Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics e-ISSN: 2656-8624
Homepage: https://ijeeemi.org/; Vol. 6, No. 3, pp. 232-244, April 2025

RESEARCH PAPER

OPEN ACCES

Hybrid Feature Selection and Balancing Data Approach for Improved Software Defect Prediction

Muhamad Michael Febrian, Setyo Wahyu Saputro, Triando Hamonangan Saragih, Friska Abadi, and Rudy Herteno

Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia

ABSTRACT

Software Defect Prediction (SDP) plays a vital role in identifying defects within software modules. Accurate early detection of software defects can reduce development costs and enhance software reliability. However, SDP remains a significant challenge in the software development lifecycle. This study employs Particle Swarm Optimization (PSO) and addresses several challenges associated with its application, including noisy attributes, high-dimensional data, and imbalanced class distribution. To address these challenges, this study proposed a hybrid filter-based feature selection and class balancing method. The feature selection process incorporates Chi-Square (CS), Correlation-Based Feature Selection (CFS), and Correlation Matrix-Based Feature Selection (CMFS), which have been proven effective in reducing noisy and redundant attributes. Additionally, the Synthetic Minority Over-sampling Technique (SMOTE) is applied to mitigate class imbalance in the dataset. The K-Nearest Neighbors (KNN) algorithm is employed as the classification model due to its simplicity, non-parametric nature, and suitability for handling the feature subsets produced. Performance evaluation is conducted using the Area Under Curve (AUC) metric with a significance threshold of 0.05 to assess classification capability. The proposed method achieved an AUC of 0.872, demonstrating its effectiveness in enhancing predictive performance. The proposed method was also superior to other combinations such as PSO SMOTE (0.0043), PSO SMOTE CS (0.0091), PSO SMOTE CFS (0.0111), and PSO SMOTE CFS CMFS (0.0007). The findings of this study show that the proposed method significantly enhances the efficiency and accuracy of PSO in software defect prediction tasks. This hybrid strategy demonstrates strong potential as a robust solution for future research and application in predictive software quality assurance.

PAPER HISTORY

Received Feb 02, 2025
Accepted April 10, 2025
Published April 23, 2025

KEYWORDS

Balancing Class;
Multi Filter Feature;
Particle Swarm Optimization;
Software Defect Prediction

AUTHOR EMAIL

michaelfebr5@gmail.com
setyo.saputro@ulm.ac.id
triando.saragih@ulm.ac.id
friska.abadi@ulm.ac.id.
rudy.herteno@ulm.ac.id

1. INTRODUCTION

Ensuring software quality and reliability is crucial in software development, establishing defect prediction as a fundamental component of the quality assurance process [1]. The presence of software defects can lead to unexpected outcomes, including system failures that may compromise reputation, violate user privacy, or even pose risks to safety in certain critical applications [2][3]. Consequently, early detection of software defects is an essential step to mitigate potential negative impacts [4][5]. However, the development of software defect prediction models faces several challenges, particularly concerning dataset quality. These challenges include noisy attributes [6], high data dimensionality [7], and imbalanced class distribution [8]. For instance, the NASA dataset revealed that noisy attributes and an uneven distribution of classes significantly reduced the accuracy of software defect prediction models.

The Software Defect Prediction (SDP) model developed in this study achieved an Area Under the Curve (AUC) score of 0.872. This study builds upon various relevant references from prior research. In 2020, Iqbal et al. proposed a multi-filter and multi-layer approach that incorporated Correlation-Based Attribute Selection and Correlation Matrix Analysis with PSO for SDP, resulting in an AUC of 0.817. During the same period, Balogun et al utilized multi-filter feature selection aggregation, including CS, which resulted in an AUC value of 0.746. Meanwhile, in 2018, Balogun et al applied the Correlation technique combined with the Bagged KNN model, achieving an AUC value of 0.81. More recently, Sharma et al. (2024) explored Ensemble Learning and Class Imbalance Handling Strategies to enhance SDP performance, reporting a final AUC of 0.72. These previous studies provided the foundation for this research, which integrates a multi-filter selection approach with class balancing

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

techniques to achieve a notable enhancement in AUC performance.

Theoretically, this study demonstrates that multi-filter feature selection methods are effective in managing noisy attributes and reducing high-dimensional data, thereby enhancing the predictive performance of machine learning models. Particle Swarm Optimization (PSO) is recognized as an efficient optimization algorithm by researchers [9], particularly for its adaptive capability in exploring the search space and identifying optimal feature subsets [10]. Compared to other metaheuristic optimization methods, such as Genetic Algorithm, PSO offers faster convergence and more efficient search space exploration, making it more suitable for prediction tasks. However, the application of PSO presents several challenges, particularly its inefficiency in handling high-dimensional data, which increases computational complexity and reduces model performance [11][12]. To address this limitation, feature selection techniques, such as the Chi-Square (CS) method, are applied. CS is particularly effective for software defect prediction (SDP) as it evaluates the statistical relationship between each feature and the target variable, allowing the selection of only the most relevant attributes [13]. SDP datasets often contain highly correlated features that CS alone cannot effectively detect. To overcome this limitation, a multi-filter aggregation approach is employed by integrating Correlation-Based Feature Selection (CFS) and Correlation Matrix-Based Feature Selection (CMFS). While CS is limited in identifying feature dependencies, CFS and CMFS assess both feature-target correlations and inter-feature relationships. These methods ensure that essential features are retained while eliminating redundancy, thereby refining the feature subset [14][15]. The integration of these selection techniques results in a smaller, more informative, and redundancy-free feature set, ultimately improving computational efficiency and enhancing the predictive accuracy of software defect prediction models.

The K-Nearest Neighbors (KNN) algorithm was chosen for this study due to its simplicity, effectiveness, and non-parametric nature, making it a suitable approach for Software Defect Prediction (SDP). Unlike more complex classification models, KNN does not require extensive parameter tuning or training, allowing it to adapt flexibly to various datasets. To optimize KNN's performance, Grid Search was employed for hyperparameter tuning, systematically exploring a predefined set of parameter values to identify the optimal configuration for classification tasks. Despite these optimizations, KNN remains susceptible to class imbalance, as the majority class often dominates the neighborhood selection process, leading to misclassification of minority class instances. To address this limitation, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to balance class distribution by generating synthetic samples for the minority class. Differing from random oversampling, which merely duplicates existing instances, SMOTE creates new synthetic samples by interpolating data points between minority class instances, preserving data variability and mitigating overfitting [16]. This enhancement allows KNN to refine its decision boundaries more effectively, ensuring that minority class patterns are better captured and improving the overall predictive performance of SDP models [17]. The integration of multiple techniques in this study aims to address the challenges associated with software defect prediction, ultimately enhancing the AUC performance of the predictive model.

2. MATERIALS AND METHOD

The proposed research employed a structured methodology integrating feature selection, class balancing, and classification techniques to enhance Software Defect Prediction (SDP). The NASA MDP dataset was duplicated into three copies to facilitate parallel processing during the feature selection phase. The Chi-Square (CS) method was applied to one copy to identify the top 10 most relevant features based on their statistical dependency with the target variable. In parallel, The feature weights obtained from Correlation-based Feature Selection and Correlation Matrix Feature Selection were averaged, and the top 10 features were selected based on this combined ranking. The selected features from CS and the aggregated features from CFS-CMFS were merged using a union operation. This process produced a final feature subset ranging from 10 to 20 features.

