Journal of Electronics, Electromedical Engineering, and Medical Informatics
Multidisciplinary: Rapid Review: Open Access Journal Vol. 6, No. 2, April 2024, pp: 169-181; eISSN: 2656-8632

RESEARCH ARTICLE OPEN ACCESS

Manuscript received February 10, 2024; revised February 26, 2024; accepted April 10, 2024; date of publication April 20, 2024
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v6i2.388

How to cite: Angga Maulana Akbar, Rudy Herteno, Setyo Wahyu Saputro, Mohammad Reza Faisal, and Radityo Adi Nugroho, Enhancing Software Defect Prediction through Hybrid Optimization for Feature Selection and Gradient Boosting Classification, Journal of Electronics, Electromedical Engineering, and Medical Informatics, vol. 6, no. 2, pp. 169-181, April 2024.

Optimizing Software Defect Prediction Models: Integrating Hybrid Grey Wolf and Particle Swarm Optimization for Enhanced Feature Selection with Popular Gradient Boosting Algorithm

Angga Maulana Akbar, Rudy Herteno, Setyo Wahyu Saputro, Mohammad Reza Faisal, and Radityo Adi Nugroho

Department of Computer Science, Lambung Mangkurat University, Banjarbaru, South Kalimantan, Indonesia
Corresponding author: rudy.herteno@ulm.ac.id

ABSTRACT Software defects, also referred to as software bugs, are anomalies or flaws in computer program that cause software to behave unexpectedly or produce incorrect results. These defects can manifest in various forms, including coding errors, design flaws, and logic mistakes, this defect have the potential to emerge at any stage of the software development lifecycle. Traditional prediction models usually have lower prediction performance. To address this issue, this paper proposes a novel prediction model using Hybrid Grey Wolf Optimizer and Particle Swarm Optimization (HGWOPSO). This research aims to determine whether the Hybrid Grey Wolf and Particle Swarm Optimization model could potentially improve the effectiveness of software defect prediction compared to base PSO and GWO algorithms without hybridization. Furthermore, this study aims to determine the effectiveness of different Gradient Boosting Algorithm classification algorithms when combined with HGWOPSO feature selection in predicting software defects. The study utilizes 13 NASA MDP dataset. These dataset are divided into testing and training data using 10-fold cross-validation. After data is divided, SMOTE technique is employed in training data. This technique generates synthetic samples to balance the dataset, ensuring better performance of the predictive model. Subsequently feature selection is conducted using HGWOPSO Algorithm. Each subset of the NASA MDP dataset will be processed by three boosting classification algorithms namely XGBoost, LightGBM, and CatBoost. Performance evaluation is based on the Area under the ROC Curve (AUC) value. Average AUC values yielded by HGWOPSO XGBoost, HGWOPSO LightGBM, and HGWOPSO CatBoost are 0.891, 0.881, and 0.894, respectively. Results of this study indicated that utilizing the HGWOPSO algorithm improved AUC performance compared to the base GWO and PSO algorithms. Specifically, HGWOPSO CatBoost achieved the highest AUC of 0.894. This represents a 6.5% increase in AUC with a significance value of 0.00552 compared to PSO CatBoost, and a 6.3% AUC increase with a significance value of 0.00148 compared to GWO CatBoost. This study demonstrated that HGWOPSO significantly improves the performance of software defect prediction. The implication of this research is to enhance software defect prediction models by incorporating hybrid optimization techniques and combining them with gradient boosting algorithms, which can potentially identify and address defects more accurately.

INDEX TERMS Boosting Algorithm, HGWOPSO, Machine Learning, Software Defect Prediction

I. INTRODUCTION

A. BACKGROUND

Software defect prediction is a crucial tasks in software engineering that can be utilized to maintain software quality [1]. Software defect is a bug, error, flaw, mistake, fault, or failure in a computer system that can cause unexpected or erroneous results or impair intended software performance [2]. To enhance the reliability of software, developers utilize software defect prediction techniques to identify potential bugs and various error [3]. Software defect prediction seeks to forecast defective software modules before they are identified [4]. Identifying software defects at an early stage can result in decreased development expenses, rework efforts, and more reliable software [5]. Identify defective software modules is important to continuously improve the quality of software [6].

B. PREVIOUS STUDIES

Software defect prediction datasets often have noisy attribute properties, high dimensional, and imbalance classes. Specifically, in the NASA MDP dataset, several attributes exhibit a wide range of values, resulting in noisy attributes. Additionally, datasets such as JM1 and MC1 have very large dimensions, which can cause algorithms to consume significant time and resources. Moreover, high-dimensional data can lead algorithms to produce suboptimal results. Furthermore, the majority of the NASA MDP datasets exhibit an imbalanced class distribution between defects and non-defects [7,8]. To overcome problems of imbalanced classes in software defect dataset, Rahardian et al [9] conducted an experiment to solve the imbalance class problem in the Nasa MDP dataset, they took several approaches, namely using Synthetic Minority Oversampling Technique (SMOTE), Tomek Links (TL), One-Sided Selection (OSS), Random Oversampling (ROS), and Random Undersampling (RUS). The results show that the highest AUC value obtained is achieved by using the SMOTE approach, with an AUC value of 0.7277. This research demonstrates that SMOTE is an effective method for addressing imbalanced classes in the NASA MDP dataset. However, this study did not incorporate feature selection into the predictive models. Feature selection involves selecting attributes that have a significant impact on predicting the class. This technique can reduce the number of input features to a classifier and enhance prediction performance. Consequently, predicting software defects without feature selection may yield suboptimal results [10]. To address this issue, a feature selection method is employed to reduce the number of features and improve prediction performance.

Futhermore a study conducted by [11] employed an experiment to handle noisy attributes. They utilized two approaches using Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) for feature selection. The researchers conducted several experiments using different classifiers, namely Neural Network, Nearest Neighbor, Support Vector Machine (SVM), Statistical Classifier, and Decision Tree on the NASA MDP dataset. The results showed that significant values were obtained when using the SVM Classifier. The Average AUC result of PSO-SVM is 0.695, while the Average AUC of GA-SVM is 0.631. This research proved that PSO and GA are effective optimization algorithm for handling noisy attributes. However, in this study, data balancing methods were not utilized, the problem of imbalanced classes still exists. Consequently, this leads to poor performance produced by the algorithm.

Another research was conducted by [12]. In this study, they conducted several experiments to enhance GA performance by employing hyperparameter tuning and SMOTE in the NASA MDP dataset. They utilized several approaches, namely Grid search, Random search, Optuna, Bayesian search, Hyperband, Tree-structured Parzen Estimator (TPE), and Nevergrad. The highest average AUC obtained was 0.806 using Hyperband and 0.805 using Optuna. Another research utilizing PSO as feature selection was conducted by [13] and [14]. In the study conducted by [13], they employed RUS, PSO, and Naïve Bayes to predict software defects in the NASA MDP dataset, with the best AUC obtained being 0.801. Meanwhile, a study conducted by [14] attempted a different balancing method, namely using Bootstrap Aggregating (Bagging) to address the issue of class imbalance. In this research, they utilized PSO for feature selection and Logistic Regression as the classification algorithm. The highest AUC result they obtained was 0.794. The results of the three previous studies have shown that it is possible to address noisy attributes and imbalanced classes by implementing balancing methods and then utilizing PSO or GA as feature selection. However, PSO and GA also have weaknesses, especially in high-dimensional datasets. These algorithms tend to generate suboptimal solutions within the search space without achieving better solutions. As a result feature selection yield suboptimal performance in the model, consume valuable time, and getting traped in local optima [15,16].

Feature Selection, especially PSO tends to have low performance without optimization. Generally, the best results can be obtained when parameter tuning is performed or when various PSO techniques are utilized [15]. According to [17], there are several techniques to enhance the PSO method, including hybridization, improved strategies such as fuzzy logic and mutation, and the utilization of different PSO variants such as binary and chaotic. These techniques can improve the performance of the PSO algorithm. Furthermore research was conducted by [18], who attempted to enhance the PSO technique by using a variant of PSO. They employed Binary PSO as feature selection with Artificial Neural Network (ANN) as classification. This method was used to predict software defects in four NASA MDP datasets: JM1, KC1, KC3, and PC1. They generated AUC values of 0.739, 0.8487, 0.882, and 0.9297, respectively, achieving an average AUC value of 0.84985. However, in this research, premature convergence occurred, leading to PSO being trapped in local optima. This issue can result in PSO yielding suboptimal results. To address this issue, our study combines PSO with algorithms that have good exploration capabilities for hybridization to prevent PSO from getting trapped in local optima in the software defect prediction model.

