Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics
Homepage: ijeeemi.org; Vol. 7, No. 1, pp. 90-101, January 2025; e-ISSN: 2656-8624.
https://doi.org/10.35882/ijeeemi.v7i1.47

RESEARCH PAPER

OPEN ACCES

Implementation of Copeland Method on Wrapper-Based Feature Selection Using Random Forest for Software Defect Prediction

Agustia Kuspita Aryanti, Rudy Herteno, Fatma Indriani, Radityo Adi Nugroho, and Muliadi
Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia

ABSTRACT

Software Defect Prediction is crucial to ensure software quality. However, high-dimensional data presents significant challenges in predictive modelling, especially identifying the most relevant features to improve model performance. Therefore, efforts are needed to address these issues, and one is to apply feature selection methods. This study introduces a new approach by applying the Copeland ranking method, which aggregates feature weights from multi-wrapper methods, including Recursive Feature Elimination (RFE), Boruta, and Custom Grid Search, using 12 NASA MDP datasets. The study also applies Random Forest classification and evaluates the model using AUC and t-Test. In addition, this study also compares the accuracy and precision values produced by each method. The results consistently show that the Copeland ranking method produces superior results compared to other ranking methods. The average AUC value obtained from the Copeland ranking method is 0.7496, higher than the Majority ranking method with an average AUC of 0.7416 and the Optimal Rank ranking method with an average AUC of 0.7343. These findings confirm that applying the Copeland ranking method in wrapper-based feature selection can enhance classification performance in software defect prediction using Random Forest compared to other ranking methods. The strength of the Copeland method lies in its ability to integrate rankings from various feature selection approaches and identify relevant features. The findings of this research demonstrate the potential of the Copeland ranking method as a reliable tool for ranking features obtained from various wrapper-based feature selection techniques. The implementation of this approach contributes to improved software defect prediction and provides new insights for the development of ranking methods in the future.

PAPER HISTORY

Received January 05, 2025
Revised January 28, 2025
Accepted February 10, 2025
Published February 24, 2025

KEYWORDS

Copeland Method;
Feature Selection;
Random Forest;
Software Defect Prediction;
Wrapper

CONTACT:

Rudy Herteno
Rudy.herteno@ulm.ac.id

1. INTRODUCTION

Currently, numerous industry sectors leverage technology, and software development is highly dependent on software quality. Software testing is essential to ensure that the software being developed is of the highest calibre and can function at its peak. Software testing's primary objective is to assess internal and external software quality using predetermined standards. Before software is used extensively, its quality is maintained by effective testing, which can identify potential flaws or problems [1]. Software is essential, and its reliability guarantees effectiveness and proper operation. Software quality highly depends on the processes applied during development and the success of its testing [2]. Software defects pose potential causes of system failures and damage, making defect detection and prevention essential in software engineering [3]. Conventional solutions to address software defects involve testing and revisions, which require significant time. Software defect prediction aims to identify potential flaws and has become a viable strategy for improving predictions in this field [4]. Early predictions can help optimize resource allocation [5]. The defect prediction process involves analyzing software metrics and then building a model to predict defects. Defect detection in software modules is performed through classification, where software modules are categorized as defective or non-defective, as conducted in several previous studies [6].

The Area Under the Curve (AUC) for Software Defect Prediction (SDP), the subject of this study, is 0.749. The Decision Tree and Naive Bayes classification methods combined with feature selection methods were applied in the 2020 study by Balogun [7]. For instance, when using the Naive Bayes classification, an AUC of 0.746 was achieved. The study by Xiaolong [8] in 2021 implemented ReliefF-based Clustering (RFC), a feature selection algorithm based on clustering, which resulted in an AUC of 0.676. Another study by Nugroho [9] in 2020 applied feature selection techniques and several classification algorithms, with the Decision Tree yielding the highest average AUC of 0.566. These previous studies are the

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

foundation for this research, which implements feature selection techniques and ranking methods to achieve a higher AUC value.

Theoretically, this study demonstrates that wrapper-based feature selection methods can reduce dimensionality and eliminate irrelevant attributes, thereby improving predictive model outcomes. Random Forest, an ensemble learning algorithm designed to enhance prediction performance and robustness, is one of the techniques used for software defect prediction [10]. The Random Forest model excels in prediction compared to models like logistic regression because it can handle the complexity arising from many features in the dataset. In contrast, models like logistic regression often face challenges in managing a high number of features [11]. The Random Forest model performs well, according to studies [12], and additional research also highlights its effectiveness in data extraction, analysis, and prediction [13]. Its reliability in handling data with numerous features makes it a popular choice in various fields, including bioinformatics, anomaly detection, medical diagnostics, and financial analysis. However, Random Forest remains less efficient in detecting all classes and encounters difficulties in processing high-dimensional data, which increases the risk of overfitting and slows down training time. This research employs the Random Forest algorithm and several other methods to enhance prediction. Wrapper-based feature selection techniques such as RFE, Boruta, and Custom Grid Search are used in this study. These techniques can improve the predictive model's performance by progressively identifying the most relevant elements and removing unimportant ones. RFE effectively eliminates less relevant features but is prone to overfitting, which Boruta's conservative approach can address to retain important features. Boruta can identify truly significant features but tends to retain too many, which can be further filtered by RFE and optimized through Custom Grid Search. Custom Grid Search enhances selection efficiency by searching for the best combination of features and hyperparameters, although it does not perform feature selection directly. To improve feature selection performance, this study also utilizes a ranking method in the hope that this approach can help achieve better feature selection. This study applies the Copeland, Majority, and Optimal Rank methods to rank features by aggregating feature weights from multi-wrapper methods and ensuring more optimal feature selection. Therefore, this research aims to address large-scale problems using wrapper-based feature selection methods (RFE, Boruta, and Custom Grid Search) while incorporating several ranking techniques. The combination of techniques used in this study is expected to help overcome challenges in software defect prediction.

2. MATERIALS AND METHOD

FIGURE 1 provides an overview of the proposed study. This study uses the Random Forest algorithm for classification and applies ranking methods to wrapper-based feature selection. The collection of the NASA MDP dataset is the first stage. After data collection, preprocessing is performed, and then the dataset is split 70:30, with 70% for training and 30% for testing. Recursive Feature Elimination (RFE), Boruta, and Custom Grid Search are wrapper-based feature selection methods used in the first stage to identify feature weights. RFE recursively removes the least important features based on the model's performance until an optimal subset of features is obtained. It ranks features by fitting the model multiple times and eliminating the least impactful features. Boruta, on the other hand, is an extension of the Random Forest algorithm that evaluates feature importance by comparing original features with shuffled shadow features, keeping only the most relevant ones. Custom Grid Search is an exhaustive search technique that aims to find the best combination of hyperparameters for feature selection by systematically evaluating different parameter sets to enhance model performance. After feature selection, this study applies three ranking methods. The Copeland method combines feature importance from multiple wrapper methods by scoring features based on their rankings. The Majority method selects features that appear most frequently among top-ranked ones. The Optimal Rank method assigns weights based on the highest observed ranking. After ranking, this study applies the Random Forest classification model.