As illustrated in Fig. 1, the refined feature subset was utilized in the classification process, followed by the application of Particle Swarm Optimization (PSO) to optimize feature selection. Within PSO, the K-Nearest Neighbors (KNN) algorithm served as the classification model, while the Synthetic Minority Over-sampling Technique (SMOTE) was applied to address class imbalance by generating synthetic samples for minority instances. To enhance classification efficiency, Grid Search was integrated within PSO to fine-tune the k-nearest neighbor parameter, identifying the optimal value for KNN. The evaluation framework implemented 10-fold cross-validation to validate model performance across multiple training and testing partitions. The predictive performance was assessed using Area Under the Curve (AUC), Precision, Recall, and F-Measure, ensuring a comprehensive evaluation of classification effectiveness. Additionally, a t-test was conducted to determine the statistical significance of improvements in AUC, validating the robustness of the proposed method. This experimental design enables a rigorous and systematic comparison across multiple variations of the proposed approach. The integration of multiple evaluation metrics and statistical analysis provides strong empirical support for the reliability and generalizability of the model across diverse software defect datasets. The comprehensive nature of this methodology further emphasizes its applicability for broader software quality assurance tasks in real-world development environments.

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

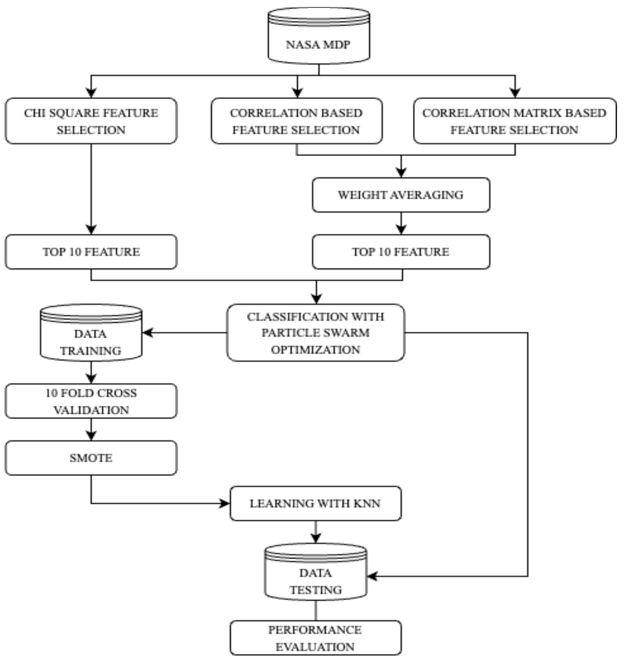

Fig. 1. Research method flowchart

A. Data Collection

This study employed the NASA Metrics Data Program (MDP) D'' dataset, an openly accessible dataset made available to support verification and reproducibility. The dataset comprises information from actual software projects spanning multiple domains, developed using programming languages such as Java, C, and C++ [19]. This dataset is widely recognized and frequently used by researchers as a benchmark for evaluating the accuracy of software defect prediction models [19]. For this study, 11 datasets were carefully chosen to encompass a range of scenarios and varying complexities in software defect prediction. The selected datasets included CM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. These datasets serve as a benchmark for building and assessing predictive models, primarily aimed at detecting software defects in the early phases of development [20]. Despite their significance, the NASA datasets pose several challenges, such as an imbalanced class distribution [21], the presence of noisy features [22], and a high number of attributes [23]. Table 1 provides a comprehensive overview of each dataset utilized in this study.

Table 1. Dataset Nasa

Dataset	Number of features	Sample size	Number of Defect	Defect %
CM1	38	327	42	12.8
KC1	22	1162	294	20.8
KC3	40	194	36	18.5
MC1	38	1988	46	2.3
MC2	40	124	44	35.4
MW1	38	253	27	10.7
PC1	38	705	61	9.7
PC2	37	722	16	2.2
PC3	38	1077	134	12.4
PC4	38	1270	176	13.8
PC5	39	1694	458	27.0

B. Splitting Dataset

The dataset is typically partitioned into two primary subsets: training and testing data, both of which play a vital role in the development and evaluation of machine learning models. Cross-validation functions as a validation approach for machine learning algorithms, designed to enhance the reliability of performance evaluation while reducing the likelihood of overfitting. In this study, the dataset is divided into 10 approximately equal folds using the Stratified K-Fold Cross-Validation technique. The model undergoes training over 10 iterations, where, in each cycle, one fold is assigned as the test set, while the remaining nine folds serve as the training set. The stratification process ensures that each fold retains the same class proportion as the original dataset, thereby maintaining class balance across all partitions and improving the robustness of model evaluation.

C. Chi-Square

Chi-Square (CS) is a filter-based feature selection method that evaluates the statistical dependency between features and the target variable. In its application, each feature in the dataset is assessed to determine its level of dependency on the target label, after which the features are ranked in descending order of significance [24]. One of the primary advantages of this method is its flexibility, as it can be applied to both nominal and numerical data types [25]. The fundamental Chi-Square (CS) formula is presented in Equation (1) [12].

$$CS(t\_k, ci) = (N(AD - CB)\hat{}2)/((A + C) + (B + D) + (A + B) + (C + D)) \tag{1}$$

The formula represents the Chi-Square value for feature tk concerning class ci, N denotes the total number of observations, and A,B,C,D correspond to frequency values derived from the contingency table. This formula measures the degree of association between a given feature and the target variable.

D. Correlation-Based Feature Selection

Correlation-Based Feature Selection (CFS) is a filter-based feature selection method designed to identify relevant features that enhance the performance of classification models [26]. CFS evaluates feature subsets based on the hypothesis that the optimal features are those with a high correlation to the target variable. This method employs the Pearson correlation coefficient, which ranges from -1 to 1, to indicate the degree of

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

positive, negative, or no correlation. The correlation is computed after normalizing the data [27]. In practice, CFS identifies features with a high correlation to the target variable as having stronger predictive potential [26]. However, this method does not account for relationships between selected features, as it primarily focuses on the direct relationship between individual features and the target variable [29]. By prioritizing feature-target correlations, CFS aims to identify significantly relevant features while disregarding those with low correlation, thereby reducing the risk of multicollinearity within the model [30][31]. The formula for calculating the Pearson Coefficient is presented in Equation (2) [28].

$$r = \frac{\sum(x_i - x)(y_i - y)}{\sqrt{\sum(x_i - x)^2 \sum(y_i - y)^2}} \quad (2)$$

By identifying features that have the highest contribution to classification performance, this approach enhances model interpretability while simultaneously reducing computational complexity.

E. Correlation-Based Feature Selection

Correlation Matrix-Based Feature Selection (CMFS) is a filter-based feature selection method designed to analyze and manage the relationships between features within a dataset [32]. This technique employs the Pearson correlation coefficient to quantify the degree of linear correlation between features in a software dataset (2) [33]. The correlation matrix consists of pairwise correlation values, where each entry represents the Pearson coefficient for a specific feature pair. By providing a comprehensive understanding of feature interdependencies, CMFS serves as an effective approach for enhancing prediction accuracy and facilitating optimal data-driven decision-making [34].