Based on this background, we proposed a model to optimize the PSO algorithm by hybridizing with the GWO algorithm, as previously mentioned by [17], doing a hybrid on PSO allows this algorithm to get more optimal results. We used PSO over GA because particle swarm optimization algorithms are easier to use, require fewer adjustable parameters, and are simpler to comprehend compared to other bionic algorithms like genetic algorithms [15]. According to [19] the right classifier is needed to be able to reduce high dimensional data and to get better performance. Research conducted by [20] found that the Gradient Boosting Algorithm can handle High Dimensional Data. Therefore, we propose a new prediction model using HGWOPSO as feature selection and popular Gradient Boosting Algorithm as classification for predicting software defect in NASA MDP Dataset. Gradient Boosting used in this study are XGBoost, LightGBM, and CatBoost.

C. OBJECTIVE

The objective of this study is to improving performance results in software defect prediction using HGWOPSO as feature selection for XGBoost, LightGBM, and CatBoost as Classifier which measured with Area Under the ROC Curve (AUC).

II. METHOD

This section describes the dataset used, Synthetic Minority Oversampling Technique (SMOTE), Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), Hybrid Grey Wolf Optimizer and Particle Swarm Optimizaion (HGWOPSO), 10 Fold cross validation, Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), Categorial Boosting (CatBoost), Area under the ROC Curve (AUC) and T-Test. The research flow of this research can be seen in FIGURE 1.

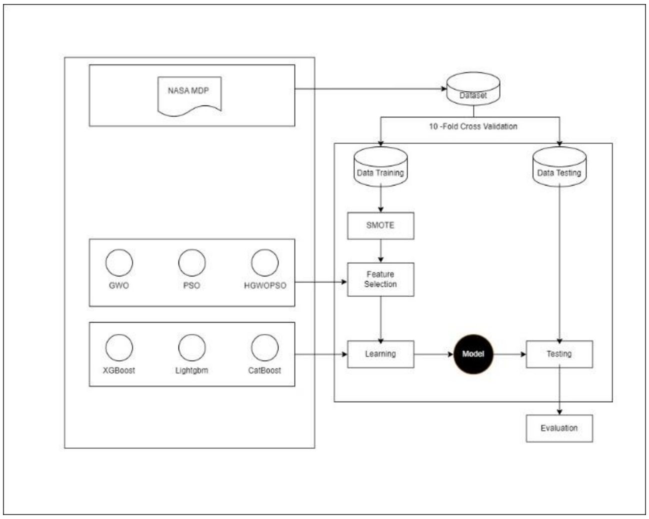

FIGURE 1. Research Flow using proposed Feature Selection and Classification Models

FIGURE 1 shows a flowchart that we used in this study. The first step is collecting the NASA MDP dataset, followed by dividing the data using cross validation. In this study we use 10-fold cross validation for the validation technique. Each NASA MDP dataset is divided into 10 sections, with 8 sections allocated for training data while the remaining 2 section are used as test data. After the data is divided, SMOTE is performed on the training data to balance the dataset, followed by feature selection and classification executed with three scenarios. Feature selection executed via PSO, GWO, and HGWOPSO. After the feature selection is executed, classification is performed using 3 different algorithms which are Xgboost, Lightgbm, and Catboost. Research evaluation uses the average AUC value. This Experiments was carried out using Jupyter Notebook.

A. DATA COLLECTION

In this study we use a software defect dataset called NASA MDP, These datasets are sourced from the NASA corpus, which encompasses real software projects across diverse domains and programming languages namely C, C++, and Java. The dataset exhibits considerable variations in code size, complexity, and functionality, offering a comprehensive representation of software development challenges. It comprises numerous software metrics, including lines of code, cyclomatic complexity, and code churn. These metrics provide valuable insights into the characteristics and attributes of software components. The primary purpose of this dataset is to facilitate the evaluation and development of predictive models aimed at identifying potentially defective software components early in the development lifecycle. In the data preprocessing phase, attributes containing categorical values are converted to nominal values, specifically 0 and 1. In the NASA MDP dataset, the Defective attribute represent Y and will converted to 1 while Non-Defective represent N and will be converted to 0. The dataset is available for download at the following link: https://github.com/klainfo/NASADefectDataset/tree/master
TABLE 1 is shows, which contains information and some general statistics about each of the datasets used.

B. 10 K-FOLD CROSS VALIDATION

To reduce the tendency or systematic error in estimating the performance of a model, random sampling in datasets is performed by implementing cross validation [21]. Cross-validation is a statistical method for evaluating the performance of an algorithm. The capability of cross-validation lies in its ability to divide the data into training and testing sets. Cross-validation is a computational method that requires information partitioning using subsets. [22]. Cross validation is also resampling data to prevent overfitting [23]. One part of the data is ultilazed to validate the model while the remaining part is utilized for training the classifier [24] At this phase, the dataset is divided into training and test data using cross-validation with a value of k = 10. The data will be split into ten subsets, each containing instances from the same class [25].

C. Synthetic Minority Oversampling Technique

SMOTE is a resampling technique that generates some samples in order to increase the number of the minority class by selecting a random point from the line segment. SMOTE linking a sample and its closest neighbor to generates a new sample [10]. The SMOTE method uses oversampling to rebalance the original training set. Instead of simply replicating minority class instances, the primary concept of SMOTE is to offer synthetic samples [26]. The idea using SMOTE in software defect prediction is to balance the defective and non-defective instances, which can increase the detection performance [27]. SMOTE can be mathematically modeled in the following equation (1) [28].

TABLE 1
Specification NASA MDP dataset

Dataset	Attribute	Instance	Defects	Non-Defects	Defects%	Non-Defects%	Programming language
CM1	38	327	42	285	12.8	87.2	C
JM1	22	7782	1672	6110	21.5	78.5	Java
KC1	22	1186	299	887	25.2	74.8	C++
KC3	40	194	36	158	18.6	81.4	Java
KC4	42	191	77	114	40.3	59.7	Java
MC1	39	1988	46	1942	2.3	97.7	C
MC2	40	125	44	81	35.2	64.8	Python
MW1	38	253	27	226	10.7	89.3	Java
PC1	38	705	61	644	8.7	91.3	C
PC2	37	745	16	729	2.1	97.9	Java
PC3	38	1077	134	943	12.4	87.6	Python
PC4	38	1287	177	1110	13.8	86.2	Python
PC5	39	1711	471	1240	27.5	72.5	Java

$$x_{new} = x + rand(0,1) \times (y[i] - x) \quad (1)$$

Consider a minority class sample *x* and one of its *k*-nearest neighbors *y*[i]. The equation generates a new synthetic sample x_{new} by linearly interpolating between *x* and *y*[i], with the extent of interpolation controlled by a random factor. The random factor, denoted by rand (0,1), scales the difference between *x* and *y*[i], allowing for variability in the synthetic sample generation process. By repeating this process for each sample in the minority class and selecting appropriate nearest neighbors, SMOTE effectively balancing the dataset, creating new synthetic samples that reflect the underlying distribution of the minority class. This method helps to rebalance the class distribution, enabling classifiers to learn more effectively from the data and improving their ability to generalize to minority class instances [28]. TABLE 2 shows before and after SMOTE.

