Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics e-ISSN: 2656-8624
Homepage: https://ijeeemi.org/; Vol. 8, No. 1, pp. 1-16, February 2026

RESEARCH PAPER OPEN ACCES

Comparative Study of Filter, Wrapper, and Hybrid Feature Selection Using Tree-Based Classifiers for Software Defect Prediction

Rahmayanti Rahmayanti, Rudy Herteno, Setyo Wahyu Saputro, Triando Hamonangan Saragih, and Friska Abadi

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

Abstract

Software defect prediction (SDP) is essential for improving software reliability by enabling the early identification of modules that may contain defects before the release stage. SDP commonly exhibits redundant or non-contributory metrics, underscoring the need for feature selection to derive a more informative subset. To address this problem, the present study investigates and compares the effectiveness of three feature-selection strategies: SelectKBest (SKB), Recursive Feature Elimination (RFE), and the hybrid SKB+RFE, in enhancing the performance of tree-based classifiers on the NASA Metrics Data Program (MDP) data collections. The study utilizes three classification algorithms, namely Random Forest (RF), Extra Trees (ET), and Bagging (Decision Tree), with Area Under the Curve (AUC) serving as the primary metric for assessing model performance. Experimental results reveal that the RFE and Extra Trees combination yields the top performance, producing an average AUC of 0.7855. This is subsequently followed by the SKB+RFE+ET configuration, which achieves an AUC of 0.7809, and SKB+ET at 0.7776. These findings demonstrate that iterative wrapper-based approaches such as RFE can identify more relevant and effective feature subsets than filter or hybrid strategies, with the RFE+Extra Trees configuration yielding the strongest overall predictive performance and wrapper-based methods exhibiting higher stability across heterogeneous datasets. Even without hyperparameter tuning and relying solely on class-weighting rather than explicit resampling techniques, the findings offer empirical insight into the isolated influence of feature selection on predictive performance. Overall, the study confirms that RFE combined with Extra Trees offers the strongest predictive performance on NASA MDP data collections and forms a foundation for developing more adaptive and robust models.

Paper History
Received Sept. 12, 2025
Revised Nov. 21, 2025
Accepted Dec. 21, 2025
Published Dec. 24, 2025

Keywords
SelectKBest;
Recursive Feature Elimination;
Tree-Based Classifiers;
Feature Selection;
Software Defect Prediction

Author Email
2211016120010@mhs.ulm.ac.id
rudy.herteno@ulm.ac.id
setyo.saputro@ulm.ac.id
triando.saragih@ulm.ac.id
friska.abadi@ulm.ac.id

I. Introduction

Software plays a fundamental role across modern industrial sectors, including manufacturing, healthcare, education, and transportation. Inadequate software quality may trigger system failures, reduce reliability, and lead to substantial losses. One essential measure to ensure the software's quality involves performing defect prediction early within the software development lifecycle. Software defect prediction (SDP) contributes significantly to improving system reliability and lowering maintenance costs by identifying modules that are likely to be defective prior to release [1]. However, many software metric datasets contain redundant or irrelevant features, which can cause overfitting and degrade the performance of classification models [2]. Consequently, feature selection is a crucial step in identifying the most relevant subset of features to support efficient and accurate predictive modeling [3].

Feature selection methods in SDP are usually divided into filter, wrapper, and hybrid approaches [4]. Filter methods identify candidate features using statistical measures such as correlation, information gain, or mutual information without considering the underlying classifier. Although they scale efficiently, their inability to capture redundancy and complex feature interactions remains a notable limitation. Wrapper methods, in contrast, assess feature subsets by evaluating the classifier's performance, allowing them to account for feature dependencies but increasing computational demands and the likelihood of overfitting. Hybrid techniques combine these paradigms by applying a fast filter step to eliminate irrelevant attributes before refining the selection using a wrapper method, thereby balancing efficiency and predictive performance [5].

Recent advances in feature-selection research have introduced more sophisticated mechanisms. Deep learning-based approaches leverage latent

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

representations to model nonlinear feature interactions [6], while attention-driven architectures assign relevance scores through learned weighting strategies [7]. In parallel, developments in automated machine learning (AutoML) have enabled automated exploration of feature subsets and model configurations using meta-learning and search-based optimization methods [8]. Despite their promise, these techniques typically necessitate considerable computational effort and access to sizeable training datasets, limiting their applicability to classical benchmark datasets such as NASA MDP. For this reason, the present study focuses on lightweight, interpretable, and computationally feasible feature-selection strategies: SelectKBest, RFE, and their hybrid integration, which offer competitive predictive capability while accommodating the practical constraints of real-world SDP settings.