F. Weight Averaging

Weight averaging method integrates the weights obtained from CFS and CMFS, generating a refined weight representation that captures both the linear relationship between features and the target, as well as inter-feature correlations. The CFS method assesses the correlation of each feature with the target variable, while CMFS evaluates the correlation among features within the dataset. The weight values derived from both methods underwent normalization and averaging to produce the final weight distribution. This aggregated weight is then utilized to select the top 10 features, ensuring high relevance to the target while minimizing redundancy among features. Table 2 presents the pseudocode for finding the weight averaging.

Table 2. Algorithm 1

Weight Averaging

1. **BEGIN**
2. Dataset = fetch_dataset("NASA MDP D")
3. Feature_weight_corr = Correlation_based_feature_selection(Dataset)
4. Feature_weight_corrMatrix = Correlation_matrix_based_feature_selection(Dataset)
5. Average_feature_weight =(Feature_weight_corr + Feature_weight_corrMatrix) / 2
6. Output_weight = Average_feature_weight
7. **END**

G. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a swarm intelligence-based algorithm that simulates the collective movement of a flock of birds to identify optimal solutions [35]. In PSO, each particle represents a candidate solution, specifically a subset of features in this context, with its position in the search space determined by a fitness function that evaluates solution quality (3). Through an iterative process, particles dynamically adjust their positions, progressively converging toward an optimal solution (4) [36]. The optimization performance of PSO is influenced by several key parameters. Particle Best (Pb) denotes the best solution found by an individual particle in previous iterations, while Global Best (Gb) represents the best solution identified by the entire swarm during the optimization process (3). Additionally, acceleration coefficients (c1 and c2) regulate the influence of personal and global best positions, whereas random factors (r1 and r2) introduce stochasticity in particle movement (3). The number of iterations is denoted by t, while the inertia weight (ω) controls the balance between exploration and exploitation, preventing premature convergence and improving search efficiency [33].

$$V_i(t+1) = \omega V_i(t) + c_1 r_1 (Pb_i(t) - X_i(t) + c_2 r_2 (Gb_i(t) - x_i(t))) \quad (3)$$

$$V\_i\,(t+1) = X\_i\,(t) + V\_i\,(t+1) \quad (4)$$

The flow of the Particle Swarm Optimization (PSO) process is as follows:

1. Particle Initialization and Fitness Evaluation: PSO randomly generates a population of particles, each representing a feature subset, and evaluates their fitness using AUC.
2. Best Solution Update (Pb and Gb): Each particle updates its Particle Best (Pb) and Global Best (Gb), adjusting its position and velocity to improve the search for an optimal solution.
3. Feature Subset Selection and Iterative Optimization: Particles refine their positions iteratively, updating Pb and Gb, until a stopping criterion is met.
4. Final Evaluation: The best feature subset is used to train the prediction model, and its performance is assessed using AUC.

H. Synthetic Minority Over-Sampling Technique

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

Synthetic Minority Over-sampling Technique (SMOTE) is a widely applied method for addressing class imbalance by generating synthetic samples for minority classes to achieve a more balanced distribution relative to the majority class [16]. Rather than duplicating existing samples, this technique enhances the learning process of classification models by synthesizing new data points. The SMOTE process begins by identifying samples from the minority class and determining their k-nearest neighbors. Synthetic samples are then generated along the vector between an original sample and one of its selected neighbors. This approach improves class distribution, enabling the predictive model to learn patterns from both majority and minority classes more effectively, thereby enhancing classification performance [19]. Table 3 presents the pseudocode of SMOTE.

Table 3. **Algorithm 2**

Synthetic Minority Over-Sampling Technique
1. **BEGIN**
2. Input:
3. O ← Original dataset
4. P ← Set of positive instances (minority class)
5. Process:
6. For each instance x in P do:
7. Find k-nearest neighbors of x within P
8. Randomly select one neighbor y from k-nearest neighbors
9. Compute the difference: difference = x - y
10. Generate a random number gap between 0 and 1
11. Create synthetic instance: n = x + (difference * gap)
12. Add n to O
13. Output:
14. O ← Augmented dataset with synthetic instances
15. **END**

I. K-Nearest Neighbour

K-Nearest Neighbors (KNN) is an instance-based machine learning algorithm commonly used for classification and regression tasks [37]. It determines the class or value of new data by analyzing the majority class or average value of its nearest neighbors. The primary objective of KNN is to classify or predict new data based on its proximity to existing data points in the feature space. KNN implementation flow [38]:

1. Initialize the number of neighbors (k) and calculate the distance between the new data and existing data.
2. Identify the k closest neighbors and determine the class by majority voting or compute the average for regression.
3. Assign the final prediction based on the majority class or calculated value.

Fig. 2 illustrates the working mechanism of the KNN algorithm, depicting how nearest neighbors influence classification and prediction outcomes [39].

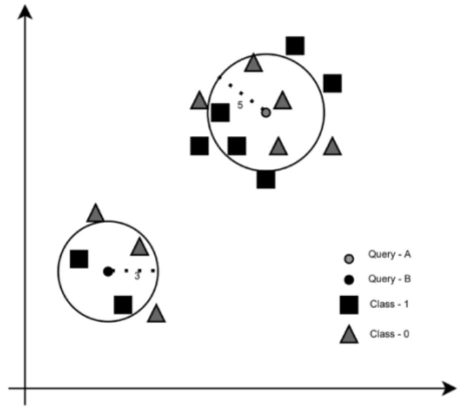

Fig. 2. **KNN ALGORITHM**

J. Grid Search

Grid Search is a systematic approach for hyperparameter tuning that aims to determine the optimal parameter values for machine learning models. One of its primary advantages is its exhaustive search process, which evaluates all possible parameter combinations within a predefined range, ensuring the selection of the most effective configuration for enhancing model performance. In this study, Grid Search is utilized to optimize the K-Nearest Neighbors (KNN) algorithm by determining the most appropriate number of neighbors. The application of this method ensures that KNN operates with the most optimal parameter configuration, thereby improving classification performance and overall model reliability.

K. Evaluation

The Area Under Curve (AUC) is employed as a classification performance evaluation metric, providing a clear measure of a model's ability to distinguish between different classes. This comprehensive approach ensures a thorough understanding and assessment of the significance of each applied method. The AUC metric, represented in Equation (5), ranges from 0 to 1, where a value of 1 signifies perfect class separation, while a value of 0.5 indicates a classification performance equivalent to random guessing. A higher AUC value denotes better predictive performance [40]. Additionally, statistical evaluation was conducted using a t-Test to compare the average performance of different models and determine the statistical significance of AUC differences [41]. A significant t-Test result reinforces confidence in the superiority of one method over another [41][42]. A significance threshold (α) of 0.05 was applied to ensure a 95% confidence level. While the α value can be adjusted based on specific requirements, 0.05 is widely recognized as a standard threshold in statistical analysis [42].

$$AUC = \frac{1 + TPR - FPR}{2} \tag{5}$$

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

To obtain this information, a confusion matrix is utilized. Equations (6) and (7) [43] are used to calculate the True Positive Rate (TPR) and False Positive Rate (FPR), where TPR represents the proportion of correctly classified positive instances, and FPR denotes the proportion of negative instances incorrectly classified as positive [43].