TABLE 2
SMOTE process

Dataset	Before	After
CM1	327	570
JM1	7782	12220
KC1	1186	1736
KC3	194	316
KC4	191	194
MC1	1988	3884
MC2	125	162
MW1	253	452
PC1	705	1288
PC2	745	1458
PC3	1077	1886
PC4	1287	2220
PC5	1711	2480

D. FEATURE SELECTION

1. PSO FEATURE SELECTION

Particle swarm optimization (PSO) is a remarkably effective metaheuristic approach that has been effeciently employed to acquire an optimal subset of features containing crucial information within a feasible time [29]. PSO begins by generating a set of random solutions and iteratively seeks for the optimal solution [15]. The PSO algorithm's concept and development were inspired by the social behaviors of fish schools and flocks of birds. In the wild, a swarm of birds flies across an area, following the leader who has closest position to the food. Birds social behavior can be translated into mathematical procedures, such as PSO, to solve optimization issues. In this approach, the swarm of birds is viewed as a swarm of particles, with each particle representing a candidate solution. [30]. A swarm of particles updates their relative positions from iteration to effectively conduct the search process. In order to obtain the optimum solution, each particle moves towards its prior personal best position (Pbest) and the global best position (Gbest) inside the swarm [17]. In order to produce the optimal feature subset, PSO will ends when the requirements are satisfied. PSO position and velocity variations are derived from basic formulas (2) and (3) [31].

$$\mathrm{x}_i^{(t+1)} = \mathrm{x}_i^t + \mathrm{v}_i^{(t+1)} \quad (2)$$

$$\mathrm{v}_i^{(t+1)} = \mathrm{v}_i^t + \mathrm{c}_1\mathrm{r}_1(\mathrm{Pbest}_i^t - \mathrm{x}_i^t) + \mathrm{c}_2\mathrm{r}_2(\mathrm{Gbest}^t - \mathrm{x}_i^t) \quad (3)$$

The first formula illustrates how the position (x_i) of a particle (i) at time step $(t+1)$ is updated from its previous position at time (t), taking into account the particle's velocity (v_i). Here $\mathrm{x}_i^{(t+1)}$ represents the updated position of particle. On the other hand, the second formula explains how the velocity of the particle at time step is updated by considering the contributions from the personal best position (Pbest) and the global best position (Gbest) that the particle itself and the entire population have achieved respectively [31]. The PSO algorithm's performance is optimized for optimal problem solving by the adjustment of coefficients (c1 and c2) and randomization (r1 and r2) [32]. In this studies we used Cognitive Coefficient (c1) = 0.5, Social Coefficient (c2) = 0.3, Inertia weight (w) = 0.9, iteration = 50 and population size = 5. TABLE 3 shown the outcome of PSO feature selection.

TABLE 3
Feature selection with PSO

Dataset	Features	Feature Selected
CM1	37	19
JM1	21	12
KC1	21	10
KC3	39	20
KC4	41	22
MC1	38	18
MC2	39	19
MW1	37	18
PC1	37	17
PC2	36	18
PC3	37	18
PC4	37	20
PC5	38	19

2. GWO FEATURE SELECTION

Grey Wolf Optimizer is metaheuristic swarm-based algorithm that mimics the social leadership and hunting behavior of grey wolves in nature [33]. The algorithm mimics how grey wolves behave in their natural environment, including their leadership structure and pursuit style [34]. Within the leadership structure of grey wolves, there exist four distinct type: alpha, beta, delta, and omega wolves. Alpha wolves symbolize the solution with the most optimal results, while beta and delta wolves denote the second and third best solutions within the population, the rest of nominated solutions are omega [35]. Hunting behavior of grey wolves consists of the following three primary parts. First part is tracking, chasing, and approaching the prey. After that the wolfs Pursuing, encircling, and harassing the prey till it stops moving. Last part is the wolves attacking the prey [36]. Grey wolf algorithm can be mathematically modeled in the following equations (4) and (5) [33]:

$$D = |\, C \times X_p\,(t) - X(t)| \tag{4}$$

$$X(t+1) = X_p(t) - A \times D \tag{5}$$

In these equations the variable t represents the number of iterations, Xp denotes the prey position, X represent the grey wolves location, while The variables A and C serve as coefficients for the vectors. their values are determined through equations (6) and (7) [36]:

$$A = a \times (2 \times r_1 - 1) \tag{6}$$

$$C = 2 \times r_2 \tag{7}$$

Here, the quantity of a exhibits a linear decrease from 2 to 0, inversely correlating with the decreasing number of iterations. r1 and r2 represent uniformly selected random numbers between [0,1].

Alpha wolves lead grey wolves to locate prey. Occasionally, beta and delta wolves assist the alpha wolf. These algorithm prioritizes alpha wolves as the optimal option, followed by beta and delta wolves. As a result, the positions of these three wolves influence the movement of the rest of the population [35].

The mathematical formulas are shown in equation (8) [35]:

$$\begin{aligned} D_\alpha &= |C_1 \times X_\alpha - X(t)|, \\ D_\beta &= |C_3 \times X_\beta - X(t)|, \\ D_\delta &= |C_3 \times X_{\alpha\delta} - X(t)|. \end{aligned} \tag{8}$$

The values X_α, X_β and X_δ represent the best three wolves in each iteration, respectively as shown in equations (9) and (10) [36].

$$\begin{aligned} X_1 &= |X_\alpha - a_1 D_\alpha|, \\ X_2 &= |X_{\alpha\beta} - a_2 D_\beta|, \\ X_3 &= |X_\delta - a_2 D_\delta|, \end{aligned} \tag{9}$$

$$X_p\,(t+1) = \frac{X_1+X_2+X_3}{3} \tag{10}$$

Here, X_p(t + 1) representing the new position of the prey, which signifies the average of the positions of the top three wolves within the group. This algorithm will finish the hunt if Grey wolves attacking the prey [36]. In this study we utilized step size (a) = 2, Alfa (A) = 0.5, Convergence Control (C) = 0.3, population size = 5, and iteration = 50. TABLE 4 shows average feature selected by GWO.

TABLE 4
FEATURE SELECTION WITH GWO

Dataset	Features	Feature Selected
CM1	37	19
JM1	21	11
KC1	21	8
KC3	39	16
KC4	41	19
MC1	38	16
MC2	39	18
MW1	37	17
PC1	37	17
PC2	36	15
PC3	37	16
PC4	37	18
PC5	38	17

3. HGWOPSO FEATURE SELECTION

Hybrid Grey Wolf Optimizer - Particle Swarm Optimization is developed without altering the fundamental operation of GWO and PSO. The PSO algorithm can successfully solve most real-world issues [17]. However, a solution is needed to prevent PSO from becoming stuck in a local minimum. The GWO algorithm is used to assist the PSO in minimizing the risk of getting trapped in a local minimum. Rather than sending certain particles to random locations, the exploration ability of the GWO can be used to partially improve some of the particle positions, which decreases the risks entailed.

Journal of Electronics, Electromedical Engineering, and Medical Informatics
Multidisciplinary: Rapid Review: Open Access Journal Vol. 6, No. 2, April 2024, pp: 169-181; eISSN: 2656-8632

Because the GWO algorithm is used in addition to the PSO algorithm, the running duration of the code is increased [37], [38]. FIGURE 2 show the flowchart of HGWOPSO method.

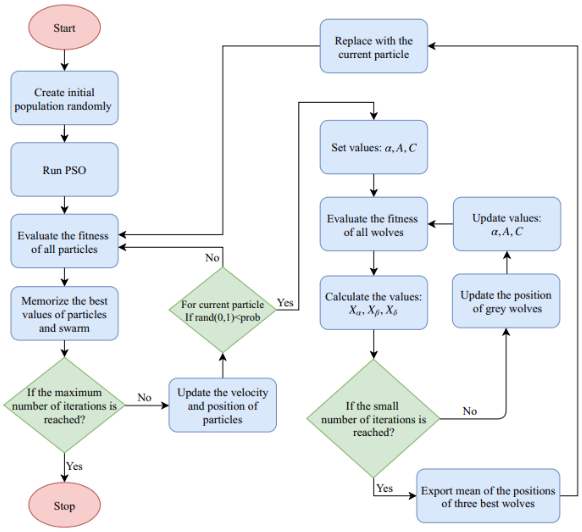

FIGURE 2. Flowchart of HGWOPSO Feature Selection[32]

In this study, we used the same parameters for both PSO and GWO algorithm, TABLE 5 shows average feature selected by HGWOPSO.