Numerous studies have employed these paradigms to improve SDP performance. Sharma [9] compared eight feature-selection techniques and found that wrapper-based techniques, including the Recursive Feature Elimination (RFE), demonstrated superior stability and accuracy over simple filter-based approaches, achieving an average AUC of 0.72. Similarly, Suntoro [10] reported that ensemble strategies combining AWEIG and AdaCost with Naïve Bayes produced an average AUC of 0.752 on the NASA MDP dataset. Balogun [11] showed that a multifilter-wrapper hybrid method achieved average AUC values between 0.75 and 0.78 on NASA MDP datasets. Other studies highlight the effectiveness of metaheuristic and multi-wrapper approaches: Maulida [12] demonstrated that a Firefly-based feature-selection method increased AUC values up to 0.77 on the NASA MDP datasets, while Aryanti [13] reported that the combination of RFE, Boruta, and Custom Grid Search with a Copeland ranking method achieved an AUC of 0.749, despite the higher computational overhead and increased risk of overfitting.

Despite these developments, several research gaps remain in the existing SDP literature. First, most studies focus on a single category of feature-selection technique and do not conduct systematic comparisons of filter-, wrapper-, and hybrid-based approaches under consistent experimental settings. Second, cross-dataset performance stability remains underexplored, even though the variability in feature dimensionality and class imbalance across NASA MDP datasets poses substantial challenges for generalization. As one of the most widely utilized and diverse benchmark repositories, NASA MDP, with its twelve heterogeneous modules, provides an appropriate foundation for assessing the robustness of feature-selection methods. Notably, Sharma [9] conducted a multi-method comparison; however, the evaluation covered only two NASA MDP datasets, thereby limiting the breadth of the findings. Third, comparative analyses involving tree-based classifiers such as Random Forest, Extra Trees, and Bagging (Decision Tree) remain limited, leaving the interplay between feature-selection strategies and tree-based learning models insufficiently understood. By evaluating three distinct feature-selection paradigms across all twelve NASA MDP datasets and three tree-based classifiers within a unified experimental framework, this study directly addresses these limitations. It provides a broader and more reliable comparative analysis. Considering that NASA MDP datasets exhibit high dimensionality and varying degrees of class imbalance [10], rigorous validation procedures are required. In this study, these challenges are addressed using stratified 10-fold cross-validation to maintain a balanced class distribution within each fold [14].

To close these gaps, this research conducts a comparative evaluation of three feature-selection strategies: SelectKBest (filter-based), Recursive Feature Elimination (wrapper-based), and a combined SelectKBest+RFE hybrid approach, applied to three tree-based classification algorithms: Random Forest, Extra Trees, and Bagging (Decision Tree). The evaluation is performed on twelve NASA MDP datasets using Stratified 10-Fold Cross-Validation, with the Area Under the Curve (AUC) employed as the primary metric, and Average Precision, Recall, Accuracy (ACC), and F1 function as supporting metrics.

This study primarily seeks to assess and compare the efficacy of the three feature-selection approaches in shaping the performance of tree-based software defect prediction models, and to determine which method achieves the best average performance across NASA MDP datasets. By examining multiple datasets with differing structures and class distributions, this investigation provides empirical insights into how different feature-selection strategies influence prediction quality and performance stability in heterogeneous SDP scenarios.

This study makes several significant contributions, which are summarized as follows. First, it provides a comprehensive comparative analysis of filter-based, wrapper-based, and hybrid feature-selection paradigms under a unified experimental framework involving twelve NASA MDP datasets and multiple tree-based classifiers. Second, it systematically evaluates the effectiveness of these feature-selection strategies under class-imbalanced conditions using Stratified 10-Fold Cross-Validation, with AUC as the primary evaluation metric. Third, the experimental results demonstrate that wrapper-based approaches, particularly RFE, exhibit superior capability in identifying relevant features for tree-based classifiers, with the Extra Trees model achieving the highest AUC performance among all evaluated configurations.