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

Table 4 presents the pseudocode outlining the steps for the final model evaluation, providing a structured approach to assessing the model's performance.

Table 4. Algorithm 3

MODEL EVALUATION

1. **BEGIN**
2. Dataset = Load_NASA_MDP()
3. Weigth_CS = Compute_ChiSquare(Dataset)
4. 10_CS_Features = Select_Top_Features(ChiSquareScores, k=10)
5. Weights_CFS = Compute_CFS_Weights(Dataset)
6. Weights_CMFS = Compute_CMFS_Weights(Dataset)
7. Feature_Names = Get_Feature_Names(Dataset)
8. Average_Weights = {}
9. For each feature in Feature_Names do
10. Average_Weights[feature] = (Weights_CFS[feature] + Weights_CMFS[feature]) / 2
11. EndFor
12. 10_Average_Weight_Features = Select_Top_Features(AverageWeights, k=10)
13. Selected_Features = Merge_Features(10_CS_Features, 10_Average_Weight_Features)
14. TrainSet, TestSet = Split_Dataset(Selected_Features, train_ratio=0.8, test_ratio=0.2)
15. TrainSet_Balanced = Apply_SMOTE(TrainSet)
16. PSO_Performance = PSO.FeatureSelection(model=CrossVal.KFold(model=KNN(data= TrainSet_Balanced), k=10), data=TestSet)
17. Performance=PSOPerformance.GetAUC()
18. **END**

3. RESULTS

This study consists of three main phases. The first phase involves feature selection, where relevant features are identified using Chi-Square (CS), Correlation-Based Feature Selection (CFS), and Correlation Matrix-Based Feature Selection (CMFS). The selected features are then combined to form an optimal subset for processing in Particle Swarm Optimization (PSO). In the second phase, PSO is employed to optimize the selected features while incorporating the K-Nearest Neighbors (KNN) algorithm as the classification model and addressing class imbalance through the Synthetic Minority Over-sampling Technique (SMOTE). Furthermore, Grid Search is applied within PSO to fine-tune the k-nearest neighbor parameter, ensuring optimal classification performance. The final phase involves validation, where model performance is assessed using the Area Under the Curve (AUC) as the primary evaluation metric. Additionally, a t-test is

Table 5. PSO AUC result

Dataset	PSO	PSO SMOTE	PSO SMOTE CS	PSO SMOTE CFS	PSO SMOTE CMFS	**PSO SMOTE CS CFS CMFS**
CM1	0.794	0.771	0.805	0.838	0.793	**0.889**
KC1	0.716	0.706	0.696	0.702	0.714	**0.724**
KC3	0.803	0.794	0.801	0.768	0.825	**0.859**
MC1	0.884	0.926	0.902	0.898	0.899	**0.916**
MC2	0.774	0.746	0.838	0.848	0.834	**0.919**
MW1	0.844	0.83	0.882	0.885	0.763	**0.880**
PC1	0.897	0.907	0.912	0.913	0.913	**0.907**
PC2	0.791	0.843	0.791	0.777	0.861	**0.962**
PC3	0.822	0.82	0.838	0.841	0.818	**0.837**
PC4	0.895	0.905	0.919	0.905	0.866	**0.924**
PC5	0.781	0.781	0.775	0.781	0.751	**0.780**
AVERAGE	0.818	0.821	0.833	0.832	0.822	**0.872**

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

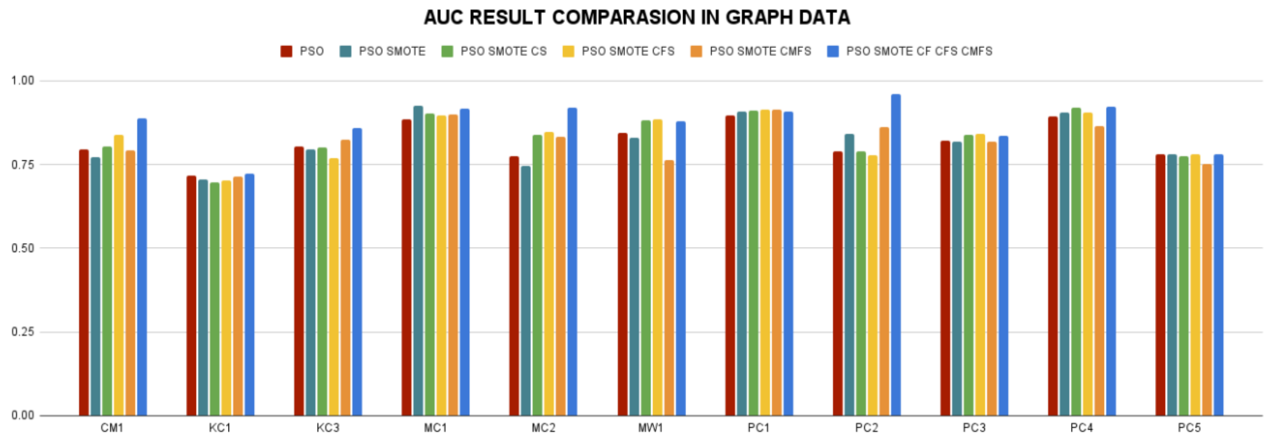

Figure. 3. AUC result comparasion in graph data

performed to determine the statistical significance of the improvements, confirming that the proposed method provides substantial enhancements in software defect prediction, particularly on the NASA MDP dataset.

To evaluate the performance of the proposed method, six approaches were conducted in this study. All approaches employed K-Nearest Neighbors as the classification model. The first approach applied the Particle Swarm Optimization method without any modifications. The Second approach integrated PSO with the Synthetic Minority Over-sampling Technique to address class imbalance. The third approach incorporated feature selection using Chi-Square before processing with PSO-SMOTE. The fourth approach utilized Correlation-Based Feature Selection for feature selection prior to processing in PSO-SMOTE. The fifth approach applied Correlation Matrix-Based Feature Selection to select features before being processed by PSO-SMOTE. Finally, the sixth approach combined feature selection using CS with feature aggregation from CFS and CMFS before processing in PSO-SMOTE.

The research findings presented in Table 5 indicate variations in AUC values obtained from 66 experiments conducted across 11 datasets. The results demonstrate a significant improvement in PSO performance when combined with various feature selection and class balancing techniques. The proposed method achieves a maximum AUC of 0.962 and a minimum AUC of 0.724, with an average AUC of 0.872, outperforming all other evaluated approaches. In comparison, PSO alone attains an average AUC of 0.818, while PSO SMOTE, PSO SMOTE CS, PSO SMOTE CFS, and PSO SMOTE CMFS achieve average AUC values of 0.821, 0.833, 0.832, and 0.822, respectively. The proposed approach exhibits superior performance in datasets such as CM1, KC1, KC3, MC2, PC2, and PC4, while in the remaining datasets, the differences in performance compared to previous methods are relatively minor. Although the proposed approach demonstrated a significant improvement in AUC values, certain datasets, such as KC1 and PC5, exhibit considerable deviations from the average AUC. The AUC values across different datasets are illustrated in Fig 3.