TABLE 5
FEATURE SELECTION WITH HGWOPSO

Dataset	Features	Feature Selected
CM1	37	23
JM1	21	18
KC1	21	12
KC3	39	18
KC4	41	17
MC1	38	13
MC2	39	21
MW1	37	16
PC1	37	21
PC2	36	18
PC3	37	20
PC4	37	23
PC5	38	25

E. CLASSIFICATION

1. XGBOOST CLASSIFICATION

Extreme Gradient Boosting is a supervised machine learning technique that combines the predictions of multiple weaker or low-performing models. This approach involves utilizing an ensemble of decision trees within the gradient boosting framework [39]. XGBoost utilizes gradient boosting as its core. However, unlike the traditional gradient boosting algorithm, XGBoost does not add weak learners sequentially. Instead, XGBoost adopts a multi-threaded approach by optimizing CPU core utilization in machines [40]. XGBoost is known for its speed and efficiency due to its implementation of parallel processing [41]. The Xgboost approach utilizes the shrinkage technique to combine multiple weak learners and reduce the possibility of model overfitting. The combination of trees can be mathematically modeled in equation (11) [42].

$$F_m(X) = F_{m-1}(X) + nf_m(X), 0 < n < 1 \tag{11}$$

Where, *fm*(*X*) denotes the m-th step in constructing the weak learner, and *Fm*(*X*) represents the m-th step in building the integrated learner. As there exists a substantial negative relationship between the parameter *n* and the number of iterations, the model's generalization properties are frequently improved when *n* assumes a lesser value [43]. $f_t(x_i)$ represents the newly constructed tree model, with t indicating the total count of base tree models. The computational process of XGBoost is shown in a schematic diagram illustrated in FIGURE 3.

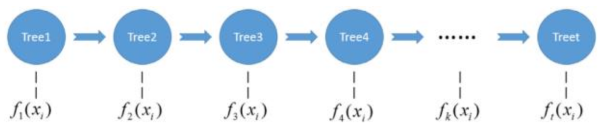

FIGURE 3. A schematic diagram of XGBoost algorithm [44]

2. LIGHTGBM CLASSIFICATION

Light Gradient Boosting Machine is a gradient boosting framework that uses tree-based learning algorithms. LightGBM is mainly featured by the decision tree algorithm based on gradient-based one-side sampling (GOSS), exclusive feature bundling (EFB), a histogram and leaf-wise growth strategy with a depth limit [45]. GOSS removes a considerable fraction of data instances with small gradients and only utilizes the remainder to estimate information gain. Because data records with bigger gradients play an important part in the computation of information gain, GOSS can produce a reasonably accurate estimate of information gain with a considerably smaller dataset. EFB reduces the amount of features by bundling mutually exclusive characteristics [46]. One unique aspect of the LightGBM algorithm compared to other gradient boosting tree algorithms is in spilting tree. When another boosting algorithms split the tree depthwise or levelwise, LightGBM growing the tree leafwise on the same leaf [47]. FIGURE 4 shows how LighGBM spliting the tree while FIGURE 5 shows how another algorithm such as XGBoost splitting the tree.

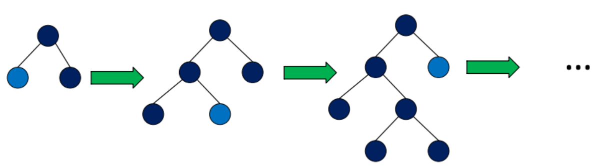

FIGURE 4. Leaf-wise tree growth in LightGBM [47]

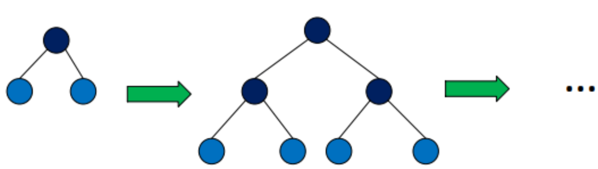

FIGURE 5. Level-wise tree growth in XGBoost [47]

LightGBM can be mathematically modeled in the following equation (12) [45]

$$\boldsymbol{y_i} = \sum_k^K \boldsymbol{f_k}(\boldsymbol{x_i}) \tag{12}$$

Here, $\boldsymbol{y_i}$ denotes the prediction generated by the model for the $\boldsymbol{i}$-th data sample. This prediction stems from the combination of predictions from each decision tree $\boldsymbol{f_k}$, where $\boldsymbol{k}$ represents the number of trees within the model. Consequently, if there are $\boldsymbol{K}$ trees in the model, the final prediction is the summation of predictions yielded by each individual tree. This illustrates the concept of ensemble learning, wherein the combination of multiple weak models can yield a stronger one. By employing this approach, LightGBM enables the modeling of complex relationships between input features and target outputs by integrating the results from several decision trees [45-47].

3. CATBOOST CLASSIFICATION

Categorical Boosting is a new gradient boosting tree that can hadle categorical data. It does not use binary substitution of categorical values, instead it performs a random permutation of the dataset and calculates the average label value [48]. Catboost use decision tree as base predictor [49]. When constructing a new split for the tree, CatBoost uses a greedy way to consider the combinations. CatBoost combines all combinations preset with all categorical features in the dataset [50]. FIGURE 6 shows how CatBoost constructing a tree.

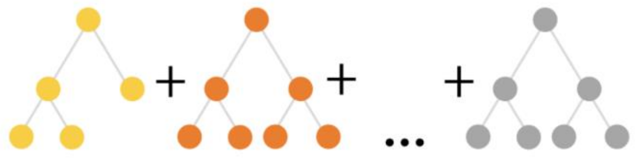

FIGURE 6. **Depth-wise tree growth in CatBoost [50]**

Due to CatBoost unique way of building trees, CatBoost has two main components in performing optimization, namely Loss Component, and Regularization component [49]. Loss component is the part that measures how well the model predicts the actual target from the training samples, Loss Component can be modeled into mathematical form in the following equation (13) [49]

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(y_i, F(x_i)) \tag{13}$$

Here θ is parameter model, N signifies the total number of samples within the dataset, representing the extent of the training data utilized to construct the CatBoost model. $l(y_i, F(x_i))$ represents the loss function, which quantifies the discrepancy between the true target value y_i and the predicted value $F(x_i)$ for the i-th sample. After the Loss component results are obtained, the results of the loss component are summed with the regularization component. Regularization Component can be modeled into mathematical form in the following equation (14) [49]

$$\Omega(\theta) = \gamma \sum_{j=1}^{M} \frac{\theta_j^2}{2} \tag{14}$$

The values M represents the total number of parameters in the model, and γ is a hyperparameter controlling the regularization strength. The regularization component aims to curb the weight of parameters, preventing them from growing excessively large, which could lead to overfitting. Meanwhile, j serves as an index used to iterate through each parameter in the model [49].

F. AREA UNDER THE ROC CURVE

The area under the Receiver Operating Characteristics curve, or simply AUC is a metric used to measure the performance of classification models. It represents the measure of separability between the models true positive rate and false positive rate across various threshold values. AUC ranges from 0 to 1, where a higher AUC indicates better model performance [51]. AUC includes False Negative (FN), False Positive (FP), True Negative (TN), and True Positive (TP). AUC can be mathematically modeled in the following equations (15) [52]

$$AUC = \frac{\left(\frac{TP}{TP+FN}\right)\times\left(\frac{TN}{TN+FP}\right)}{2} \tag{15}$$

Moreover, interpreting the AUC value provides insights into the models capacity to differentiate between positive and negative classes. Additionally, AUC serves as a useful tool for model selection and comparison, allowing practitioners to assess the relative effectiveness of different classifiers [53]. TABLE 6 presents a list of several AUC values for categorization [54].

TABLE 6
Category of classification result based on AUC values

AUC Values	Category
0.90 – 1.00	Excellent
0.80 – 0.90	Good
0.70 – 0.80	Fair
0.60 – 0.70	Poor
0.50 – 0.60	Failure

G. T-TEST

The t-test is a statistical test employed to determine if there is a significant difference between the means of two groups. It is commonly employed in scientific research to assess whether the means of two populations are statistically different from each other [55]. The t-test calculates the t-value, which signifies the difference between the means of the two groups relative to the variation within each group, factoring in sample sizes and standard deviations. Subsequently, this t-value is compared against a critical value derived from the t-distribution to determine the statistical significance of the observed difference [56] If the t-test value is less than 0.05, then the results of both comparisons can be considered significant [57]. T-test can be calculated uses equations (16) below [56].

$$T = \frac{y_1 - y_2}{\sqrt{s_p^2\left(\frac{1}{n_1}+\frac{1}{n_{q2}}\right)}} \tag{16}$$