II. Materials and Method

Fig 1 presents the overall procedural workflow underlying the proposed study. The research incorporates three feature-selection techniques: SelectKBest (SKB), Recursive Feature Elimination (RFE), and the combined SelectKBest+RFE method. The process begins with collecting data from the NASA Metrics Data Program (MDP), which consists of twelve software modules. Following data acquisition, a preprocessing stage is performed, including converting the target label to binary

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

(0 or 1). The evaluation is then carried out using Stratified 10-Fold Cross-Validation to preserve the proportional allocation of defect and non-defect class groups within every fold. SelectKBest utilizes the mutual information scoring function to assess the relevance of each feature to the defect label. In contrast, RFE iteratively eliminates features with minimal contribution until an optimal subset is obtained. The SelectKBest+RFE hybrid approach combines the advantages of both techniques by applying mutual information-based filtering prior to RFE's iterative elimination process.

Building on this workflow, each pairing of feature-selection method and classification model is assessed with the Area Under the Curve (AUC) metric applied to evaluate software defect prediction effectiveness. The study systematically examines and contrasts the performance of three feature-selection approaches on tree-based software defect prediction models, aiming to determine which method provides the highest average performance across the NASA MDP datasets. Through this comparative investigation, the study highlights the influence of different feature-selection strategies on prediction quality across datasets with diverse characteristics.

In this study, the comparative assessment of feature-selection techniques is purposefully confined to tree-based classifiers, notably Random Forest (RF), Extra Trees (ET), and Bagging with Decision Tree-based learners. These ensemble models are commonly utilized in SDP because they perform reliably on high-dimensional code metrics, effectively capture nonlinear relationships, and naturally accommodate heterogeneous feature scales without requiring explicit normalization. They also generate feature-importance measures that align with the impurity-based criteria used in feature selection, thereby strengthening the interpretability of the selected metrics. Relative to kernel-based approaches such as Support Vector Machines or neural network models, the selected classifiers provide a practical balance between predictive accuracy, computational efficiency, and model transparency, an essential consideration when experiments must be consistently executed across multiple NASA MDP datasets. As a result, non-tree-based models are excluded from the present study and are identified as a direction for future research.

This study also highlights the importance of maintaining performance stability across the NASA MDP datasets, which differ considerably in class imbalance levels and feature characteristics. To reinforce the model's generalization capability, all experiments were executed uniformly across all twelve datasets. Through this approach, the analysis becomes more comprehensive in determining whether a given feature-selection technique can sustain stable predictive performance across datasets that vary in dimensionality, distribution patterns, and noise levels. Accordingly, the study evaluates both the contributions of the selected features and the robustness of each method when confronted with the heterogeneous characteristics of the NASA MDP datasets.

A. Data Collection

The present research makes use of twelve datasets provided by the NASA MDP D", all of which are publicly available at: https://github.com/klainfo/NASADefectDataset/tree/master. The NASA MDP collection (CM1-PC5) is widely regarded as a benchmark source in SDP research and is commonly utilized to construct and evaluate SDP models

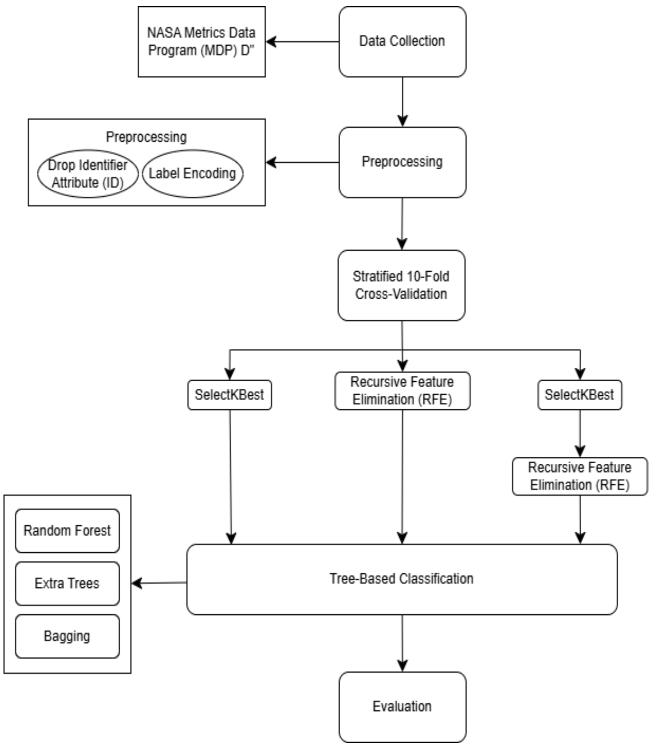
NASA Metrics Data Program (MDP) D"
Data Collection
Preprocessing
Drop Identifier Attribute (ID)
Label Encoding
Preprocessing
Stratified 10-Fold Cross-Validation
SelectKBest
Recursive Feature Elimination (RFE)
SelectKBest
Recursive Feature Elimination (RFE)
Random Forest
Extra Trees
Bagging
Tree-Based Classification
Evaluation