The model is trained using the selected features, and its performance is evaluated based on the Area Under the Curve (AUC), accuracy, and precision. A statistical significance analysis uses the t-test to assess differences between ranking methods. This evaluation determines the optimal feature ranking method for improving defect prediction performance. This study analyzes and compares various feature ranking approaches to identify the best strategy for enhancing model performance. The model is trained using the selected features, and its performance is evaluated based on the Area Under the Curve (AUC), accuracy, and precision. A statistical significance analysis uses the t-test to assess differences between ranking methods. This evaluation determines the optimal feature ranking method for improving defect prediction performance. This study analyzes and compares various feature ranking approaches to identify the best strategy for enhancing model performance. By conducting a comparative analysis, this research provides insights into the strengths and weaknesses of each

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

process, helping to select the most suitable strategy for feature selection in software defect prediction.

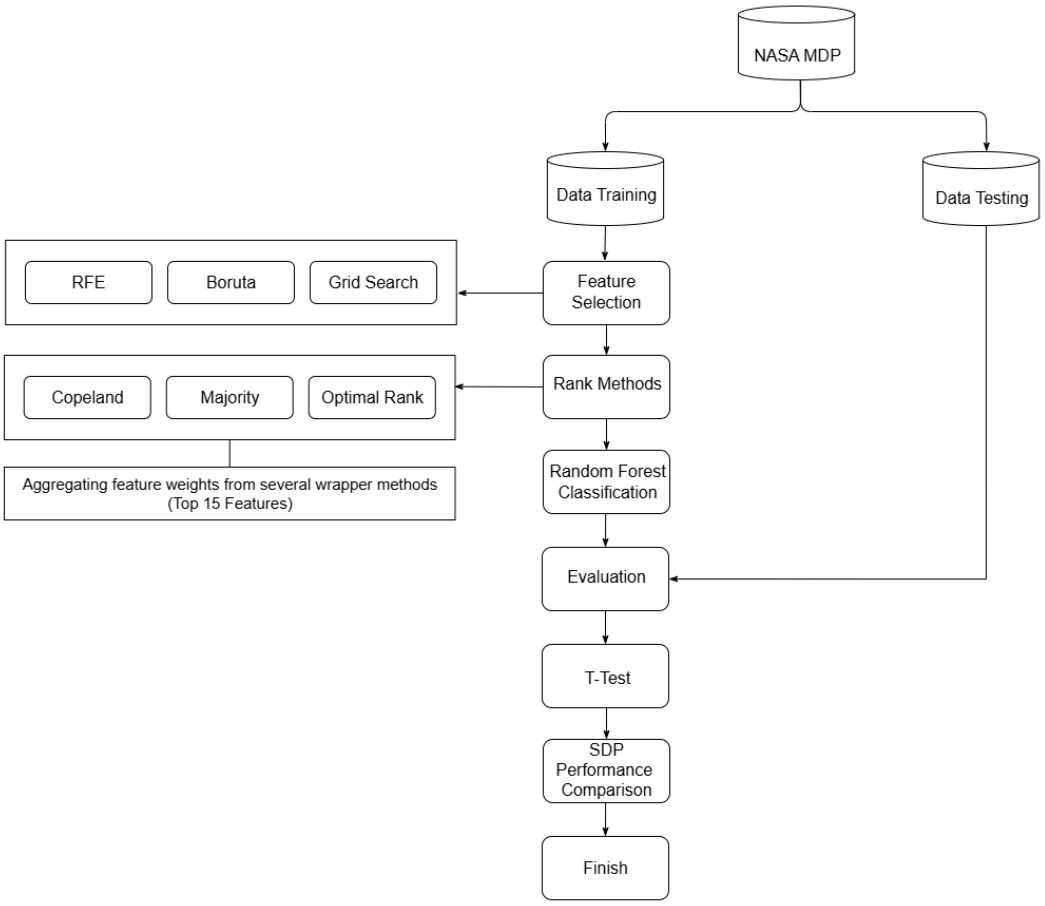

FIGURE 1. Research Method

A. Dataset

This study utilizes 12 datasets from the NASA Metrics Data Program (NASA MDP), including CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. The NASA MDP repository can be accessed via the following link: https://github.com/klainfo/NASADefectDataset. This dataset covers various software projects with different characteristics, making it useful for analyzing defect patterns in diverse development environments. This dataset contains static code metrics for each constituent module, where the term module can refer to a method, function, or procedure within the software [14]. It has been widely used in previous studies focusing on software defect prediction because it includes various amounts of repeated data, such as observations of module metrics and their corresponding defect data [14], making it one of the primary benchmarks for developing defect detection methods. The NASA MDP dataset contains hundreds to thousands of software modules with various features reflecting their characteristics. This dataset is relevant to software defect prediction research as it provides real-world data from large-scale systems and supports the development of machine learning models to identify potentially defective modules.

TABLE 1. NASA MDP Datasets Specifications

Datasets	Attributes	Instances	Non-Defects	Defects	Defective (%)
CM1	38	327	285	42	12.8
JM1	22	7720	6110	1672	21.5
KC1	22	1162	869	314	26.5
KC3	40	194	158	36	18.6
MC1	39	1952	1942	46	2.3
MC2	40	124	81	44	35.2
MW1	38	250	226	27	10.7
PC1	38	679	644	61	8.7
PC2	37	722	729	16	2.1

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

PC3	38	1053	943	134	12.4
PC4	38	1270	1110	177	13.8
PC5	39	1694	1240	471	27.5

B. Preprocessing Data

In order to prepare the data for usage in different Machine Learning models, data preprocessing is an essential stage in the data mining process [15]. This process ensures that the dataset is ready to be processed by algorithms by verifying and correcting missing values and managing factors that could affect the performance of machine learning models more effectively [16]. Label encoding is a step in data preprocessing that attempts to convert string data into a numeric form so that the prediction process can run more effectively [17]. In this study, as seen in the CM1 dataset, the label "Y" was changed to "1" and the label "N" to "0," which can be observed in TABLE 2 and TABLE 3.

TABLE 2. Before Preprocessing

id	LOCK_BLANK	BRANCH_OUT	…	NUMBER_OF_LINES	LOC TOTAL	Defective
1	2	3	…	9	9	N
2	3	3	…	19	13	N
3	38	35	…	218	109	N
4	1	7	…	68	41	Y
…	…	…	…	…	…	…
325	3	3	…	18	12	N
326	6	9	…	61	32	N
327	1	3	…	12	10	N

TABLE 3. After Preprocessing

id	LOCK_BLANK	BRANCH_OUT	…	NUMBER_OF_LINES	LOC TOTAL	Defective
1	2	3	…	9	9	0
2	3	3	…	19	13	0
3	38	35	…	218	109	0
4	1	7	…	68	41	1
…	…	…	…	…	…	…
325	3	3	…	18	12	0
326	6	9	…	61	32	0
327	1	3	…	12	10	0

C. Split Data

The dataset is typically divided into two main parts, namely training data and testing data, to develop and evaluate machine learning models effectively [18]. After the dataset undergoes preprocessing to clean the data and ensure its quality, the next step is the data splitting process. The division uses the Stratified K-Fold Cross-Validation method with k=10 [19]. This method allows the data to be divided into 10 distinct folds, where each fold maintains the same class proportions to ensure balanced class distribution across all parts of the data. This data-splitting process is crucial to producing balanced class representation in the training and testing datasets, enabling the model to learn effectively and provide accurate and reliable evaluation results.