TABLE 7
AUC results in NASA MDP dataset

Dataset	Method								
	PSO XGB	PSO LGBM	PSO CAT	GWO XGB	GWO LGBM	GWO CAT	HGWOPSO XGB	HGWOPSO LGBM	HGWOPSO CAT
CM1	0.816736	0.8289104	0.731449	0.8315148	0.86271552	0.782981	0.878109606	0.845849754	**0.896231527**
JM1	0.685194	0.6857372	0.670386	0.6730202	0.69608008	0.67958	**0.717293548**	0.700623141	0.68144923
KC1	0.782656	0.7839955	0.660456	0.7923177	0.81152083	0.857281	0.824322917	0.808244048	**0.865983796**
KC3	0.836911	0.8116299	0.823056	0.8860417	0.85041667	0.856639	**0.943489583**	0.858854167	0.93125
KC4	0.878191	0.8537245	0.827231	0.8460952	0.82255556	0.857222	0.927469136	0.920634921	**0.94047619**
MC1	0.907869	0.9387123	0.937868	0.8231137	0.84708664	0.863512	0.964251916	**0.966330459**	0.934996696
MC2	0.889444	0.8296714	0.843356	0.8946875	0.89569444	0.761694	**0.942142857**	0.919374999	0.929375
MW1	0.808141	0.8272518	0.911699	0.8712121	0.86238472	0.923913	0.885902503	0.908285756	**0.951119895**
PC1	0.913314	0.8930403	0.906566	0.8827094	0.87596448	0.866934	0.930438416	**0.937964744**	0.934698375
PC2	0.900036	0.9228108	0.871235	0.8187508	0.7929395	0.819145	0.930793379	0.949570861	**0.973287671**
PC3	0.840378	0.8385834	0.839678	0.8106041	0.8238258	0.817827	0.864275316	**0.870578184**	0.858522144
PC4	0.944723	0.9393189	0.948541	0.9291876	0.93476135	0.934573	**0.958888300**	0.95693046	0.953285639
PC5	0.799486	0.8022568	0.807941	0.7786452	0.78559151	0.787343	**0.822342570**	0.819245453	0.782796056

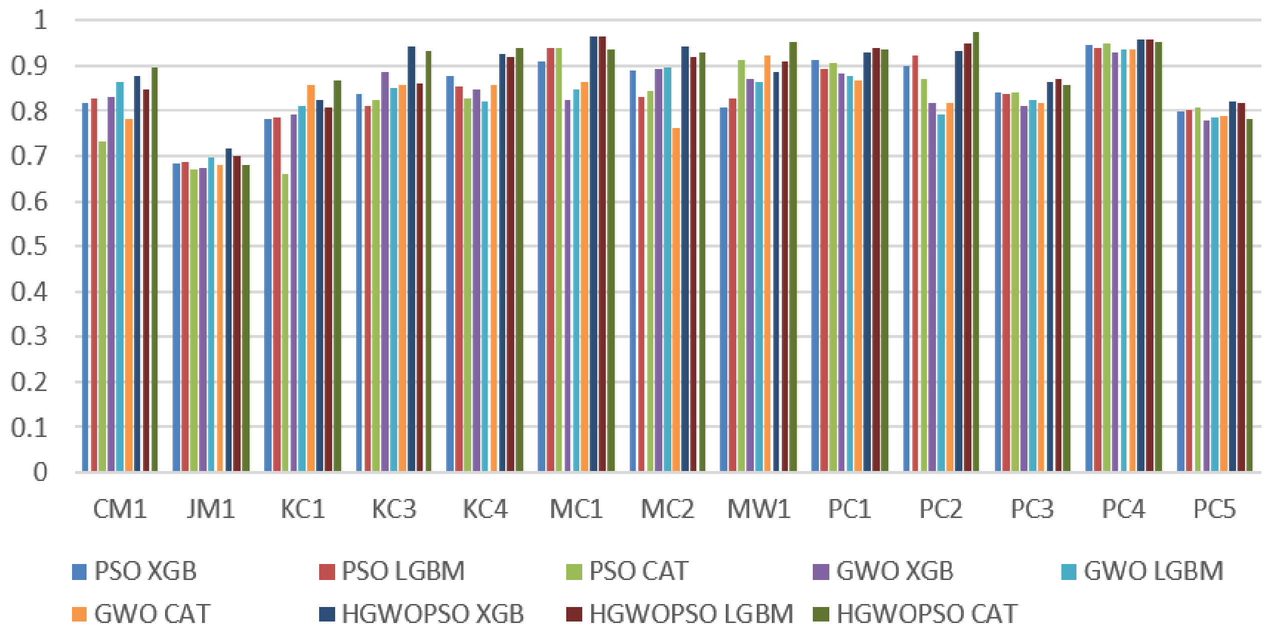

Figure 7. **AUC RESULTS IN NASA MDP DATASET**

Here y_1 and y_2 are the mean values from groups 1 and 2, s_p is an estimate of the pooled s of the measurements, and n1 and n2 are the number of observations for each group [56].

III. RESULT

TABLE 7 and FIGURE 7 shows the performance of each model on NASA MDP dataset. In this research, we observed that our proposed method of hybridizing the PSO algorithm with the GWO algorithm maximizes the results of the PSO algorithm. TABLE 7 and FIGURE 7 show that The HGWOPSO feature selection outperforms both the PSO and GWO algorithms across all 13 NASA MDP datasets. The average results for these three feature selection methods are presented in TABLE 8, While TABLE 9 shows increase value of each methods.

TABLE 8
AVERAGE AUC OF ALL METHOD

Method	Average AUC
PSO - XGBoost	0.846391
PSO - LightGBM	0.8427418
PSO - CatBoost	0.829189
GWO - XGBoost	0.8336846
GWO - LightGBM	0.83550285

GWO - CatBoost	0.831434
HGWOPSO - XGBoost	0.891516927
HGWOPSO - LightGBM	0.881729765
HGWOPSO - CatBoost	0.894882478

TABLE 9
AVERAGE INCREASE AUC VALUE OF ALL METHOD

Method Comparison	Increase Value
HGWPSO XG – PSO XG	0.04512
HGWOPSO LGBM – PSO LGBM	0.03898
HGWOPSO CAT – PSO CAT	0.06569
HGWOPSO XG – GWO XG	0.05783
HGWOPSO LGBM – GWO LGBM	0.04622
HGWOPSO CAT –GWO CAT	0.06344

After the average AUC results were obtained, we conducted a significance test using T-test to see if our proposed method was significant to the model before hybridization. T-test result can be seen in TABLE 10.

TABLE 10
T-TEST RESULT FOR EVERY METHOD

Method Comparison	T-test Value ($\alpha = 0.05$)	Significance
HGWPSO XG – PSO XG	0.00004	Significant
HGWOPSO LGBM – PSO LGBM	0.00013	Significant
HGWOPSO CAT – PSO CAT	0.00552	Significant
HGWOPSO XG – GWO XG	0.00006	Significant
HGWOPSO LGBM – GWO LGBM	0.00678	Significant
HGWOPSO CAT – GWO CAT	0.00148	Significant

Here in the TABLE 8, TABLE 9, and TABLE 10, is evident that there is a significant improvement between the HGWOPSO algorithm and the GWO or PSO algorithms. The results indicate that the highest outcome is achieved by HGWOPSO CatBoost with an Average AUC of 0.894. This represents an increase of 6.5% compared to PSO CatBoost, with a significance value of 0.005, and an increase of 6.3% compared to GWO CatBoost, with a significance value of 0.001. This test proved that our proposed method stands out by demonstrating a consistently higher level of significance compared to traditional PSO or GWO algorithms that do not utilize hybridization.

IV. DISCUSSION

The results showed that our proposed method could enchane software defect prediction using HGWOPSO as feature selection and gradient boosted tree as classifier such as XGBoost, LightGBM and CatBoost. As we can see in TABLE 10, We conducted a two-tailed t-test between HGWOPSO and PSO, and GWO individually. The results of all t-tests showed values smaller than 0.05. This means there is a significant difference between HGWPSO and PSO, as well as between HGWOPSO and GWO.

From the result above, our method has proven successfully in optimizing software defect prediction. This is evidenced that our method is superior compared to prior study, TABLE 11 shown the comparasion between our proposed method and other PSO method.