Fig 1. **Study Workflow Diagram**

[15]. Each dataset provides static code metrics at the module level, where a module may correspond to a method, function, or internal routine in the software system [16]. The selection of these twelve datasets is intended to capture a broad spectrum of scenarios and levels of complexity associated with software defect prediction.

Despite their extensive use, the NASA MDP datasets exhibit several well-known challenges, particularly severe class imbalance [17] and the presence of noisy or irrelevant features [18]. Class imbalance is particularly severe in several datasets, such as MC1 (1.8% defective modules) and PC2 (2.2% defective modules), where defective modules are extremely scarce. In contrast, datasets such as MC2 (35.5%) and PC5 (27%) exhibit more proportionate class distributions. Such disparities have notable implications for machine-learning behavior, influencing model sensitivity to minority classes, the stability of feature-selection results, and the dependability of performance metrics. Consequently, a clear understanding of each dataset's imbalance profile is vital to ensure equitable and reliable model evaluation across all experimental settings. A comprehensive summary of the dataset characteristics used in the analysis is presented in Table 1.

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

Table 1. NASA-MDP Dataset Details

Datasets	Attributes	Instances	Non-Defects	Defects	Defective (%)
CM1	38	327	285	42	12.8
JM1	22	7720	6108	1612	20.9
KC1	22	1162	868	294	25.3
KC3	40	194	158	36	18.6
MC1	39	1952	1916	36	1.8
MC2	40	124	80	44	35.5
MW1	38	250	225	25	10
PC1	38	679	624	55	8.1
PC2	37	722	706	16	2.2
PC3	38	1053	923	130	12.3
PC4	38	1270	1094	176	13.9
PC5	39	1694	1236	458	27

Although several NASA MDP datasets contain pronounced class imbalance, this study deliberately excludes any resampling-based balancing methods. The choice is made to ensure that the influence of the feature-selection techniques may be objectively evaluated without confounding effects arising from synthetic data generation or modified class distributions. Instead, model-level class weighting is applied where supported, and Stratified K-Fold Cross-Validation is implemented to maintain the original class proportions across folds. In this design, the comparative analysis remains focused on the inherent behavior and effectiveness of the feature-selection strategies under naturally imbalanced conditions.

B. Preprocessing

The data preprocessing stage was carried out to ensure that the dataset met the required standards of quality, consistency, and cleanliness prior to model training. This stage is essential for preparing raw data in order that it can be handled efficiently by ML algorithms [19]. In this study, several preprocessing steps were implemented. The process began by removing non-predictive attributes, such as the ID column, which serves solely as a unique identifier and provides no information relevant to defect labels [20].

Table 2. Before Preprocessing

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

Subsequently, label encoding was used to transform the categorical target variable into a numeric form, enabling the model to interpret it appropriately [21]. Specifically, across all datasets, the label 'Y' indicating a defective module was transformed into '1', whereas the label 'N' indicating a non-defective module was transformed into '0'. Examples of this label transformation are presented in Table 2 and Table 3.