D. Recursive feature Elimination (RFE)

RFE works by progressively removing less important features, one at a time. As a wrapper-based feature selection method, RFE ranks existing features and generates a candidate subset with corresponding accuracy measures [20], helping to reduce redundant information and enhance classification performance [21]. RFE is an iterative technique that uses a specific algorithm to select features recursively. The process filters features in the dataset to assess classification performance [22] and is carried out step by step, removing irrelevant features until only the most significant ones remain [23]. RFE is widely used in research and has shown exceptional results [24]. The RFE process starts with training a model using all available features. The model then assigns weights or importance to each feature, and the feature with the lowest weight, considered least significant, is removed. The model is retrained without that feature, and this process continues until only relevant features remain. The classifier is used as the estimator due to its ability to handle datasets with

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

many features and naturally estimate feature importance. The parameter n_features_to_select=1 ranks all features, allowing the selection of a certain number of top features based on the final ranking.

E. Boruta

The Random Forest classifier served as the foundation for the Boruta technique. This feature selection algorithm was developed to identify the best features [25] and can enhance the performance of classification models [26][27]. The parameters used in Boruta include n_estimators='auto', which allows the optimal number of trees to be determined automatically, and random_state=42 to ensure consistent results. Boruta is chosen because it can distinguish truly significant features from those that do not influence the target. Boruta works by adding shadow features, which are created by shuffling the original values, to evaluate the importance of each feature in the dataset. The selection process is performed iteratively by removing less essential features than shadow ones, ensuring that only genuinely relevant features are retained. This technique is widely used in various domains, including bioinformatics, finance, and big data analysis, due to its reliability in handling datasets with numerous features and its ability to enhance model interpretability.

F. Custom Grid Search

Custom Grid Search is designed to optimize the search for the best parameters by automating the often time-consuming trial-and-error process. It allows the selection of the best parameters from a predefined list of alternatives [28]. In the study [29], Custom Grid Search was applied to determine the most optimal parameters. Custom Grid Search with Grid Search CV (Cross-Validation) using cv=5 (5-fold cross-validation) is employed to optimally tune the Random Forest parameters by testing various combinations, such as n_estimators, max_depth, min_samples_split, and min_samples_leaf. The 5-fold cross-validation technique ensures that each parameter combination is thoroughly tested by splitting the data into five parts for alternating training and testing, thereby preventing overfitting and ensuring the model performs well on new data.

G. Copeland

To order the features of a software flaw, the Copeland technique uses the pairwise comparison principle to assess each feature's degree of relevance. The outcomes of the feature selection procedure are combined using the Copeland approach ion performance [30]. The Copeland approach ranks features by aggregating feature weights from multi-wrapper methods, evaluating the difference between one alternative's wins and losses relative to others. The initial step in this method involves constructing a matrix, which is then used to perform pairwise comparisons for each method [31]. Features that rank higher more frequently in each pair earn victory points, while lower-ranked features receive loss points. The final score is determined by subtracting total loss points from total victory points, with the highest-scoring feature considered the most significant (ALGORITHM 1).

H. Random Forest

Random Forest is widely used across various fields due to its super [32]. Several studies have shown that applying the Random Forest method can yield exceptionally high performance. According to the study [33], the Random Forest model consists of many independent decision trees, which provide high resilience to external influences [34]. In this classification, the parameters used are n_estimators = [50, 100, 200], max_depth = [None, 10, 20, 30], min_samples_split = [2, 5, 10], and min_samples_leaf = [1, 2, 4] to find the best combination. This technique ensures that the model is evaluated on various data subsets, prevents overfitting, and selects the optimal parameters to improve accuracy and generalization capability. Additionally, Random Forest can handle high-dimensional data without experiencing significant performance degradation. This algorithm also automatically manages missing values and maintains prediction stability even under varying data conditions.

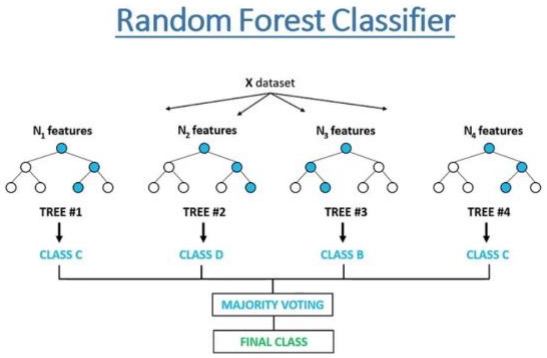

FIGURE 2. Classification Flow of Random Forest Algorithm [34]

I. Evaluation

The Area Under the Curve (AUC) statistic is used in this study to evaluate the prediction model's performance. The selection of an appropriate evaluation technique is crucial for accurately assessing the performance of machine learning models [35]. AUC is used as the primary indicator, while other indicators, such as accuracy and precision, are utilized to evaluate the model's effectiveness further. Software is essential for evaluating the strengths and limitations of the model [36][37]. The AUC statistic is used in this study due to its ability to handle class imbalance in software defect datasets. AUC provides an overview of classification performance [38] by considering the false positive rate and sensitivity, where a higher value indicates a better-quality algorithm [39][40]. In contrast, the t-test evaluates the significance of performance differences between models. This test helps

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

determine whether the differences in results between models are statistically significant. Additionally, it provides further insights into the model's reliability and assists in selecting the most effective method.

ALGORITHM 1. Pseucode of the Proposed Method

```
Begin
  NasaData = Datasets.GetNasaMDP()
  WeightsRFE = FeatureSelection.RFE(RandomForestClassifier(), NasaData)
  WeightsBoruta = FeatureSelection.Boruta(RandomForestClassifier(), NasaData)
  WeightsGridSearch = FeatureSelection.GridSearchCV(RandomForestClassifier(random_state=42), NasaData)
  RankedFeatures = copeland_rank = rk.rank(rk.pairwise(pd.DataFrame({"RFE": WeightsRFE, "Boruta":
  WeightsBoruta, "GridSearch": WeightsGridSearch}))).sort_values(ascending=True)
  SelectedFeatures = RankedFeatures.index[:15]
  FeatureWeights = {feature: weight for feature, weight in zip(X.columns, BestModel.feature_importances_)}
  RankedFeatures = rk.rank(rk.pairwise(pd.DataFrame({'Feature': FeatureWeights.keys(),
  'Weight':FeatureWeights.values()}).set_index('Feature'))).sort_values(ascending=True).index[:15]
  FinalModel = RandomForestClassifier(n_estimators=100, random_state=42).fit(X_train, y_train)
  y_pred = FinalModel.predict_proba(X_test)[:, 1]
  Performance = Evaluation.AUC(FinalModel, Test)
End
```

3. RESULTS

There are two primary phases to this investigation. The Random Forest algorithm with wrapper-based feature selection, which includes the RFE, Boruta, and Custom Grid Search methods, is used in the first stage. The Copeland method is then used for ranking. The second stage also uses the Random Forest algorithm with wrapper-based feature selection using the same techniques, but the features are ranked using the Majority and Optimal Rank methods. This study aims to determine the AUC value on the NASA MDP dataset using Random Forest classification by applying the Copeland method, which aggregates feature weights from various wrapper methods.