TABLE 11
COMPARASION OF AUC RESULT WITH PREVIOUS PSO STUDIES

Researcher	Method	AUC
[11]	PSO -SVM	0.695
[13]	PSO -NB	0.805
[14]	PSO -LR	0.794
[18]	BPSO(BCO) -ANN	0.849
	HGWOPSO - XGB	0.891
Our Research	HGWOPSO –LGBM	0.881
	HGWOPSO - CAT	0.894

With the significance of the results we obtained, compared to previous PSO research in NASA MDP dataset, where the highest AUC result is 0.849 using binary cross-entropy PSO and ANN, we obtained a higher result of 0.894, representing an increase value of 0.045. This demonstrates that our PSO model outperforms previous research. The increase in AUC from the previous result indicates that the optimization we conducted on the PSO algorithm successfully generated a superior model for software defect prediction.

In previous research on software defect prediction, especially in the NASA MDP dataset, various models were employed. Researchers employ different approaches to achieve optimal results, such as parameter tuning, combining multiple learning models, and seeking effective combinations between different methods. Because of that, we also strive to compare our research findings with different methodologies. TABLE 12 shown the comparasion between our proposed method and various methodologies.

TABLE 12
Comparasion of AUC result with other research method

Researcher	Method	AUC
[58]	FGA -NB	0.856
	BGA -LR	0.866
[59]	FLDA -MLP	0.866
[60]	MLP-MFFS ROS	0.817
[61]	FFeSSTri	0.834
	HGWOPSO - XGB	0.891
Our Research	HGWOPSO –LGBM	0.881
	HGWOPSO - CAT	0.894

TABLE 13
Detail comparison with other research method

Researcher	Method	Dataset			
		JM1	KC1	KC3	PC1
[18]	BPSO-ANN	**0.739**	0.848	0.882	0.929
[58]	BGA-LR	0.719	0.823	0.86	0.886
Proposed Research	HGWOPSO-CAT	0.681	**0.865**	**0.931**	**0.934**

TABLE 12 present a comprehensive analys compared to various methodologies. Compared to previous study where the highest AUC result is 0.866, We achieved a higher result using HGWOPSO CatBoost with a percentage increase of 0.028. It is clear that the result of this research outperform the methodology of previous studies. TABLE 13 and FIGURE 8 compares the performance of previous studies where the highest AUC was achieved, using the BPSO-ANN and BGA-LR methods. The study was conducted on the JM1, KC1, KC3, and PC1 datasets.

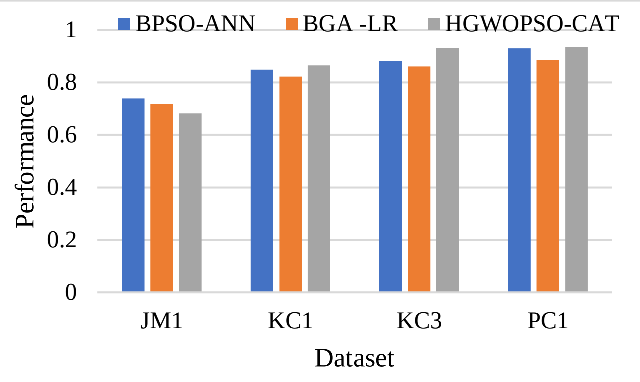

FIGURE 8. Detail comparison with other research method

Based on the data presented in TABLE 13 and FIGURE 8, a comparison is made between different methods for predicting software defects. The proposed method using HGWOPSO and CatBoost demonstrates the best performance in KC1, KC3, and PC1 datasets. HGWOPSO CatBoost achieves superior results compared to other methods because HGWOPSO optimizes the performance of PSO through the exploration capabilities inherited from the GWO algorithm. This enables HGWOPSO to select more relevant features and attain better results. Additionally, CatBoost's unique approach to constructing and splitting trees also plays a crucial role in classification. However, the method used in this study also has limitations. Specifically, the resulting model's performance fails to reach optimal levels in the JM1 dataset. HGWOPSO CatBoost yields an AUC of 0.681, as indicated in TABLE 6, which falls into the Poor Category. This is attributed to the excessively high-dimensional data and class imbalance present in the JM1 dataset, resulting in suboptimal results from the method we employed

In this study, our findings in software defect prediction using HGWOPSO have significant implications both in industry and research. Industrially, the prediction model we developed can be implemented in software development companies to enhance the quality assurance process. By accurately predicting software defects, companies can allocate resources more efficiently, prioritize testing efforts, and ultimately deliver high-quality software products to their clients. Additionally, IT consultancy firms can leverage our prediction model to offer better risk assessment and mitigation strategies to their clients, helping businesses anticipate potential software defects and take proactive measures to minimize their impact on operations. On the research front, our contribution in developing the HGWOPSO approach as a novel method for defect prediction provides a substantial contribution to the field of software engineering. Our findings can serve as a foundation for future research in building more advanced defect prediction models and improved methodologies. Furthermore, the dataset and methodology we utilized can serve as a benchmark for future studies in software defect prediction, facilitating the evaluation and enhancement of prediction models in the field. Thus, our research not only advances knowledge in software defect prediction but also has practical implications for various industries and research domains.

V. CONCLUSION

Software defect prediction is a crucial task in software engineering that can be utilized to maintain software quality. Identifying software defects at an early stage can result in decreased development expenses, rework efforts, and more reliable software. Software defect prediction datasets, specifically the NASA MDP dataset, have noisy attribute properties, high dimensionality, and imbalanced classes. To overcome these issues, we propose a method using HGWOPSO as feature selection and gradient boosting trees for classification, namely XGBoost, LightGBM, and CatBoost. The proposed method, which utilizes HGWOPSO, has been found to enhance AUC performance compared to the previous PSO study. The average AUC values yielded by HGWOPSO XGBoost, HGWOPSO LightGBM, and HGWOPSO CatBoost are 0.891, 0.881, and 0.894, respectively. We also conducted a two-tailed t-test between HGWOPSO and PSO, as well as between HGWOPSO and GWO individually. The results of all t-tests showed values smaller than 0.05. This indicates a significant difference between HGWPSO and PSO, as well as between HGWOPSO and GWO. This is prove that our proposed method successfully maximizes the results of the PSO algorithm. The findings of the research shows that employing HGWOPSO feature selection with CatBoost classification results in superior performance compared to the method used in the previous study.

This research still has several limitations. As we can see in TABLE 13, it is evident that the method we used yielded suboptimal performance compared to previous studies, spesifically in the JM1 dataset. Our best method, HGWOPSO CatBoost, resulted in an AUC of 0.681, falling into the 'poor' category. This could be attributed to the dataset's excessively large high-dimensional data and highly imbalanced classes. For future research, we recommend focusing on examining this dataset, given its excessively high-dimensional data and highly imbalanced classes. To mitigate the imbalanced classes, we suggest changing the sampling method used, such as RUS, ROS, TL, or OSS, This change aims to address class imbalance and improve model performance. Additionally, we recommend changing the classification method in order to select a more suitable approach. The objective is to address issues associated with high-dimensional data, This is evident when we change the classification yields beeter performance result, as shown in TABLE 7 and FIGURE 7, where the HGWOPSO - XGBoost method outperformed HGWOPSO - CatBoost with an AUC of 0.717. Furthermore, we suggest employing hyperparameter tuning in future research the aim for this study is to achieve more optimal results in software defect prediction.

REFERENCES

[1] M. K. Thota, F. H. Shajin, dan P. Rajesh, "Survey on software defect prediction techniques," International Journal of Applied Science and Engineering, vol. 17, no. 4, pp. 331-344, 2020, doi: 10.6703/IJASE.202012_17(4).331.

[2] J. Li, P. He, J. Zhu and M. R. Lyu, "Software Defect Prediction via Convolutional Neural Network," 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, Czech Republic, pp. 318-328, 2017. doi: 10.1109/QRS.2017.42.

[3] M. S. Rawat and S. K. Dubey, "Software defect prediction models for quality improvement: A literature study," Int. J. Comput. Sci. Issues, vol. 9, no. 5 5–2, pp. 288–296, 2022. [Online], Available: https://www.researchgate.net/publication/287793526_Software_Defect_Prediction_Models_for_Quality_Improvement_A_Literature_Study

[4] Z. Li, X. Y. Jing, X. Zhu, "Progress on approaches to software defect prediction," IET Software, vol. 12, no. 3, pp. 161-175, 2018, doi: 10.1049/iet-sen.2017.0148.

[5] J. Ren, K. Qin, Y. Ma, and G. Luo, "Survey on Software Defect Prediction Using Machine Learning Techniques," J. Appl. Math.,vol. 3, no. 12, pp. 2319–7064, 2018, doi: 10.1155/2014/785435.