Table 3. Following Preprocessing

LOCK_BLANK	BRANCH_OUT	…	LOC TOTAL	Defective
2	3	…	9	0
3	3	…	13	0
38	35	…	109	0
1	7	…	41	1
…	…	…	…	…
3	3	…	12	0
6	9	…	32	0
1	3	…	10	0

C. Stratified 10-Fold Cross-Validation

After completing the preprocessing stage to ensure data cleanliness and quality, the study proceeded to the data partitioning phase. At this stage, the Stratified 10-Fold Cross-Validation method was applied, splitting the dataset across ten folds while maintaining consistent proportions between defect and non-defect categories within every fold [22]. This method was chosen to address the challenges posed by imbalanced class distributions, where a purely random split may produce folds containing very few or no samples from the minority class, potentially introducing bias into the model performance assessment [23]. As illustrated in Fig 2 [24], the stratified approach ensures that every fold proportionally reflects the overall data distribution. In each iteration, nine folds are used for training, with the remaining fold serving as the evaluation set; this cycle continues until each fold serves as the test set [25].

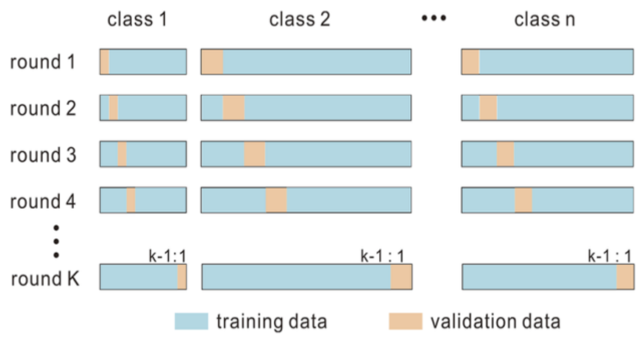

Fig 2. Stratified K-Fold CV Scheme [24]

D. SelectKBest

This present study adopts the SelectKBest (SKB) technique, a filter-based feature-selection approach that identifies the *k* highest-scoring features by evaluating their quantitative relevance to the target variable. The approach is selected for its computational efficiency and straightforward implementation [26]. In this work, Mutual Information (MI) serves as the scoring metric. Formally, the MI between a feature X and the target class Y is expressed as Eq. (1) [27].

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) \quad (1)$$

which quantifies the extent to which uncertainty about *Y* diminishes when *X* is observed. A higher MI value indicates a stronger dependency between the feature and the target, including non-linear relationships that cannot be captured by correlation-based metrics [28]. Using this formulation, SKB ranks all available features and selects the top *k* features as defined by the *K_FINAL* parameter. The SKB algorithm is summarized in the pseudocode shown in Table 4.

Table 4. Algorithm 1

SelectKBest (Mutual Information)

```
1.  BEGIN
2.  Dataset = load_csv(FOLDER_PATH)
3.  (X, y) = split_features_and_target(Dataset)
4.  X = impute_median(X)
5.  skb = SelectKBest(score_func=mutual_information,
    k=K_FINAL)
6.  X_skb = skb.fit_transform(X, y)
7.  SelectedFeatures = skb.get_support_indices()
8.  OUTPUT SelectedFeatures, X_skb
9.  END
```

E. Recursive Feature Elimination

In the wrapper-based feature-selection stage, this study uses Recursive Feature Elimination (RFE), a wrapper method that evaluates model performance as it progressively removes the least informative features. During each iteration, RFE orders all predictors according to model-specific importance measures such as impurity-based feature importance in tree-based classifiers and discards those contributing minimally to predictive accuracy. By relying on the model's internal evaluation rather than independent statistical measures, this iterative procedure ensures that feature reduction is closely aligned with the classifier's learning behavior. RFE is selected because it accommodates interactions among features and adapts the selection process to the characteristics of the underlying model [9] [29]. Through this mechanism, the approach allows the production of a highly representative and relevant feature subset relative to filter-based methods. Given these advantages, RFE offers a balanced trade-off between complexity and performance, thereby serving as an effective feature selection technique within the context of software defect prediction [9]. Table 5 presents the pseudocode for the RFE algorithm.

Table 5. Algorithm 2

Recursive Feature Elimination

```
1.  BEGIN
2.  Dataset = load_csv(FOLDER_PATH)
3.  (X, y) = split_features_and_target(Dataset)
4.  X = impute_median(X)
5.  base_model = RandomForest(n_estimator=200,
    class_weight=balanced, random_state=42)
6.  rfe = RFE(estimator=base_model,
    n_selected_features=K_FINAL, step_size=0.1)
7.  X_rfe = rfe.fit_transform(X, y)
8.  SelectedFeatures = rfe.get_support_indices()
9.  OUTPUT SelectedFeatuers, X_rfe
10. END
```