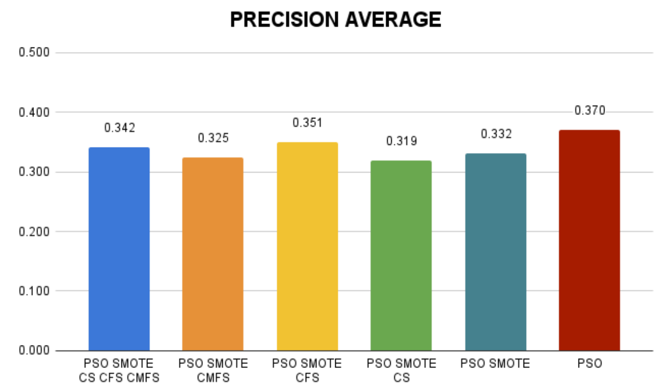

Fig. 4. Precision average

Fig 4 presents a comparative analysis of the average precision values obtained from various methods. The results indicate that the PSO method achieves the highest precision with an average value of 0.370, followed by PSO SMOTE CFS with a precision of 0.351. The proposed method, PSO SMOTE CS CFS CMFS, attains a precision of 0.342, surpassing PSO SMOTE (0.332), PSO SMOTE CMFS (0.325), and PSO SMOTE CS (0.319). Although the proposed method does not yield the highest precision, the difference remains relatively small, indicating its competitive performance in software defect prediction. This suggests that while the approach optimizes recall effectively, additional fine-tuning may be required to enhance precision without compromising other performance metrics. Furthermore, the relatively close precision values across different methods highlight the challenge of balancing recall and precision in software defect classification, emphasizing the need for further refinement in feature selection and classification strategies.

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

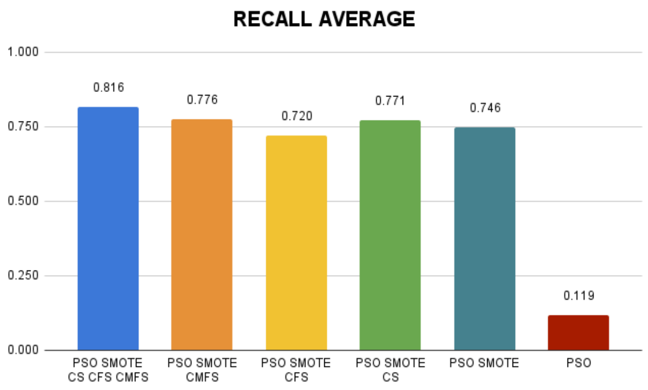

Fig. 5. Recall average

Fig 5 presents a comparison of the average recall values across different methods. The results indicate that the proposed method, PSO SMOTE CS CFS CMFS, achieves the highest recall with a value of 0.816, demonstrating a significant improvement over the other methods. PSO SMOTE CMFS follows with a recall of 0.776, while PSO SMOTE CS and PSO SMOTE attain recall values of 0.771 and 0.746, respectively. PSO SMOTE CFS achieves a recall of 0.720, whereas the standard PSO method records a substantially lower recall of 0.119.

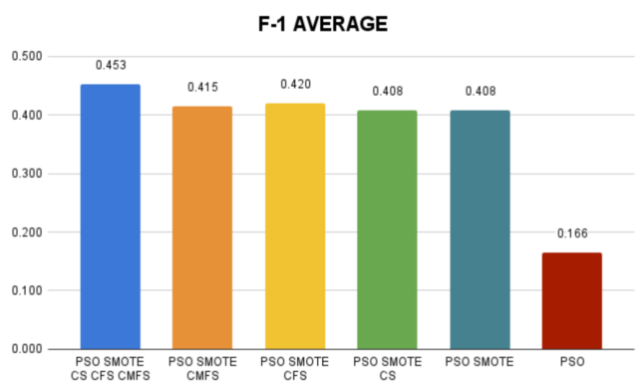

Fig. 6. F-1 average

Fig 6 presents a comparison of the average F1-score across different methods. The results show that the proposed method, PSO SMOTE CS CFS CMFS, achieves the highest F1-score of 0.453, surpassing all other approaches. PSO SMOTE CFS attains an F1-score of 0.420, followed by PSO SMOTE CMFS with 0.415, while PSO SMOTE CS and PSO SMOTE both obtain identical scores of 0.408. The baseline PSO method demonstrates a significantly lower performance, with an F1-score of 0.166. These findings highlight the effectiveness of the proposed approach in balancing precision and recall, making it a more reliable choice for software defect prediction.

Table 6. Methods Significance Comparasion

Method Comparison	t-test value	Significance
PSO SMOTE CS CFS CMFS - PSO	0.0027	Significance
PSO SMOTE CS CFS CMFS - PSO SMOTE	0.0043	Significance
PSO SMOTE CS CFS CMFS - PSO CMOTE CS	0.0091	Significance
PSO SMOTE CS CFS CMFS - PSO CMOTE CFS	0.0111	Significance
PSO SMOTE CS CFS CMFS - PSO CMOTE CMFS	0.0007	Significance

Furthermore, the significance of the results in this study was evaluated using a t-test, as presented in Table 6, which compares the t-test values between the standalone PSO, PSO with SMOTE, and the combination of PSO SMOTE with feature selection techniques, including CS, CFS, and CMFS. The first column represents the methods being compared, the second column displays the corresponding t-test values, and the third column indicates the statistical significance of the results. A t-test value below the alpha threshold of 0.050 is categorized as 'significant,' indicating a meaningful improvement in performance, whereas values above 0.050 are classified as 'insignificant.' In Table 6, the proposed method demonstrates a statistically significant improvement over the standalone and combined PSO methods, with t-test values of 0.0027 compared to PSO, 0.0043 compared to PSO SMOTE, 0.0091 compared to PSO SMOTE CS, 0.0111 compared to PSO SMOTE CFS, and 0.0007 compared to PSO SMOTE CMFS. As all obtained values are below the alpha threshold of 0.050, these findings confirm that the proposed method provides a substantial enhancement in performance. These results further validate the effectiveness of the proposed approach in addressing the limitations of PSO and improving the predictive capability of the model based on AUC values.

4. DISCUSSION

The study demonstrates a significant improvement in AUC through the integration of Particle Swarm Optimization (PSO), Synthetic Minority Over-sampling Technique (SMOTE), Chi-Square (CS), Correlation-Based Feature Selection (CFS), and Correlation Matrix Feature Selection (CMFS). As presented in Table 5 and Fig 3, the PSO SMOTE CS CFS CMFS method outperforms all other tested approaches. The method achieves optimal performance on the PC2 dataset,

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

obtaining an AUC of 0.962, while the lowest performance is observed on the KC1 dataset, with an AUC of 0.724. The most substantial improvement is observed in the PC2 dataset, where the proposed method attains an AUC of 0.962, significantly surpassing the performance of PSO (0.791), PSO SMOTE (0.843), PSO SMOTE CS (0.791), PSO SMOTE CFS (0.777), and PSO SMOTE CMFS (0.861). This enhancement underscores the effectiveness of the proposed approach in selecting the most relevant and highly correlated features. Furthermore, consistent improvements observed in other datasets, such as CM1 and KC3, indicate that the proposed method exhibits adaptability and robustness across datasets with varying characteristics.