This study utilizes the t-Test to determine the significance level and evaluate the results after applying various techniques to the dataset analysis. The first phase of the research involves the application of RFE, Boruta, and Custom Grid Search, aimed at improving model performance by selecting the most relevant features. RFE retrains the model using the remaining features until the ideal number of features is obtained, after removing the least significant features based on their relevance score, as determined by the base model, such as Random Forest (RF). Feature ranking is then performed using the Copeland, Majority, and Optimal Rank techniques by aggregating feature weights from multiple wrapper methods after identifying key features using RFE, Boruta, and Custom Grid Search.

This study compares three primary methods. The first approach uses a Random Forest with wrapper-based feature selection, including Boruta, Custom Grid Search, and Recursive Feature Elimination (RFE), and ranks the features using the Copeland method. The second approach uses the same feature selection and classification but ranks the features using the Majority method. Meanwhile, the third approach also uses the same feature selection and classification; however, feature ranking is performed using the Optimal Rank method. TABLE 4 compares the AUC values generated by the three applied methods, with each AUC value displayed in a separate column. The first column shows the dataset used, and the AUC values obtained from the combination of wrapper-based feature selection with Random Forest classification using the Copeland ranking method are presented in the second column. The AUC values obtained from wrapper-based feature selection with Random Forest classification using the Majority and Optimal Rank ranking methods are shown in the third and fourth columns, respectively. The last row presents the average values obtained from each technique applied. These results provide a clear overview of how each method performs across the different datasets used in this study. The conclusions drawn from this table will serve as a starting point for further research on the best approach for predicting software defects. The results presented in the table show the AUC values across the different datasets used. Some datasets, such as PC5, show higher AUC values than others, indicating that the Copeland ranking method is more effective for specific datasets. On the other hand, datasets like KC1 have AUC values comparable to other methods, where the AUC values are not higher, which may be due to the complexity of the data or a feature distribution that does not sufficiently support the model's performance.

These performance differences can be attributed to several factors, including class imbalance levels within the dataset and the complexity of the relationship patterns between features and target labels. Therefore, further analysis is needed to understand the specific characteristics of each dataset that influence the performance of the applied methods. This table will serve as a starting point for future research to identify the best approach for software defect prediction. Additionally,

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

evaluating model performance across various testing scenarios can help determine more effective strategies.

TABLE 4 AUC Performance of Every Experiment

Dataset	Wrapper + Copeland + RF	Wrapper + Majority + RF	Wrapper + Optimal Rank + RF
CM1	**0.73193**	0.72478	0.72731
JM1	**0.66215**	0.66215	0.66215
KC1	**0.69309**	0.69309	0.69309
KC3	**0.66220**	0.66220	0.64966
MC1	**0.85671**	0.85671	0.77839
MC2	**0.69583**	0.69583	0.69583
MW1	**0.71543**	0.69239	0.65552
PC1	**0.82690**	0.80896	0.82690
PC2	**0.71990**	0.67679	0.71990
PC3	**0.76694**	0.76694	0.76694
PC4	**0.90798**	0.90798	0.88489
PC5	**0.75679**	0.75167	0.75194
Average	**0.74965**	0.74162	0.73438

FIGURE 3 compares AUC values obtained from wrapper-based feature selection and classification using Random Forest, applying three ranking methods: Copeland, Majority, and Optimal Rank. In this chart, the vertical axis represents the AUC values, while the horizontal axis displays the various datasets used in this study. Each line on the vertical axis represents the AUC value obtained from each ranking method. Therefore, this graph provides a clear overview of the performance of each method across various datasets and facilitates the comparison of the effectiveness of the three methods in producing prediction model performance. The graph also illustrates that some datasets produce more consistent AUC values across all methods while others exhibit more significant fluctuations. These variations may indicate that certain datasets are more sensitive to specific ranking methods or that the effects of feature selection techniques vary depending on the dataset's characteristics. Therefore, understanding this variability is crucial for developing more adaptive feature selection and ranking strategies to enhance the reliability of software defect prediction.

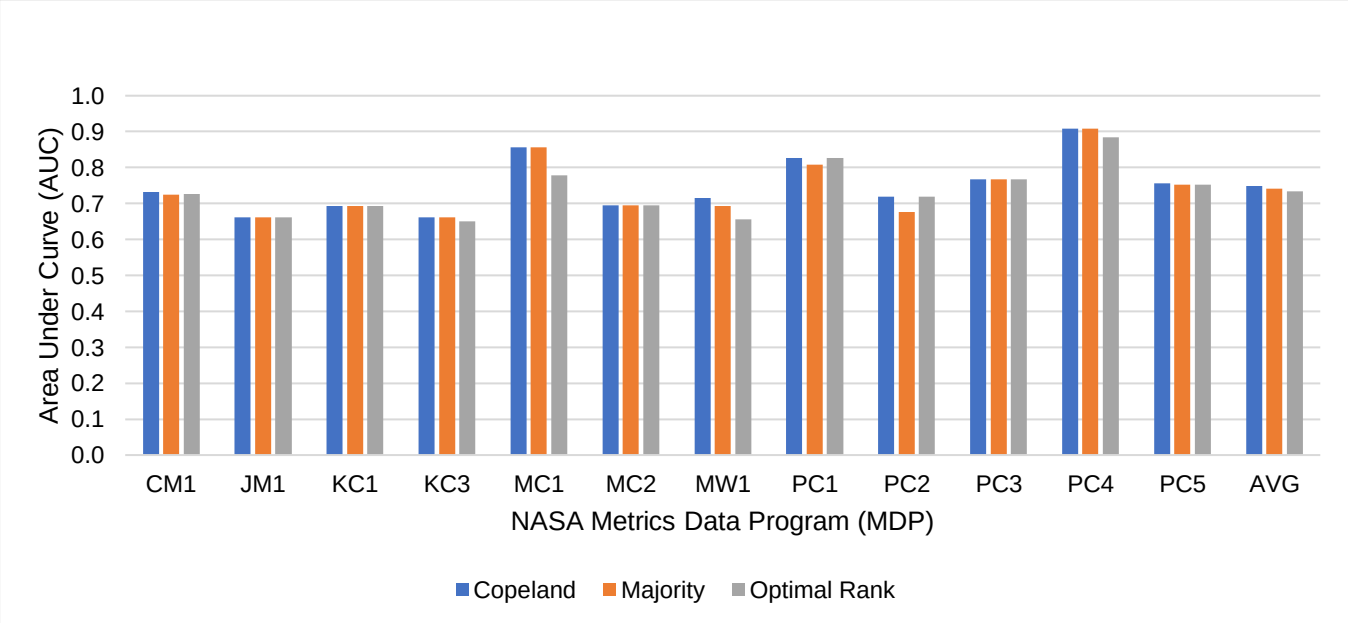

FIGURE 3. Comparison of AUC for Every Experiment

FIGURE 4 compares the average AUC values obtained from three feature ranking methods: Copeland, Majority, and Optimal Rank. The results shown in the graph indicate that the Copeland method achieves the highest average AUC 0.74965, followed by Majority 0.74162 and Optimal Rank 0.73438. This difference suggests that the Copeland approach is more effective in improving the performance of software defect prediction models than the other methods. The relatively small difference in AUC indicates that each approach performs competitively.