[6] G. Czibula, Z. Marian, I. G. Czibula, "Software defect prediction using relational association rule mining," Information Sciences, vol. 264, pp. 260-278, 2014, doi: 10.1016/j.ins.2013.12.031.

[7] B. Khan, R. Naseem, M. A. Shah, K. Wakil, A. Khan, M. I. Uddin, M. Mahmoud, "Software defect prediction for healthcare big data: an empirical evaluation of machine learning techniques," Journal of Healthcare Engineering, 2021, doi: 10.1155/2021/8899263.

[8] K. K. Bejjanki, J. Gyani, and N. Gugulothu, "Class imbalance reduction (CIR): A novel approach to software defect prediction in the presence of class imbalance," *Symmetry (Basel).*, vol. 12, no. 3, 2020, doi: 10.3390/sym12030407.

[9] H. Rahardian, M. R. Faisal, F. Abadi, R. A. Nugroho, R. Herteno, "Implementation of Data Level Approach Techniques to Solve Unbalanced Data Case on Software Defect Classification," Journal of Data Science and Software Engineering, vol. 1, no. 01, pp. 53-62, 2020, doi: 10.20527/jdsse.v1i01.13.

[10] K. Khadijah, P. S. Sasongko, "Software Defect Prediction Using Synthetic Minority Over-sampling Technique and Extreme Learning Machine," *Journal of Telematics and Informatics (JTI)*, vol. 7, no. 2, pp. 60-68, 2019. doi: 10.12928/jti.v7i2.

[11] R. S. Wahono, N. Suryana, S. Ahmad, "Metaheuristic optimization based feature selection for software defect prediction," Journal of Software, vol. 9, no. 5, pp. 1324-1333, 2014, doi:10.4304/jsw.9.5.1324-1333.

[12] M. K. Suryadi, K, H. Rudy, S. W. Saputro, M. R. Faisal, and R. A. Nugroho, "A Comparative Study of Various Hyperparameter Tuning on Random Forest Classification With SMOTE and Feature Selection Using Genetic Algorithm in Software Defect Prediction," Journal of Electronics, Electromedical Engineering, and Medical Informatics, vol. 6, no. 2, 2024. doi: 10.35882/jeeemi.v6i2.375.

[13] A. Suryadi, "Integration of feature selection with data level approach for software defect prediction," *Sinkron: Jurnal dan Penelitian Teknik Informatika*, vol. 4, no. 1, pp. 51-57, 2019. doi: 10.33395/sinkron.v3i1.10137

[14] R. S. Wahono and N. Suryana, "Combining particle swarm optimization based feature selection and bagging technique for software defect prediction," International Journal of Software Engineering and Its Applications, vol. 7, no. 5, pp. 153-166, 2013. doi: 10.14257/ijseia.2013.7.5.16

[15] M. Cai, "An Improved Particle Swarm Optimization Algorithm and Its Application to the Extreme Value Optimization Problem of Multivariable Function," *Comput. Intell. Neurosci.*, vol. 2022, 2022, doi: 10.1155/2022/1935272.

[16] F. Catak and T. Bilgem, "Genetic algorithm based feature selection in high dimensional text dataset classification," *WSEAS Transactions On Information Science And Applications*, vol. 12, no. 28, pp. 290-296, 2015, [Online], Available: https://www.researchgate.net/publication/283661718_Genetic_Algorithm_based_Feature_Selection_in_High_Dimensional_Text_Dataset_Classification

[17] A. G. Gad, *Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review*, vol. 29, no. 5. pp. 2531-2561, 2022. doi: 10.1007/s11831-021-09694-4.

[18] R. Malhotra, A. Shakya, R. Ranjan, and R. Banshi, "Software defect prediction using Binary Particle Swarm Optimization with Binary Cross Entropy as the fitness function," *Journal of Physics: Conference Series*, vol. 1767, no. 1, p. 012003, February 2021. doi: 10.1088/1742-6596/1767/1/012003

[19] V. Pappu, P. M. Pardalos, "High-dimensional data classification," Clusters, Orders, and Trees: Methods and Applications: In Honor of Boris Mirkin's 70th Birthday, pp. 119-150, 2014, doi: 10.1007/978-1-4939-0742-7_8.

[20] R. Blagus, L. Lusa, "Boosting for high-dimensional two-class prediction," BMC Bioinformatics, vol. 16, pp. 1-17, 2015, doi: 10.1186/s12859-015-0868-7.

[21] S. Ghosh, A. Rana, and V. Kansal, "A Nonlinear Manifold Detection based Model for Software Defect Prediction," *Procedia Comput. Sci.*, vol. 132, pp. 581–594, 2018, doi: 10.1016/j.procs.2018.05.012.

[22] R. T. Yunardi, R. Apsari, and M. Yasin, "Comparison of Machine Learning Algorithm For Urine Glucose Level Classification Using Side-Polished Fiber Sensor," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 2, no. 2, pp. 33–39, 2020, doi: 10.35882/jeeemi.v2i2.1.

[23] D. Berrar, "Cross-validation," *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, vol. 1–3, no. January 2018, pp. 542–545, 2018, doi: 10.1016/B978-0-12-809633-8.20349-X.

[24] M. Anbu and G. S. Anandha Mala, "Feature selection using firefly algorithm in software defect prediction," *Cluster Computing*, vol. 22, no. 4, pp. 10925–10934, 2019, doi: 10.1007/s10586-017-1235-3.

[25] M. M. Mafazy, "Classification of COVID-19 Cough Sounds using Mel Frequency Cepstral Coefficient (MFCC) Feature Extraction and Support Vector Machine Telematika Classification of COVID-19 Cough Sounds using Mel Frequency Cepstral Coefficient (MFCC) Feature Extraction," no. August, 2023, doi: 10.35671/telematika.v16i2.2569.

[26] A. Fernández, S. Garcia, F. Herrera, dan N. V. Chawla, "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of Artificial Intelligence Research*, vol. 61, pp. 863-905, 2018, doi: 10.1613/jair.1.11192

[27] A. Alazba dan H. Aljamaan, "Software defect prediction using stacking generalization of optimized tree-based ensembles," *Applied Sciences*, vol. 12, no. 9, pp. 4577, 2022, doi: 10.3390/app12094577

[28] C. Zhang, J. Song, Z. Pei, and J. Jiang, "An Imbalanced Data Classification Algorithm of De-noising Auto-Encoder Neural Network Based on SMOTE," MATEC Web of Conferences ICCAE 2016, 2016, doi: 10.1051/conf/2016.

[29] Z. Ye, Y. Xu, Q. He, M. Wang, W. Bai, and H. Xiao, "Feature Selection Based on Adaptive Particle Swarm Optimization with Leadership Learning," *Comput. Intell. Neurosci.*, vol. 2022, 2022, doi: 10.1155/2022/1825341.

[30] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 10031–10061, 2022, doi: 10.1109/ACCESS.2022.3142859.

[31] M. Banga, A. Bansal, and A. Singh, "Proposed hybrid approach to predict software fault detection," *Int. J. Performability Eng.*, vol. 15, no. 8, pp. 2049–2061, 2019, doi: 10.23940/ijpe.19.08.p4.20492061.

[32] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Survey," *IEEE Access*, vol. 10, pp. 10031–10061, 2022, doi: 10.1109/ACCESS.2022.3142859.

[33] S. Mirjalili, "How effective is the Grey Wolf optimizer in training multi-layer perceptrons," Applied Intelligence, vol. 43, pp. 150-161, 2015, doi: 10.1007/s10489-014-0645-7

[34] O. O. Akinola, A. E. Ezugwu, J. O. Agushaka, R. A. Zitar, dan L. Abualigah, "Multiclass feature selection with metaheuristic optimization algorithms: a review," Neural Computing and

Applications, vol. 34, no. 22, pp. 19751-19790, 2022, doi: 10.1007/s00521-022-07705-4

[35] A. Kaveh dan P. Zakian, "Improved GWO algorithm for optimal design of truss structures," Engineering with Computers, vol. 34, pp. 685-707, 2018, doi: 10.1007/s00366-017-0567-1

[36] S. Mirjalili, S. M. Mirjalili, dan A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46-61, 2014, doi: 10.1016/j.advengsoft.2013.12.007

[37] F. A. Şenel et al., "A novel hybrid PSO-GWO algorithm for optimization problems," Engineering with Computers, vol. 35, pp. 1359-1373, 2019, doi: 10.1007/s00366-018-0668-5

[38] J. Teng, J. Lv, L. Guo, "An improved hybrid grey wolf optimization algorithm," Soft Computing, vol. 23, pp. 6617-6631, 2019, doi: 10.1007/s00500-018-3310-y.

