, doi: 10.1016/j.knosys.2024.111835.

[10] D. K. Thai, D. N. Le, Q. H. Doan, T. H. Pham, and D. N. Nguyen, "A hybrid model for classifying the impact damage modes of fiber reinforced concrete panels based on XGBoost and Horse Herd Optimization algorithm," *Structures*, vol. 60, Feb. 2024, doi: 10.1016/j.istruc.2024.105872.

[11] Q. Yao, Y. Tu, J. Yang, and M. Zhao, "Hybrid XGB model for predicting unconfined compressive strength of solid waste-cement-stabilized cohesive soil," *Constr Build Mater*, vol. 449, Oct. 2024, doi: 10.1016/j.conbuildmat.2024.138242.

[12] A. S. Wibowo, H. Tayara, and K. T. Chong, "XGB5hmC: Identifier based on XGB model for RNA 5-hydroxymethylcytosine detection," *Chemometrics and Intelligent Laboratory Systems*, vol. 238, Jul. 2023, doi: 10.1016/j.chemolab.2023.104847.

[13] Q. Zhang, P. Liu, X. Wang, Y. Zhang, Y. Han, and B. Yu, "StackPDB: Predicting DNA-binding proteins based on XGB-RFE feature optimization and stacked ensemble classifier," *Appl Soft Comput*, vol. 99, Feb. 2021, doi: 10.1016/j.asoc.2020.106921.

[14] S. Rajeashwari and K. Arunesh, "Chronic disease prediction with deep convolution based modified extreme-random forest classifier," *Biomed Signal Process Control*, vol. 87, Jan. 2024, doi: 10.1016/j.bspc.2023.105425.

[15] A. F. Amiri, H. Oudira, A. Chouder, and S. Kichou, "Faults detection and diagnosis of PV systems based on machine learning approach using random forest classifier," *Energy Convers Manag*, vol. 301, Feb. 2024, doi: 10.1016/j.enconman.2024.118076.

[16] P. S. Rao, P. Parida, G. Sahu, and S. Dash, "A multi-view human gait recognition using hybrid whale and gray wolf optimization algorithm with a random forest classifier," *Image Vis Comput*, vol. 136, Aug. 2023, doi: 10.1016/j.imavis.2023.104721.

[17] F. Abramovich, V. Grinshtein, and T. Levy, "Multiclass classification by sparse multinomial logistic regression," *IEEE Trans Inf Theory*, vol. 67, no. 7, pp. 4637–4646, Jul. 2021, doi: 10.1109/TIT.2021.3075137.

[18] P. A. Gutiérrez, C. Hervás-Martínez, and F. J. Martínez-Estudillo, "Logistic regression by means of evolutionary radial basis function neural networks," *IEEE Trans Neural Netw*, vol. 22, no. 2, pp. 246–263, Feb. 2011, doi: 10.1109/TNN.2010.2093537.

[19] R. A. Khan, S. Yang, S. Fahad, S. U. Khan, and Kalimullah, "A Modified Particle Swarm Optimization with a Smart Particle for Inverse Problems in Electromagnetic Devices," *IEEE Access*, vol. 9, pp. 99932–99943, 2021, doi: 10.1109/ACCESS.2021.3095403.

[20] D. C. Li, S. Y. Wang, K. C. Huang, and T. I. Tsai, "Learning class-imbalanced data with region-impurity synthetic minority oversampling technique," *Inf Sci (N Y)*, vol. 607, pp. 1391–1407, Aug. 2022, doi: 10.1016/j.ins.2022.06.067.

[21] H. Yi, Q. Jiang, X. Yan, and B. Wang, "Imbalanced Classification Based on Minority Clustering Synthetic Minority Oversampling Technique with Wind Turbine Fault Detection Application," *IEEE Trans Industr Inform*, vol. 17, no. 9, pp. 5867–5875, Sep. 2021, doi: 10.1109/TII.2020.3046566.

[22] S. Yan, Z. Zhao, S. Liu, and M. Zhou, "BO-SMOTE: A Novel Bayesian-Optimization-Based Synthetic Minority Oversampling Technique," *IEEE Trans Syst Man Cybern Syst*, vol. 54, no. 4, pp. 2079–2091, Apr. 2024, doi: 10.1109/TSMC.2023.3335241.

[23] Q. Zhao and C. Li, "Two-Stage Multi-Swarm Particle Swarm Optimizer for Unconstrained and Constrained Global Optimization," *IEEE Access*, vol. 8, pp. 124905–124927, 2020, doi: 10.1109/ACCESS.2020.3007743.

[24] J. H. Seo, C. H. Im, S. Y. Kwak, C. G. Lee, and H. K. Jung, "An improved particle swarm optimization algorithm mimicking territorial dispute between groups for multimodal function optimization problems," *IEEE Trans Magn*, vol. 44, no. 6, pp. 1046–1049, Jun. 2008, doi: 10.1109/TMAG.2007.914855.