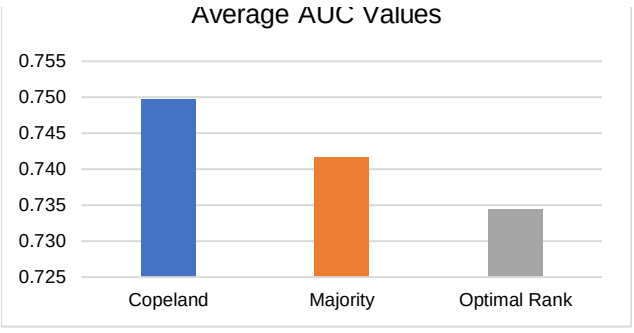

FIGURE 4. Average AUC Values

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

However, Copeland remains the most superior in improving the accuracy of software defect prediction.

FIGURE 5 presents a graph comparing the average accuracy values across different ranking methods. According to the results, the Copeland method achieves the highest accuracy, with an average value of 0.79145, followed by the Majority method, which attains an accuracy of 0.78478. These findings indicate that the Copeland approach provides better performance in software defect prediction compared to other methods. Additionally, the small accuracy difference suggests that

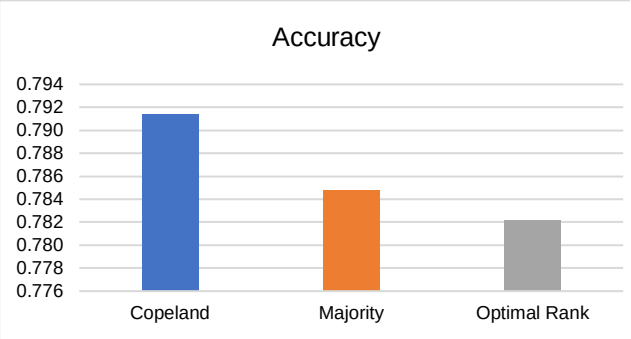

FIGURE 5. Average Accuracy Values

both methods perform competitively, although Copeland maintains an advantage.

FIGURE 6 compares the average precision values for various tested methods. The graph indicates that the Copeland method achieves the highest average precision, with a value of 0.43448, followed by the Majority method at 0.42571 and the Optimal Rank method at 0.41603. These results suggest that the Copeland method

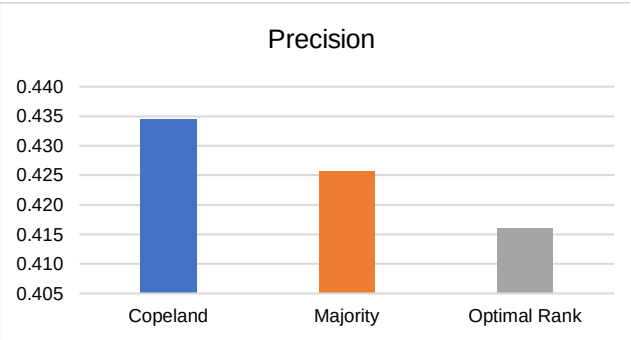

FIGURE 6. Average Precision Values

is superior in improving precision compared to other ranking methods.

TABLE 5 presents the t-Test results, which compare the performance quality of the proposed approach—Random Forest classification and wrapper-based feature selection with Copeland ranking—with other ranking techniques such as Majority and Optimal Rank. This test uses a significance level (α) of 0.05 to evaluate performance differences between the compared strategies. The t-Test used in this study is an independent sample t-test, which aims to determine whether a significant difference exists between the two groups of models being compared. This test has several fundamental assumptions, including the normality of the sample distribution, equality of variances (homogeneity of variances) between groups, and sample independence. The Kolmogorov-Smirnov or Shapiro-Wilk test can be used to ensure that the normality assumption is met. If the assumption of equal variances is not satisfied, heterogeneity can be addressed using an adjusted t-test (Welch's t-test). In this study, the t-test is used to validate whether the differences between the compared ranking techniques are not merely the result of random variations in the data. With a significance level of 0.05, this test provides statistical evidence that the Copeland ranking method is significantly better than other methods. The test results analysis indicates that the performance differences between the compared methods did not occur by chance but were influenced by the advantages of the approach used.

TABLE 5. The T-Test Results for Every Experiment

Method Comparison	t-Test Value (α = 0.05)	Significance
Copeland - Majority	**0.032194**	**Significant**
Copeland - Optimal Rank	**0.035042**	**Significant**

TABLE 6 compares the results of this study with previous research, highlighting the advantages of the proposed approach in improving the accuracy of software defect prediction. The method used in this study, namely Random Forest with wrapper-based feature selection, including RFE, Boruta, Custom Grid Search, and the Copeland ranking method, achieves the highest AUC value compared to other methods applied in previous studies. The integration of wrapper-based feature selection techniques with the Copeland ranking method has significantly enhanced the model's ability to distinguish between defective and non-defective software modules. The strength of this approach lies in the combination of feature selection and ranking strategies, which optimally capture patterns within the data. With more accurate results, this method can serve as a foundation for developing more reliable software defect prediction models.

TABLE 6. Comparison with Related Works

Source	Method	AUC
[7]	NB + Rank Aggregation-Based Multi-Filter	0.746
[8]	J48 Classifier + CFS	0.676
[9]	DT + Metrics Based Feature Selection	0.566
Our	RF + RFE, Boruta, Grid Search + Copeland	0.749

4. DISCUSSION

The study's TABLE 4 and FIGURE 3 findings show the AUC values for the Copeland, Majority, and Optimal Rank ranking techniques. Specifically, applying the Copeland

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

method demonstrates a higher average AUC than other ranking methods, namely Majority and Optimal Rank. This table compares the AUC values between Random Forest combined with the feature selection methods RFE, Boruta, and Custom Grid Search using the Copeland ranking method and Random Forest with the Majority and Optimal Rank ranking methods across various datasets. For example, in the CM1 dataset, the Copeland method achieved an AUC value of 0.73193, outperforming the Majority method with an AUC of 0.72478 and the Optimal Rank method with an AUC of 0.72731. Similarly, in the MW1 dataset, the Copeland method produced an AUC value of 0.71543, higher than the Majority method with an AUC of 0.69239 and the Optimal Rank method with an AUC of 0.65552. Additionally, in the PC5 dataset, the Copeland method demonstrated superior performance with an AUC value of 0.75679, compared to the Majority method with an AUC of 0.75167 and the Optimal Rank method with an AUC of 0.75194. However, in some datasets, the Copeland method produced the same AUC value as other methods, such as in the JM1 dataset with an AUC of 0.66215 and the KC1 dataset with an AUC of 0.69309. In these cases, the Copeland method did not achieve a higher AUC value, but this does not mean that the AUC produced by the Copeland method was lower than other methods. Overall, the Copeland method remains superior in improving AUC performance compared to traditional methods due to its ability to comprehensively integrate results from various feature selection techniques. This lower performance is likely due to specific dataset characteristics, such as inconsistent feature distribution or the presence of less relevant features in distinguishing defective and non-defective modules. Copeland contributes to the feature ranking process compared to Majority and Optimal Rank, as it uses pairwise comparison to combine the strengths of each feature selection technique. This approach allows the Copeland method to identify important features in distinguishing defective and non-defective software modules more effectively. With this approach, Copeland reduces bias in single-ranking methods and enhances model generalization across different datasets. Overall, the Copeland method consistently delivers superior performance in terms of AUC and the ability to handle high-dimensional data compared to other ranking methods.