Although the proposed method demonstrates superiority across most datasets, as presented in Table 5, there are instances where the AUC value obtained by the PSO SMOTE CS CFS CMFS method exhibits a slight decline. This decrease is observed in the MC1, MW1, PC1, PC3, and PC5 datasets. However, despite this variation, the difference in AUC values remains minimal. For instance, in the PC5 dataset, the PSO SMOTE CS CFS CMFS method achieved an AUC of 0.780, only marginally lower than the PSO SMOTE method, which recorded an AUC of 0.781. A similar pattern is observed in other datasets, where the variations in AUC values remain within a competitive range. Furthermore, based on the overall evaluation, PC5 and KC1 exhibited the lowest AUC values. This could be attributed to the large number of data entries, which increases the complexity of the classification task and may impact model performance. As shown in Table 1, these datasets contain a higher number of samples compared to others, potentially influencing prediction accuracy and overall model effectiveness.

In this study, as illustrated in Fig 4, the PSO SMOTE CS CFS CMFS method achieved a precision of 0.342, ranking third after PSO (0.370) and PSO SMOTE CFS (0.351). However, it outperformed other approaches, including PSO SMOTE (0.332), PSO SMOTE CMFS (0.325), and PSO SMOTE CS (0.319). Although the proposed method does not achieve the highest precision, the differences between methods are relatively minor, indicating that while there is no significant decline, there is also no substantial improvement. Furthermore, the precision values across all methods remain below 0.5, indicating a high probability of false alarms, where non-defective software components may be incorrectly classified as defective. This limitation stem from sub-optimal tuning of the KNN parameters and the application of SMOTE. The use of SMOTE generates a substantial number of synthetic positive class instances, which may introduce bias in the KNN classification process. This, coupled with the insufficient tuning of KNN parameters, is likely to have contributed to the lower precision values observed in the results.

On the other hand, as illustrated in Fig 5, the PSO SMOTE CS CFS CMFS method achieved a recall value of 0.816, significantly outperforming other methods, including PSO SMOTE CMFS (0.776), PSO SMOTE CFS (0.720), PSO SMOTE CS (0.771), PSO SMOTE (0.746), and PSO (0.119). These results indicate that the model effectively identifies nearly all software defects, demonstrating the success of the proposed approach in enhancing software defect prediction performance. The multi-filter feature selection approach proved to be effective in eliminating irrelevant features, while SMOTE played a crucial role in assisting the KNN model by generating synthetic data, thereby improving the model's ability to accurately predict software defects.

Furthermore, as depicted in Fig 6, the PSO SMOTE CS CFS CMFS method attained an F1-score of 0.453, surpassing the performance of PSO SMOTE CMFS (0.415), PSO SMOTE CFS (0.420), PSO SMOTE CS (0.408), PSO SMOTE (0.408), and PSO (0.166). This result highlights the effectiveness of the proposed method in improving F1-score performance compared to other approaches. However, all tested methods yielded an F1-score below 0.5, indicating an imbalance between precision and recall. This imbalance, particularly in cases where recall is high but precision is low, makes the model more susceptible to generating false alarms, leading to the incorrect classification of non-defective software as defective. One contributing factor to this issue is the application of SMOTE with the KNN model. Although these two methods work synergistically, the selection of a sub-optimal number of k-neighbors can cause the model to be biased toward synthetic data generated by SMOTE, thereby increasing the risk of overfitting.

Table 7. AUC Result Comparasion Of The Proposed Method With Other Studies

Study	Method	Avg. AUC
Balgoun et al. [44]	CfsSubsetEval Bagged KNN	0.759
Balogun et al. [13]	RMFFS NB CS	0.746
Iqbal and Aftab [45]	MLP MFFS ROS	0.817
Sharma et al. [46]	SMOTE MI RFE CV PCA KNN	0.72
Febrian et al.	**PSO SMOTE CS CFS CMFS**	**0.872**

Table 7 presents a comparative analysis between this study and previous research. The findings demonstrate that the proposed method significantly outperforms prior studies in terms of AUC, indicating its superior predictive capability. Compared to previous research using similar datasets, the proposed approach achieves a remarkable improvement in AUC, highlighting the effectiveness of combining PSO with multi-filter feature selection and SMOTE for class balancing. This notable enhancement confirms the potential of the proposed method in addressing noisy attributes, high-dimensional data and

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

class imbalance issues, making it a robust solution for software defect prediction.

Although this study demonstrated a significant improvement in AUC and recall values, certain aspects of the model's performance require further attention. However, while achieving high AUC and recall, the low precision and F1-score values across all methods indicate overfitting, making the models prone to false alarms. This issue arises due to bias in the KNN model, likely caused by sub-optimal tuning of the k parameter and the influence of synthetic data generated by SMOTE. Further research is necessary to refine model selection and the tuning process, both for the classification algorithm and the SMOTE technique, to enhance the balance between precision and recall while mitigating the risk of overfitting in software defect prediction.

Despite these limitations, our study demonstrated that the PSO SMOTE CS CFS CMFS method has significant potential for software defect prediction, especially when integrated with multi-filter feature selection and class balancing techniques. The findings indicate that the combination of feature selection and class balancing enhances PSO's capability in handling noisy attributes, high-dimensional data and imbalanced datasets. Further research is required to explore the optimal integration of PSO with other feature selection and classification techniques to further enhance its effectiveness in software defect prediction. This exploration could lead to the development of more robust, efficient, and accurate prediction models.

5. CONCLUSION

The findings of this research indicate that the PSO SMOTE CS CFS CMFS method holds substantial promise in predicting software defects, particularly through the application of multi-filter feature selection and class balancing techniques. The results highlight that incorporating feature selection and class balancing strengthens PSO's capability to process complex datasets, including those with noisy attributes, a high number of features, and class distribution imbalances. Further investigation is necessary to refine the integration of PSO with alternative feature selection and classification strategies, aiming to maximize its performance in predicting software defects. Future advancements in this area may contribute to the development of more resilient, efficient, and precise prediction models.

The proposed approach consistently exhibits strong classification performance, attaining an average AUC of 0.872. In comparison, applying PSO alone resulted in a lower AUC of 0.817. Incorporating SMOTE with PSO led to a marginal improvement, producing an AUC of 0.819. Additionally, integrating PSO SMOTE with CS contributed to a more pronounced enhancement, reaching an AUC of 0.833. Conversely, the combination of PSO SMOTE with CFS led to a minor reduction in AUC, decreasing to 0.832. Similarly, PSO SMOTE combined with CMFS also showed a slight decline, yielding an AUC of 0.822.

To evaluate the statistical significance of these results, a t-Test was performed by comparing the performance of the proposed PSO SMOTE CS CFS CMFS method with other alternative approaches. The findings revealed that the α-value remained below the 0.05 threshold, confirming that the differences in performance between the proposed method and the other tested approaches were statistically significant. This conclusion is further supported when comparing the proposed method to techniques from previous research, reinforcing its effectiveness. The statistical evaluation validates that PSO SMOTE CS CFS CMFS effectively addresses challenges related to noisy attributes, high data dimensionality, and class imbalance in SDP. Moreover, it significantly enhances prediction accuracy, demonstrating its robustness as an approach for improving software defect prediction models.