[25] R. Suryawanshi and A. Kadam, "Software Defect Prediction by Logistic Regression with Gradient Descent Cost Computation," in *2024 International Conference on Emerging Smart Computing and Informatics, ESCI 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ESCI59607.2024.10497199.

[26] Y. Shen, S. Hu, S. Cai, and M. Chen, "Software Defect Prediction based on Bayesian Optimization Random Forest," in *Proceedings - 2022 9th International Conference on Dependable Systems and Their Applications, DSA 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1012–1013. doi: 10.1109/DSA56465.2022.00149.

[27] T. Zivkovic, B. Nikolic, V. Simic, D. Pamucar, and N. Bacanin, "Software defects prediction by metaheuristics tuned extreme gradient boosting and analysis based on Shapley Additive Explanations," *Appl Soft Comput*, vol. 146, Oct. 2023, doi: 10.1016/j.asoc.2023.110659.

[28] Nivedita, R. Garg, S. Agrawal, A. Sharma, and M. K. Sharma, "Fuzzy hybrid approach for advanced teaching learning technique with particle swarm optimization in the diagnostic of dengue disease," *Systems and Soft Computing*, vol. 6, Dec. 2024, doi: 10.1016/j.sasc.2024.200160.

[29] G. Wei, W. Mu, Y. Song, and J. Dou, "An improved and random synthetic minority oversampling technique for imbalanced data," *Knowl Based Syst*, vol. 248, Jul. 2022, doi: 10.1016/j.knosys.2022.108839.

[30] A. Iqbal and S. Aftab, "A classification framework for software defect prediction using multi-filter feature selection technique and MLP," *International Journal of Modern Education and Computer Science*, vol. 12, no. 1, pp. 18–25, 2020, doi: 10.5815/ijmecs.2020.01.03.
[31] B. Khan *et al.*, "Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques," *J Healthc Eng*, vol. 2021, 2021, doi: 10.1155/2021/8899263.
[32] N. M. Shailee, A. Alam, T. Ahmed, R. Al Mamun Rudro, and K. Nur, "Software Bug Prediction using Machine Learning on JM1 Dataset," in *2024 International Conference on Advances in Computing, Communication, Electrical, and Smart Systems: Innovation for Sustainability, iCACCESS 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/iCACCESS61735.2024.10499572.
[33] R. Shrimankar, M. Kuanr, J. Piri, and N. Panda, "Software Defect Prediction: A Comparative Analysis of Machine Learning Techniques," in *Proceedings - 2022 International Conference on Machine Learning, Computer Systems and Security, MLCSS 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 38–47. doi: 10.1109/MLCSS57186.2022.00016.
[34] G. Wei, W. Mu, Y. Song, and J. Dou, "An improved and random synthetic minority oversampling technique for imbalanced data," *Knowl Based Syst*, vol. 248, Jul. 2022, doi: 10.1016/j.knosys.2022.108839.
[35] R. A. Khan, S. Yang, S. Fahad, S. U. Khan, and Kalimullah, "A Modified Particle Swarm Optimization with a Smart Particle for Inverse Problems in Electromagnetic Devices," *IEEE Access*, vol. 9, pp. 99932–99943, 2021, doi: 10.1109/ACCESS.2021.3095403.
[36] Q. Zhao and C. Li, "Two-Stage Multi-Swarm Particle Swarm Optimizer for Unconstrained and Constrained Global Optimization," *IEEE Access*, vol. 8, pp. 124905–124927, 2020, doi: 10.1109/ACCESS.2020.3007743.
[37] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-Label Confusion Matrix," *IEEE Access*, vol. 10, pp. 19083–19095, 2022, doi: 10.1109/ACCESS.2022.3151048.
[38] M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani, and R. Budiarto, "Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking," *IEEE Access*, vol. 8, pp. 90847–90861, 2020, doi: 10.1109/ACCESS.2020.2994222.
[39] D. Krstinic, L. Seric, and I. Slapnicar, "Comments on 'MLCM: Multi-Label Confusion Matrix,'" 2023, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2023.3267672.
[40] M. Ohsaki, P. Wang, K. Matsuda, S. Katagiri, H. Watanabe, and A. Ralescu, "Confusion-matrix-based kernel logistic regression for imbalanced data classification," *IEEE Trans Knowl Data Eng*, vol. 29, no. 9, pp. 1806–1819, Sep. 2017, doi: 10.1109/TKDE.2017.2682249.
[41] A. Iqbal and S. Aftab, "A classification framework for software defect prediction using multi-filter feature selection technique and MLP," *International Journal of Modern Education and Computer Science*, vol. 12, no. 1, pp. 18–25, 2020, doi: 10.5815/ijmecs.2020.01.03.
[42] B. Khan *et al.*, "Software Defect Prediction for Healthcare Big Data: An Empirical Evaluation of Machine Learning Techniques," *J Healthc Eng*, vol. 2021, 2021, doi: 10.1155/2021/8899263.