Apart from AUC, based on the research findings presented in FIGURE 5 and FIGURE 6, the Copeland method has demonstrated superior performance compared to other ranking methods, namely Majority and Optimal Rank. This superiority is evident from the higher average accuracy and precision values. As shown in the table, the Copeland method achieved an accuracy of 0.79145, higher than the Majority method 0.78478 and the Optimal Rank method 0.78214. Additionally, the precision value obtained using the Copeland method outperformed the others, with a value of 0.43448, compared to the Majority of 0.42571 and Optimal Rank of 0.41603. The superiority of the Copeland method in this ranking demonstrates that a ranking aggregation-based approach can provide more consistent and accurate results than other methods. The superior performance of the Copeland method is due to its ability to efficiently combine information from various ranking sources, resulting in more stable and reliable decisions. Furthermore, the differences in accuracy and precision values indicate that this method can reduce ranking errors and improve the accuracy of identifying the appropriate categories.

Based on the t-Test results presented in TABLE 5, important information is found in the second and third columns. The second column shows the significant quality improvement of the methods combined with wrapper-based feature selection and the ranking methods proposed in this study. Meanwhile, the third column shows the significance level of the results. If the obtained AUC value is smaller than the alpha value of 0.050, the performance improvement can be considered significant. Conversely, performance improvement is insignificant if the AUC if the AUC value is larger than alpha. In comparing wrapper-based feature selection methods, namely RFE, Boruta, and Custom Grid Search, ranked using the Copeland method proposed in this study with Random Forest, the obtained value was 0.032194. This value is lower than the t-Test Value with a significance level 0.05, indicating that the performance quality improvement is significant. Furthermore, comparing the Copeland and Optimal Rank methods resulted in a value of 0.035042, indicating a significant performance improvement. These results further strengthen the fact that the Copeland method has significant potential as a practical approach to improving the quality of software defect prediction models. Overall, the performance results show significance in all comparisons, proving that the proposed Copeland ranking method outperforms other methods with quality performance improvement reflected by better AUC values.

Although this study provides valuable insights, several limitations must be acknowledged. This study only used the NASA MDP dataset, which may limit the generalization of the results to other datasets with different characteristics. This research has successfully demonstrated the effectiveness of combining wrapper-based feature selection methods such as Recursive Feature Elimination (RFE), Boruta, and Custom Grid Search, along with ranking methods like Copeland, Majority, and Optimal Rank in improving software defect prediction using the Random Forest algorithm. However, further development is still needed to improve the results. Additionally, feature selection techniques such as RFE, Boruta, and Custom Grid Search can introduce bias in the process, especially if the dataset has specific characteristics that affect model performance. Other factors, such as the selection of model parameters influencing the complexity and performance of the algorithm, can also affect the effectiveness of the approach used in the software defect prediction process, as suboptimal parameter choices may lead to underfitting or overfitting.

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

The combination of wrapper-based feature selection methods, ranking method, and the Random Forest classification algorithm was chosen based on its advantages in optimizing the feature selection, ranking, and prediction processes. Previous studies [41] also support the effectiveness of wrapper-based feature selection combined with Random Forest for software defect prediction, leading to high classification accuracy. RFE was used for its strength in iterative approaches that remove less relevant features while retaining the most significant ones. Boruta complements this process by ensuring that important features remain identified through comparisons with random data, increasing confidence in the election results. Custom Grid Search provides high flexibility in finding optimal parameters that are more adaptive to the dataset's characteristics. The Copeland ranking method was chosen because it allows the integration of results from various feature selection methods, resulting in more robust decisions while minimizing bias. Meanwhile, Random Forest as a classification algorithm excels in handling high-dimensional datasets, offering resilience to overfitting and demonstrating consistent performance in classification. This study's results indicate that applying of the Copeland method yields a higher average AUC compared to other ranking methods. Additionally, the Copeland method also achieves higher average accuracy and precision values than the other methods evaluated, further reinforcing the superiority of this approach. The key findings of this study indicate that the Copeland ranking method consistently achieves the highest AUC values compared to Majority and Optimal Rank, making it the best choice in this research. These findings highlight the importance of selecting a ranking method that aligns with the data structure and the potential for further development to enhance software defect prediction.

5. CONCLUSION

This study demonstrates that, compared to other ranking techniques, the performance of the Random Forest algorithm can be improved using the Copeland ranking approach combined with wrapperbased feature selection. Random Forest classification faces challenges such as highdimensional data and irrelevant features. Therefore, this study applies wrapper-based feature selection methods, including RFE, Boruta, and Custom Grid Search, which are then ranked using the Copeland, Majority, and Optimal Rank methods before being implemented in the Random Forest classification algorithm. Applying this approach leads to the conclusion that the Copeland method is superior to other ranking methods. The comparison of the average AUC values between the Copeland ranking method and other ranking techniques shows that the Copeland method delivers the best results, with an AUC value of 0.7496. The superiority of the Copeland method is also evident from the higher average accuracy and precision values, where the Copeland method achieved an accuracy of 0.79145, higher than the Majority method 0.78478 and the Optimal Rank method 0.78214. Additionally, the precision value obtained using the Copeland method outperformed the other methods, with a value of 0.43448, compared to the Majority method 0.42571 and the Optimal Rank method 0.41603. Meanwhile, the Majority method achieves an average AUC of 0.7416, and the Optimal Rank method has an average AUC of 0.7343. Regarding the t-test results, the wrapper-based feature selection methods, including RFE, Boruta, and Custom Grid Search, ranked using the Copeland method as the proposed approach compared to the Majority method, yield a value of 0.032194. This value is smaller than the t-Test significance level of 0.05, indicating a significant improvement in performance quality. Additionally, comparing the Copeland and Optimal Rank methods results in a t-Test value of 0.035042, demonstrating significant performance improvement. These findings further reinforce that the Copeland method has great potential as a practical approach to enhancing the quality of software defect prediction models compared to other ranking techniques. Overall, the performance analysis results indicate significance across all comparisons, proving that the proposed approach using the Copeland ranking method outperforms other methods. This approach can enhance performance and quality significantly while addressing existing challenges. However, the methods used in this study have certain limitations that must be acknowledged. Feature selection techniques such as RFE, Boruta, and Custom Grid Search may introduce bias in the feature selection process, especially if the dataset has specific characteristics that affect model performance. Additionally, although Random Forest is known for its resilience to overfitting, it still has the potential to overfit, particularly in high-dimensional spaces, if the number of trees is not optimally configured or the training data is not sufficiently representative.

For future research, given that the Copeland ranking method has demonstrated superior performance, further studies could explore how this method can be adapted or integrated with more advanced techniques, such as deep learning-based feature selection, as well as develop more diverse feature selection method combinations through a hybrid approach. Additionally, further research on new ranking methods or the development of innovative approaches is necessary to capture more complex and deeper relationships between features. Using algorithms other than Random Forest, including modern or ensemble-based methods, could provide new perspectives by comparing classification performance more comprehensively. In addition to method exploration, further evaluation of various datasets with different characteristics is also essential to understand the effectiveness of this approach in different scenarios. Thus, future research can contribute to developing more accurate and adaptive software defect prediction systems, enabling effective implementation across various industrial environments.