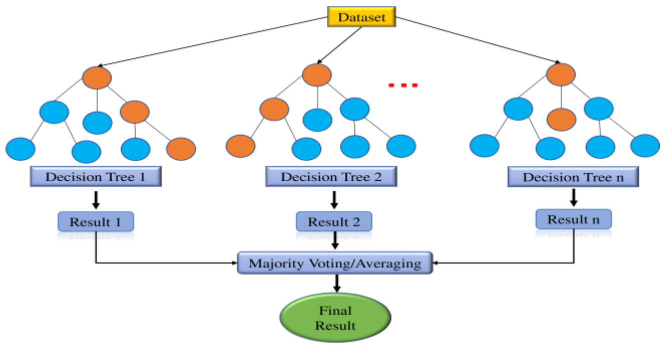

Fig 3. **Random Forest Model [30]**

F. Hybrid Feature Selection Strategy (SKB + RFE)

This study employs a hybrid feature-selection workflow that integrates SelectKBest (SKB) with Recursive Feature Elimination (RFE) to combine the strengths of filter-based and wrapper-based techniques. The process begins with SKB, which uses mutual-information-based univariate scoring to select the top 25 features (K_FIRST) in accordance with the study's fixed experimental setup. This initial step reduces dimensionality by discarding features that exhibit low independent discriminative capability. The resulting 25 features are then passed to the RFE module, which evaluates inter-feature relationships through an iterative, model-driven elimination process. During each iteration, the tree-based classifier generates importance rankings for all remaining predictors, and RFE removes the predictors with the lowest contributions to the model. This procedure continues until the feature subset is refined to 15 features (K_FINAL). Through sequential filtering and wrapper-based refinement, the hybrid method produces a compact, classifier-aligned feature subset. The hybrid algorithm is summarized in the pseudocode shown in Table 6.

Table 6. Algorithm 3

Hybrid (SelectKBest+RFE)

```
1.  BEGIN
2.  Dataset = load_csv(FOLDER_PATH)
3.  (X, y) = split_features_and_target(Dataset)
4.  X = impute_median(X)
5.  skb = SelectKBest(score_func=mutual_information,
    k=K_FIRST)
6.  X_skb = skb.fit_transform(X, y)
7.  Selected_SKB = skb.get_support_indices()
8.  base_model = RandomForest(n_estimator=200,
    class_weight=balanced, random_state=42)
9.  rfe = RFE(estimator=base_model,
    n_selected_features=K_FINAL, step_size=0.1)
10. X_hybrid = rfe.fit_transform(X_skb, y)
11. Selected_RFE_local = rfe.get_support_indices()
12. Selected_Hybrid = map_indices(Selected_SKB,
    Selected_RFE_local)
13. OUTPUT Selected_Hybrid, X_hybrid
14. END
```

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

G. Random Forest

The Random Forest (RF) model represents an ensemble-based algorithm that constructs numerous decision trees, each one trained independently with bootstrap sampling and random feature selection to promote model diversity. The overall prediction is produced using a majority-vote mechanism that aggregates the outputs across the ensemble trees, which enhances its overall generalization capacity and reduces the likelihood of overfitting [22]. RF is also well regarded for its robustness in handling high-dimensional data, missing values, and outliers, while maintaining the ability to capture nonlinear feature interactions [31]. Within each tree, an impurity measure is used to select the optimal feature split, with the Gini Index being a standard metric [32]. The Gini impurity for a dataset *D* is expressed as Eq. (2) [22].

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{2}$$

where p_i represents the proportion among the samples belonging to the *i*-th class, and *m* denotes the number of classes. When a node K is divided into two subgroups D_1 and D_2, containing T_1 and T_2, samples respectively, the total impurity at that node is computed as Eq. (3) [22].

$$TotGini(K) = \frac{T_1}{T} Gini(D_1) + \frac{T_2}{T} Gini(D_2) \tag{3}$$

with T indicating the overall count of samples at the node K. A smaller impurity value reflects a more effective split, thereby supporting improved classification performance in the Random Forest model.