REFERENCES

[1] Z. Sun, J. Zhang, H. Sun, and X. Zhu, “Collaborative filtering based recommendation of sampling methods for software defect prediction,” Applied Soft Computing Journal, vol. 90, 2020, doi: 10.1016/j.asoc.2020.106163.

[2] C. Ni, X. Chen, F. Wu, Y. Shen, and Q. Gu, "An empirical study on pareto based multi-objective feature selection for software defect prediction," Journal of Systems and Software, vol. 152, pp. 215-238, Jun. 2019, doi: 10.1016/j.jss.2019.03.012.

[3] 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," Applied Soft Computing, vol. 146, p. 110659, Oct. 2023, doi: 10.1016/j.asoc.2023.110659.

[4] M. Ali et al., "Analysis of Feature Selection Methods in Software Defect Prediction Models," IEEE Access, vol. 11, pp. 145954-145974, 2023, doi: 10.1109/ACCESS.2023.3343249.

[5] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu, and Z. Chen, "SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 4, pp. 2244-2258, 2022, doi: 10.1109/TDSC.2021.3051525.

[6] M. J. Hernández-Molinos, A. J. Sánchez-García, R. E. Barrientos-Martínez, J. C. Pérez-Arriaga, and J. O. Ocharán-Hernández, “Software Defect Prediction with Bayesian Approaches,” Mathematics, vol. 11, no. 11, Jun. 2023, doi: 10.3390/math11112524.

[7] B. Khan et al., "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

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

imbalance reduction (CIR): A novel approach to software defect prediction in the presence of class imbalance," Symmetry, vol. 12, no. 3, Mar. 2020, doi: 10.3390/sym12030407.

[9] A. G. Gad, Particle swarm optimization algorithm and its applications: a systematic review, vol. 29, no. 5. Springer Netherlands, 2022.x

[10] 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.

[11] J. Divasón, A. Pernia-Espinoza, and F. J. Martinez-de-Pison, “HYB-PARSIMONY: A hybrid approach combining Particle Swarm Optimization and Genetic Algorithms to find parsimonious models in high-dimensional datasets,” Neurocomputing, vol. 560, Dec. 2023, doi: 10.1016/j.neucom.2023.126840.

[12] S. Bahassine, A. Madani, M. Al-Sarem, and M. Kissi, "Feature selection using an improved Chi-square for Arabic text classification," Journal of King Saud University - Computer and Information Sciences, vol. 32, no. 2, pp. 225-231, Feb. 2020, doi: 10.1016/j.jksuci.2018.05.010.

[13] A. O. Balogun et al., "Empirical analysis of rank aggregation-based multi-filter feature selection methods in software defect prediction," Electronics (Switzerland), vol. 10, no. 2, pp. 1-16, Jan. 2021, doi: 10.3390/electronics10020179.

[14] A. O. Balogun et al., "Data Sampling-Based Feature Selection Framework for Software Defect Prediction," in Lecture Notes in Networks and Systems, Springer Science and Business Media Deutschland GmbH, 2021, pp. 39-52. doi: 10.1007/978-3-030-80216-5_4.

[15] F. Wang, J. Ai, and Z. Zou, "A Cluster-Based Hybrid Feature Selection Method for Defect Prediction," in Proceedings - 19th IEEE International Conference on Software Quality, Reliability and Security, QRS 2019, Institute of Electrical and Electronics Engineers Inc., Jul. 2019, pp. 1-9. doi: 10.1109/QRS.2019.00014.

[16] V. Rupapara, F. Rustam, H. F. Shahzad, A. Mehmood, I. Ashraf, and G. S. Choi, "Impact of SMOTE on Imbalanced Text Features for Toxic Comments Classification Using RVVC Model," IEEE Access, vol. 9, pp. 78621-78634, 2021, doi: 10.1109/ACCESS.2021.3083638.

[17] J. H. Joloudari, A. Marefat, M. A. Nematollahi, S. S. Oyelere, and S. Hussain, "Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks," Applied Sciences (Switzerland), vol. 13, no. 6, Mar. 2023, doi: 10.3390/app13064006.

[18] S. Zhang, "Challenges in KNN Classification," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 10, pp. 4663-4675, Oct. 2022, doi: 10.1109/TKDE.2021.3049250.

[19] F. Matloob et al., "Software defect prediction using ensemble learning: A systematic literature review," IEEE Access, vol. 9. Institute of Electrical and Electronics Engineers Inc., pp. 98754-98771, 2021. doi: 10.1109/ACCESS.2021.3095559.

[20] Angga Maulana Akbar, R. Herteno, S. W. Saputro, M. R. Faisal, and R. A. Nugroho, "Optimizing Software Defect Prediction Models: Integrating Hybrid Grey Wolf and Particle Swarm Optimization for Enhanced Feature Selection with Popular Gradient Boosting Algorithm," Journal of Electronics, Electromedical Engineering, and Medical Informatics, vol. 6, no. 2, pp. 169-181, Apr. 2024, doi:10.35882/jeeemi.v6i2.388.

[21] S. Mcmurray and A. H. Sodhro, "A Study on ML-Based Software Defect Detection for Security Traceability in Smart Healthcare Applications," Sensors, vol. 23, no. 7, Apr. 2023, doi: 10.3390/s23073470.

[22] H. Alsawalqah et al., "Software defect prediction using heterogeneous ensemble classification based on segmented patterns," Applied Sciences (Switzerland), vol. 10, no. 5, Mar. 2020, doi: 10.3390/app10051745.

[23] H. Wei, C. Hu, S. Chen, Y. Xue, and Q. Zhang, "Establishing a software defect prediction model via effective dimension reduction," Inf Sci (N Y), vol. 477, pp. 399-409, Mar. 2019, doi: 10.1016/j.ins.2018.10.056.

[24] W. BinSaeedan and S. Alramlawi, "CS-BPSO: Hybrid feature selection based on chi-square and binary PSO algorithm for Arabic email authorship analysis," Knowl Based Syst, vol. 227, Sep. 2021, doi: 10.1016/j.knosys.2021.107224.

[25] F. Uğurlu, S. Yıldız, M. Boran, Ö. Uğurlu, and J. Wang, “Analysis of fishing vessel accidents with Bayesian network and Chisquare methods,” Ocean Engineering, vol. 198, Feb. 2020, doi: 10.1016/j.oceaneng.2020.106956.

[26] M. Shobana et al., "Classification and Detection of Mesothelioma Cancer Using Feature Selection-Enabled Machine Learning Technique," Biomed Res Int, vol. 2022, 2022, doi: 10.1155/2022/9900668.

[27] S. Chen, G. I. Webb, L. Liu, and X. Ma, "A novel selective na√Øve Bayes algorithm," Knowl Based Syst, vol. 192, Mar. 2020, doi: 10.1016/j.knosys.2019.105361.