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

6. ACKNOWLEDGMENT

We extend our gratitude to everyone involved in the Computer Science program, Faculty of Mathematics and Natural Sciences, Lambung Mangkurat University. We also sincerely appreciate all project team members' dedication and contributions, which have significantly impacted us. We highly appreciate the feedback and suggestions, which have been extremely helpful in refining this research.

REFERENCES

[1] Y. Xue, C. Zhang, F. Neri, M. Gabbouj, and Y. Zhang, "An external attention-based feature ranker for large-scale feature selection," *Knowl Based Syst*, vol. 281, Dec. 2023, doi: 10.1016/j.knosys.2023.111084.

[2] T. Sharma, A. Jatain, S. Bhaskar, and K. Pabreja, "Ensemble Machine Learning Paradigms in Software Defect Prediction," in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 199–209. doi: 10.1016/j.procs.2023.01.002.

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

[4] G. Giray, K. E. Bennin, Ö. Köksal, Ö. Babur, and B. Tekinerdogan, "On the use of deep learning in software defect prediction," *Journal of Systems and Software*, vol. 195, Jan. 2023, doi: 10.1016/j.jss.2022.111537.

[5] S. Pandey and K. Kumar, "Software Fault Prediction for Imbalanced Data: A Survey on Recent Developments," in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 1815–1824. doi: 10.1016/j.procs.2023.01.159.

[6] G. Czibula, I. G. Chelaru, I. G. Czibula, and A. J. Molnar, "An unsupervised learning-based methodology for uncovering behavioural patterns for specific types of software defects," in *Procedia Computer Science*, Elsevier B.V., 2023, pp. 2644–2653. doi: 10.1016/j.procs.2023.10.256.

[7] 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.

[8] X. Xu, W. Chen, and X. Wang, "RFC: A feature selection algorithm for software defect prediction," *Journal of Systems Engineering and Electronics*, vol. 32, no. 2, pp. 389–398, Apr. 2021, doi: 10.23919/JSEE.2021.000032.

[9] R. A. Nugroho, F. Abadi, M. R. Faisal, R. Herteno, and R. Ramadhani, "Metrics Based Feature Selection for Software Defect Prediction," *jurnal komputasi*, vol. 8, no. 2, Oct. 2020, doi: 10.23960/komputasi.v8i2.2670.

[10] F. Özen, "Random forest regression for prediction of Covid-19 daily cases and deaths in Turkey," *Heliyon*, vol. 10, no. 4, Feb. 2024, doi: 10.1016/j.heliyon.2024.e25746.

[11] A. Sanjurjo-de-No, A. M. Pérez-Zuriaga, and A. García, "Analysis and prediction of injury severity in single micromobility crashes with Random Forest," *Heliyon*, vol. 9, no. 12, Dec. 2023, doi: 10.1016/j.heliyon.2023.e23062.

[12] H. Yu *et al.*, "Prediction of myofascial pelvic pain syndrome based on random forest model," *Heliyon*, vol. 10, no. 11, Jun. 2024, doi: 10.1016/j.heliyon.2024.e31928.

[13] C. M. Awais, W. Gu, G. Dlamini, Z. Kholmatova, and G. Succi, "A Meta-analytical Comparison of Naive Bayes and Random Forest for Software Defect Prediction," *Intelligent Systems Design and Applications. ISDA 2022. Lecture Notes in Networks and Systems, vol 716*, Jun. 2023.

[14] H. Alsghaier and M. Akour, "Software fault prediction using particle swarm algorithm with genetic algorithm and support vector machine classifier," *Softw Pract Exp*, vol. 50, no. 4, pp. 407–427, Apr. 2020, doi: 10.1002/spe.2784.

[15] A. Ghavidel, P. Pazos, R. del A. Suarez, and A. Atashi, "Predicting the Need for Cardiovascular Surgery: A Comparative Study of Machine Learning Models," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 6, no. 2, pp. 92–106, Apr. 2024, doi: 10.35882/jeeemi.v6i2.359.

[16] Y. F. Zamzam, T. H. Saragih, R. Herteno, Muliadi, D. T. Nugrahadi, and P. H. Huynh, "Comparison of CatBoost and Random Forest Methods for Lung Cancer Classification using Hyperparameter Tuning Bayesian Optimization-based," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 6, no. 2, pp. 125–136, Apr. 2024, doi: 10.35882/jeeemi.v6i2.382.

[17] K. Marzuki, L. Ganda Rady Putra, H. Hairani, L. Zazuli Azhar Mardedi, and J. Ximenes Guterres, "Performance Improvement of The Random Forest Method Based on Smote-Tomek Link on Lombok Tourism Analysis Sentiment," *Jurnal Bumigora Information Technology (BITe)*, vol. 5, no. 2, pp. 151–158, 2023, doi: 10.30812/bite/v5i1.3166.

[18] V. R. Joseph and A. Vakayil, "SPlit: An Optimal Method for Data Splitting," *Technometrics*, vol. 64, no. 2, pp. 166–176, 2022, doi: 10.1080/00401706.2021.1921037.

[19] H. Ghinaya, R. Herteno, M. R. Faisal, A. Farmadi, and F. Indriani, "Analysis of Important Features in Software Defect Prediction Using Synthetic Minority Oversampling Techniques (SMOTE), Recursive Feature Elimination (RFE) and Random Forest," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 6, no. 3, pp. 276–288, May 2024, doi: 10.35882/jeeemi.v6i3.453.

[20] Q. Chen, Z. Meng, X. Liu, Q. Jin, and R. Su, "Decision variants for the automatic determination of optimal feature subset in RF-RFE," *Genes (Basel)*, vol. 9, no. 6, Jun. 2018, doi: 10.3390/genes9060301.

[21] X. Lin, C. Ren, Y. Li, W. Yue, J. Liang, and A. Yin, "Eucalyptus Plantation Area Extraction Based on SLPSO-RFE Feature Selection and Multi-Temporal Sentinel-1/2 Data," *Forests*, vol. 14, no. 9, Sep. 2023, doi: 10.3390/f14091864.

[22] S. Age Abdulkareem and Z. Olorunbukademi Abdulkareem, "An Evaluation of the Wisconsin Breast Cancer Dataset using Ensemble Classifiers and RFE Feature Selection Technique," *International Journal of Sciences: Basic and Applied Research (IJSBAR) International Journal of Sciences: Basic and Applied Research*, vol. 55, no. 2, pp. 67–80, 2021.

[23] B. Sani Mahmoud and A. Baita Garko, "A Machine Learning Model for Malware Detection Using Recursive Feature Elimination (RFE) For Feature Selection and Ensemble Technique," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 24, no. 1, pp. 23–30, 2022, doi: 10.9790/0661-2401032330.

[24] B. Zhang, X. Dong, Y. Hu, X. Jiang, and G. Li, "Classification and prediction of spinal disease based on the SMOTE-RFEXGBoost model," *PeerJ Comput Sci*, vol. 9, pp. 1–20, 2023, doi: 10.7717/PEERJ-CS.1280.

[25] A. A. Heidari, M. Akhoondzadeh, and H. Chen, "A Wavelet PM2.5 Prediction System Using Optimized Kernel Extreme Learning with Boruta-XGBoost Feature Selection," *Mathematics*, vol. 10, no. 19, Oct. 2022, doi: 10.3390/math10193566.