[39] S. Mehta and K. S. Patnaik, "Improved prediction of software defects using ensemble machine learning techniques," *Neural Computing and Applications*, vol. 33, no. 16, pp. 10551-10562, 2021, doi: 10.1007/s00521-021-05811-3

[40] S. Ramraj, N. Uzir, R. Sunil, and S. Banerjee, "Experimenting XGBoost algorithm for prediction and classification of different datasets," *International Journal of Control Theory and Applications*, vol. 9, no. 40, pp. 651-662, 2016, [Online], Available: https://www.researchgate.net/publication/318132203_Experimenting_XGBoost_Algorithm_for_Prediction_and_Classification_of_Different_Datasets

[41] R. Hoque, S. Das, M. Hoque, and E. Haque, "Breast Cancer Classification using XGBoost," World Journal of Advanced Research and Reviews, vol. 21, no. 2, pp. 1985-1994, 2024, doi: 10.30574/wjarr.2024.21.2.0625

[42] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785-794, 2016 doi: 10.1145/2939672.2939785

[43] M. R. Ansyari, M. I. Mazdadi, F. Indriani, D. Kartini, and T. H. Saragih, "Implementation of Random Forest and Extreme Gradient Boosting in the Classification of Heart Disease Using Particle Swarm Optimization Feature Selection," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 5, no. 4, pp. 250-260, 2023. doi: 10.35882/jeemi.v5i4.322

[44] H. Mo, H. Sun, J. Liu, and S. Wei, "Developing window behavior models for residential buildings using XGBoost algorithm," *Energy and Buildings*, vol. 205, p. 109564, 2019. doi: 10.1016/j.enbuild.2019.109564

[45] Y. Wang and T. Wang, "Application of improved LightGBM model in blood glucose prediction," *Applied Sciences*, vol. 10, no. 9, p. 3227, 2020. doi: 10.3390/app10093227

[46] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems 30, 2017, [Online], Available: https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[47] D. D. Rufo, T. G. Debelee, A. Ibenthal, and W. G. Negera, "Diagnosis of diabetes mellitus using gradient boosting machine (LightGBM)," *Diagnostics*, vol. 11, no. 9, p. 1714, 2021. doi: 10.3390/diagnostics11091714

[48] S. Jhaveri, I. Khedkar, Y. Kantharia, and S. Jaswal, "Success prediction using random forest, catboost, xgboost and adaboost for kickstarter campaigns," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1170-1173, IEEE, 2019. doi: 10.1109/ICCMC.2019.8819828

[49] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems 31*, 2018, [Online], Available: https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html

[50] G. Huang, L. Wu, X. Ma, W. Zhang, J. Fan, X. Yu, W. Zeng, and H. Zhou, "Evaluation of CatBoost method for prediction of reference evapotranspiration in humid regions," *Journal of Hydrology*, vol. 574, pp. 1029-1041, 2019. doi: 10.1016/j.jhydrol.2019.04.085

[51] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299-310, 2005. doi: 10.1109/TKDE.2005.50

[52] D. Valero-Carreras, J. Alcaraz, and M. Landete, "Comparing two SVM models through different metrics based on the confusion matrix," Comput Oper Res, vol. 152, Apr. 2023, doi: 10.1016/j.cor.2022.106131

[53] C. Cortes and M. Mohri, "AUC optimization vs. error rate minimization," in *Advances in Neural Information Processing Systems 16*, 2003.

[54] A. R. Syulistyo, D. M. J. Purnomo, M. F. Rachmadi, and A. Wibowo, "Convolutions Subsampling Convolutions Gaussian connection Full connection Full connection Subsampling," JIKI (*Jurnal Ilmu Komput. dan Informasi*) UI, vol. 9, no. 1, pp. 52–58, 2016

[55] X. U. Manfei, D. Fralick, J. Z. Zheng, B. Wang, and F. E. N. G. Changyong, "The differences and similarities between two-sample t-test and paired t-test," *Shanghai Archives of Psychiatry*, vol. 29, no. 3, p. 184, 2017. doi: 10.11919/j.issn.1002-0829.217070

[56] G. B. Limentani, M. C. Ringo, F. Ye, M. L. Bergquist, and E. O. McSorley, "Beyond the t-test: statistical equivalence testing,", pp. 221-A, 2005, doi: 10.1021/ac053390m

[57] R. Sefira, A. Setiawan, R. Hidayatullah, and R. Darmayanti, "The Influence of the Snowball Throwing Learning Model on Pythagorean Theorem Material on Learning Outcomes," Edutechnium Journal of Educational Technology, vol. 2, no. 1, pp. 1-7, 2024. [Online], Available:https://edutechnium.com/journal/index.php/edutechnium/article/view/37

[58] K. Muthukumaran, A. Rallapalli, and N. L. Bhanu Murthy, "Impact of feature selection techniques on bug prediction models," *ACM Int. Conf. Proceeding Ser.*, vol. 18-20-Febr, pp. 120–129, 2015, doi: 10.1145/2723742.2723754.

[59] A. Kalsoom, M. Maqsood, M. A. Ghazanfar, F. Aadil, and S. Rho, *A dimensionality reduction-based efficient software fault prediction using Fisher linear discriminant analysis (FLDA)*, vol. 74, no. 9. Springer US, 2018. doi: 10.1007/s11227-018-2326-5.

[60] A. Iqbal and S. Aftab, "A classification framework for software defect prediction using multi-filter feature selection technique and MLP," *Int. J. Mod. Educ. Comput. Sci.*, vol. 12, no. 1, pp. 18–25, 2020, doi: 10.5815/ijmecs.2020.01.03.

[61] F. Meng, W. Cheng, and J. Wang, "An Integrated Semi-supervised Software Defect Prediction Model," *Journal of Internet Technology*, vol. 24, no. 6, pp. 1307-1317, 2023, doi:10.53106/160792642023112406013

BIBLIOGRAPHY

Angga Maulana Akbar originated in Banjarbaru, South Kalimantan. Since 2020, he has pursued his academic endeavors as a student of the Computer Science Department at Universitas Lambung Mangkurat. His current area of research interest is centered on software defect prediction. Additionally, his final assignment involves research centered on predicting defects in software. The goal of his research is to improving defect prediction in software, especially in NASA MDP dataset.

Rudy Herteno is currently a lecturer in the Faculty of Mathematics and Natural Science, Lambung Mangkurat University. He received his bachelor's degree in Computer Science from Lambung Mangkurat University and a master's degree in Informatics from STMIK Amikom University. His research interests include software engineering, software defect prediction and deep learning.
Email: rudy.herteno@ulm.ac.id.

Setyo Wahyu Saputro is a lecturer in Computer Science Department, Faculty of Mathematics and Natural Science, Lambung Mangkurat University in Banjarbaru. He received bachelor's degree also in Computer Science from Lambung Mangkurat Univesity, and received his master's degree in Informatics from STMIK Amikom University. His research interests include software engineering and artifial intelligence applications. He can be contacted at email: setyo.saputro@ulm.ac.id

Mohammad Reza Faisal received the B.Sc. and M.Eng. degrees in physics and informatics from Bandung Institute of Technology, Bandung, Indonesia, in 2004 and 2013. He also received a B.Eng. degree in informatics from Pasundan University, Bandung, Indonesia, in 2002 and a Ph.D. in computer science from Kanazawa University, Ishikawa, Japan, in 2018. He is currently a lecturer in the Computer Science Department, Faculty of Mathematics and Natural Sciences, Lambung Mangkurat University in Banjarbaru, Indonesia. His research interests include artificial intelligence applications, text mining, and software engineering. He can be contacted at email: reza.faisal@ulm.ac.id.

Radityo Adi Nugroho received his bachelor's degree in Informatics from the Islamic University of Indonesia and a master's degree in Computer Science from Gadjah Mada University. Currently, he is an assistant professor in the Department of Computer Science at Lambung Mangkurat University. His research interests include software defect prediction and computer vision. He can be contacted at email: radityo.adi@ulm.ac.id.