[28] B. Sen Peng, H. Xia, Y. K. Liu, B. Yang, D. Guo, and S. M. Zhu, "Research on intelligent fault diagnosis method for nuclear power plant based on correlation analysis and deep belief network," Progress in Nuclear Energy, vol. 108, pp. 419-427, Sep. , doi: 10.1016/j.pnucene.2018.06.003.

[29] Y. He et al., "A correlation-based feature selection algorithm for operating data of nuclear power plants," Science and Technology of Nuclear Installations, 2021, doi: 10.1155/2021/9994340.

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

[30] J. Jiarpakdee, C. Tantithamthavorn, and C. Treude, "The impact of automated feature selection techniques on the interpretation of defect models," Empirical Software Engineering, vol. 25, no. 5, pp. 3590-3638, Sep. 2020, doi: 10.1007/s10664-020-09848-1.

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

[32] U. Islam et al., "Enhancing Economic Stability with Innovative Crude Oil Price Prediction and Policy Uncertainty Mitigation in USD Energy Stock Markets," Fluctuation and Noise Letters, vol. 23, no. 2, Apr. 2024, doi: 10.1142/S0219477524400212.

[33] E. Liu, J. Li, A. Zheng, H. Liu, and T. Jiang, "Research on the Prediction Model of the Used Car Price in View of the PSO-GRA-BP Neural Network," Sustainability (Switzerland), vol. 14, no. 15, Aug. 2022, doi: 10.3390/su14158993.

[34] Z. Zhang and J. Ma, "Feature selection combined feature resolution with attribute reduction based on correlation matrix of equivalence classes," International Journal of Performability Engineering, vol. 15, no. 4, pp. 1131-1140, 2019, doi: 10.23940/ijpe.19.04.p8.11311140.

[35] S. Feng et al., "COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction," Information and Software Technology, vol. 129, Jan. 2021, doi: 10.1016/j.infsof.2020.106432.

[36] B. Fu, Y. He, Q. Guo, and J. Zhang, "An improved competitive particle swarm optimization algorithm based on deheterogeneous information," Journal of King Saud University - Computer and Information Sciences, vol. 35, no. 6, Jun. 2023, doi: 10.1016/j.jksuci.2022.12.012.

[37] S. Zhang, “Challenges in KNN Classification,” IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 10, pp. 4663–4675, Oct. 2022, doi: 10.1109/TKDE.2021.3049250.

[38] B. Mumtaz, S. Kanwal, S. Alamri, and F. Khan, “Feature selection using artificial immune network: An approach for software defect prediction,” Intelligent Automation and Soft Computing, vol. 29, no. 3, pp. 669–684, 2021, doi: 10.32604/iasc.2021.018405.

[39] S. Uddin, I. Haque, H. Lu, M. A. Moni, and E. Gide, “Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction,” Scientific Reports, vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-10358-x.

[40] M. H. Murad, A. K. Balla, M. S. Khan, A. Shaikh, S. Saadi, and Z. Wang, "Thresholds for interpreting the fragility index derived from sample of randomised controlled trials in cardiology: a meta-epidemiologic study," BMJ Evidence-Based Medicine, vol. 28, no. 2, pp. 133-136, 2023, doi: 10.1136/bmjebm-2021-111858.

[41] J. L. Ortega, "The presence of academic journals on Twitter and its relationship with dissemination (tweets) and research impact (citations)," Aslib Journal of Information Management, vol. 69, no. 6, pp. 674-687, 2017, doi: 10.1108/AJIM-02-2017-0055.

[42] N. S. Mohamed et al., "Impact factors of orthopaedic journals between 2010 and 2016: trends and comparisons with other surgical specialties," Annals of Translational Medicine, vol. 6, no. 7, pp. 114-114, 2018, doi: 10.21037/atm.2018.03.02.

[43] C. S. Hong and T. G. Oh, "TPR-TNR plot for confusion matrix," Commun Stat Appl Methods, vol. 28, no. 2, pp. 161-169, 2021, doi: 10.29220/CSAM.2021.28.2.161.

[44] A. O. Balogun, A. O. Bajeh, V. A. Orie, and A. W. Yusuf-Asaju, "Software Defect Prediction Using Ensemble Learning: An ANP Based Evaluation Method," FUOYE Journal of Engineering and Technology, vol. 3, no. 2, 2018.

[45] 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.

[46] T. Sharma, S. Bhaskar, A. Jatain, and K. Pabreja, “Library Progress International Optimizing Software Defect Detection using advanced Feature Selection, Ensemble Learning, and Class Imbalance Solutions,” 2024. [Online]. Available: www.bpasjournals.com

AUTHOR BIOGRAPHY

MUHAMAD MICHAEL FEBRIAN is a dedicated student at Universitas Lambung Mangkurat, where he has been pursuing his education in the Department of Computer Science. His primary research interest lies in software development, focusing on optimization techniques and predictive modeling for improving software performance. Muhamad is passionate about exploring innovative solutions in software engineering to address real-world challenges. Beyond academics, he actively participates in collaborative research projects to expand his practical skills and contribute to advancements in his field. Email: michaelfebr5@gmail.com.

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67

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 University in 2011, and received his master's degree in Informatics from STMIK Amikom University in 2016. He is active as an information technology practitioner and consultant, being a project manager or systems analyst working on several projects in government and private agencies in South Kalimantan province since 2017. His research interests include software engineering, human computer interaction, and artificial intelligence applications. He can be contacted at email: setyo.saputro@ulm.ac.id.

TRIANDO HAMONANGAN SARAGIH, currently holding the position of a lecturer within the Department of Computer Science at Lambung Mangkurat University, is heavily immersed in the realm of academia, with a profound focus on the multifaceted domain of Data Science. His academic pursuits commenced with the successful completion of his bachelor's degree in Informatics at the esteemed Brawijaya University, located in the vibrant city of Malang, back in the year 2016. Building upon this foundational achievement, he proceeded to further enhance his scholarly credentials by enrolling in a master's program in Computer Science at Brawijaya University, Malang, culminating in the conferral of his advanced degree in 2018. The research field he is involved in is Data Science. Email: triando.saragih@ulm.ac.id. Orcid ID: 0000-0003-4346-3323.

FRISKA ABADI received his Bachelor's degree in Computer Science from Lambung Mangkurat University, Banjarbaru, Indonesia, in 2011, and his Master's degree in Informatics from STMIK Amikom Yogyakarta in 2016. He is currently a lecturer in the Department of Computer Science, Faculty of Mathematics and Natural Sciences, Lambung Mangkurat University, Banjarbaru, Indonesia. His research interests include data mining and software engineering. He can be contacted at: friska.abadi@ulm.ac.id.

RUDY HERTENO received his bachelor'sdegree in Computer Science from Lambung Mangkurat Universityin 2011. After completing his studies, he worked as a software developer for several years to gain more experience in the field. During this period, he developed various software applications, particularly to support the needs of local governments. In 2017, he obtained a master's degree in Informatics from STMIK Amikom University. Currently, he is a lecturer in the Computer Science program at Lambung Mangkurat University. His research interests include software engineering, software defect prediction, and deep learning, aiming to improve software quality, optimize error detection in systems, and develop artificial intelligence-based solutions.He can be contacted at email: rudy.herteno@ulm.ac.id

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm ac.id, Department of Computer Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v7i2.67