[26] S. Subbiah, K. S. Muthu Anbananthen, S. Thangaraj, S. Kannan, and D. Chelliah, "Intrusion Detection Technique in Wireless Sensor Network using Grid Search Random Forest with Boruta Feature Selection Algorithm," *Journal of Communications and Networks*, vol. 24, no. 2, pp. 264–273, Apr. 2022, doi: 10.23919/JCN.2022.000002.

[27] R. Febrian and A. Mudya Yolanda, "Comparison of Recursive Feature Elimination and Boruta as Feature Selection in Greenhouse Gas Emission Data Classification," 2024.

[28] S. M. Malakouti, M. B. Menhaj, and A. A. Suratgar, "The usage of 10-fold cross-validation and grid search to enhance ML methods performance in solar farm power generation prediction," *Clean Eng Technol*, vol. 15, Aug. 2023, doi: 10.1016/j.clet.2023.100664.

[29] P. Misra, A. Yadav, and A. Singh Yadav, "Improving the Classification Accuracy using Recursive Feature Elimination with

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

Cross-Validation," *International Journal on Emerging Technologies*, vol. 11, no. 3, pp. 659–665, 2020.

[30] A. Özdağoğlu, M. Kemal Keleş, A. Altinata, and A. Ulutaş, "COMBINING DIFFERENT MCDM METHODS WITH THE COPELAND METHOD: AN INVESTIGATION ON MOTORCYCLE SELECTION." doi: 10.5937/jpmnt9-34120.

[31] M. Polatgil and A. Güler, "The use of Different Criteria Weighting and Multi-Criteria Decision Making Methods for University Ranking: Two-Layer Copeland," *Üniversite Araştırmaları Dergisi*, vol. 7, no. 1, pp. 60–73, Mar. 2024, doi: 10.32329/uad.1398302.

[32] 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.

[33] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," *Stata Journal*, vol. 20, no. 1, pp. 3–29, Mar. 2020, doi: 10.1177/1536867X20909688.

[34] M. F. Banjar, I. Irawati, F. Umar, and L. N. Hayati, "Analysis of Stroke Classification Using Random Forest Method," *ILKOM Jurnal Ilmiah*, vol. 14, no. 3, pp. 186–193, Dec. 2022, doi: 10.33096/ilkom.v14i3.1252.186-193.

[35] A. Alqarni and H. Aljamaan, "Leveraging Ensemble Learning with Generative Adversarial Networks for Imbalanced Software Defects Prediction," *Applied Sciences (Switzerland)*, vol. 13, no. 24, Dec. 2023, doi: 10.3390/app132413319.

[36] M. Tsutsuura *et al.*, "The monitoring of vancomycin: a systematic review and meta-analyses of area under the concentration-time curve-guided dosing and trough-guided dosing," *BMC Infect Dis*, vol. 21, no. 1, Dec. 2021, doi: 10.1186/s12879-021-05858-6.

[37] S. A. Ajagbe, K. A. Amuda, M. A. Oladipupo, O. F. AFE, and K. I. Okesola, "Multi-classification of alzheimer disease on magnetic resonance images (MRI) using deep convolutional neural network (DCNN) approaches," *International Journal of Advanced Computer Research*, vol. 11, no. 53, pp. 51–60, Mar. 2021, doi: 10.19101/ijacr.2021.1152001.

[38] A. D. Putri, F. Sholekhah, E. Dadynata, L. Efrizoni, R. Rahmaddeni, and N. Sapina, "Penerapan Algoritma Decesion Tree C4.5 untuk Memprediksi Tingkat Kelangsungan Hidup Pasien Kanker Tiroid," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 4, pp. 1485–1495, Sep. 2024, doi: 10.57152/malcom.v4i4.1532.

[39] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, "Finding the Best Classification Threshold in Imbalanced Classification," *Big Data Research*, vol. 5, pp. 2–8, Sep. 2016, doi: 10.1016/j.bdr.2015.12.001.

[40] A. M. Akbar, R. Herteno, S. W. Saputro, M. R. Faisal, and R. A. 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, Apr. 2024, doi: 10.35882/jeeemi.v6i2.388.

[41] A. Rifa'i, J. Suntoro, and G. G. Setiaji, "GA-SVM Wrapper Feature Selection untuk Penanganan Data Berdimensi Tinggi," *Jurnal Transformatika*, vol. 21, no. 2, p. 64, Mar. 2024, doi: 10.26623/transformatika.v21i2.8886.

AUTHORS BIOGRAPHY

Agustia Kuspita Aryanti is currently a bachelor's degree student in Computer Science from Lambung Mangkurat University. She was actively involved in organizations and student activities at the faculty and university levels. She has an interest in software engineering. With an academic background in technology, she continues to expand her knowledge of software analysis and modeling. In addition, she is also interested in UI/UX design and graphic design, which complement her technical skills. Her final project focuses on improving the performance of software defect prediction. Through her research, she aims to contribute to software development by providing a more efficient approach. She can be contacted at email: 2111016120004 @ulm.ac.id.

Rudy Herteno received his bachelor's degree in Computer Science from Lambung Mangkurat University in 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.

Fatma Indriani is a lecturer in the Department of Computer Science at Lambung Mangkurat University, Indonesia. She earned her master's degree in Computer Science from Monash University, Australia, in 2012 and a PhD in Bioinformatics from Kanazawa University, Japan, in 2022. Her research focuses on data preprocessing, natural language processing, bioinformatics, image processing, and machine learning applications in healthcare and biomedical sciences. She has worked extensively on feature engineering techniques for improving predictive models, deep learning for image classification, and text mining for social media and educational data analysis. In addition to her research, she contributes to academic development through teaching and mentoring students in computer science. She can be contacted at email: f.indriani@ulm.ac.id.

Radityo Adi Nugroho received his bachelor's degree in Informatics from the Islamic University of Indonesia in 2004 and a master's degree in Computer Science from Gadjah Mada University in 2007. Currently, he is an assistant professor in the Department of Computer Science at Lambung Mangkurat University. In addition, he served as the Head of the Center for Information and Communication Technology at ULM from 2014 to 2023. He is actively involved in research and development, focusing on software defect prediction and computer vision to enhance system reliability and optimize visual data processing across various applications. He can be contacted at email: radityo.adi@ulm.ac.id.

Muliadi is a lecturer in the Department of Computer Science at Lambung Mangkurat University, where he specializes in Artificial Intelligence, Decision Support Systems, and Data Science. His expertise aims to develop intelligent solutions for decision-making and optimize data analysis for various industrial and academic needs. His academic journey began with a bachelor's degree in Informatics Engineering from STMIK Akakom in 2004 and a master's degree in Computer Science from Gadjah Mada University in 2009. With expertise in Data Science, he also brings valuable skills in Start-up Business Development, Digital Entrepreneurship, and Data Management Staff, supporting technological innovation and efficient data management in both business and academic fields. He can be contacted at email: muliadi@ulm.ac.id.

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Copyright © 2025 by the authors. Published by Jurusan Teknik Elektromedik, Politeknik Kesehatan Kemenkes Surabaya Indonesia. This work is an open-access article and licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).