In this study, RF is instantiated via the scikit-learn library with n_estimators = 200, class_weight = “balanced”, and random_state = 42, while all other structural hyperparameters, such as max_depth or min_samples_split, remain at their default values. This setup provides a robust yet sufficiently general baseline without relying on extensive hyperparameter optimization, thereby allowing the investigation to focus on the comparative performance of the feature-selection techniques. The identical RF configuration is also adopted as the base estimator within the RFE procedure to maintain methodological consistency between the wrapper-based selection process and the final classification model. The architecture of the Random Forest model employed in this study is shown in Fig 3 [30].

H. Extra Trees

The Extra Trees (ET) model, originally proposed by Pierre Geurts et al. (2006) [33], represents an advancement of the traditional Decision Tree model. Unlike Random Forest, which depends on bootstrap sampling, Extra Trees generates an ensemble of unpruned decision trees by leveraging the entire training dataset and introducing greater randomness in both feature-selection and the determination of split points at each node. This heightened randomness promotes the formation of more heterogeneous trees and reduces correlations across the ensemble, which in turn boosts the model’s generalization performance and improves computational efficiency. The algorithm is also known for its resilience to overfitting and outliers, as well as its capability to process high-dimensional datasets that exhibit nonlinear interactions among features. For these reasons, Extra Trees is employed in this study for its stability, rapid training, and its balanced trade-off between accuracy and efficiency [34]. To determine each split, the algorithm samples a random threshold t_j for feature j from a uniform distribution, as formally defined in Eq. (4) [33].

$$t_j \sim Uniform(a_j, b_j) \tag{4}$$

Each threshold produces a pair of candidate splits, which are assessed based on the reduction in node impurity, as formally defined in Eq. (5) [33].

$$\Delta I(s) = I(D) - (\frac{T_1}{T} I(D_1) + \frac{T_2}{T} I(D_2)) \tag{5}$$

Where I denotes the impurity measure, and T_1, T_2, and T represent the number of instances in the resulting subsets.

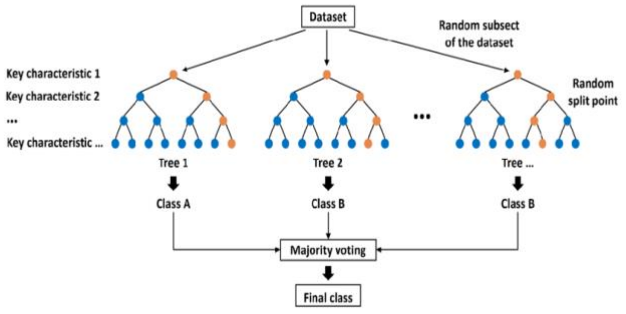

Fig 4. **Extra Trees Model [35]**

In this study, ET is configured using n_estimators = 300, random_state = 42, and n_jobs = -1 to leverage parallel processing, while all remaining hyperparameters retain their scikit-learn default values. Class weighting is applied when available to help counteract the substantial label imbalance present in the NASA MDP datasets. As with the Random Forest configuration, the model is not subjected to extensive hyperparameter tuning. Instead, a robust and stable baseline setup is used to ensure that any observed performance differences arise primarily from the feature-selection techniques rather than from variations driven by aggressive hyperparameter optimization. The architecture of the Extra Trees model employed in this study is shown in Fig 4 [35].

I. Bagging (Decision Tree)

The Bagging (Bootstrap Aggregating) algorithm, which utilizes a Decision Tree as its base estimator, represents a type of ensemble-based method designed to enhance classification accuracy and model stability by effectively lowering variance and minimizing overfitting. Introduced by Breiman (1996) [36], the method generates multiple training subsets using bootstrap sampling, random sampling with replacement, and trains an independent Decision Tree on each subset. Each model contributes an individual prediction, and the overall output is then produced through a majority-vote strategy for classification or by mean aggregation for regression. Through this aggregation mechanism, Bagging consistently delivers more robust and reliable performance than a single standalone model [37] [38].

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

In this study, Bagging is implemented using scikit-learn's BaggingClassifier, employing a Decision Tree-based learner configured with class_weight = "balanced" and random_state = 42. The ensemble comprises 200 estimators, while all other parameters, such as max_depth or min_samples_split, are kept at their default settings. This configuration is selected to enhance variance reduction and mitigate the effects of class imbalance, while deliberately avoiding extensive hyperparameter tuning that might interfere with the comparative evaluation of the filter-based, wrapper-based, and hybrid feature-selection approaches. The workflow of the Bagging method implemented in this study

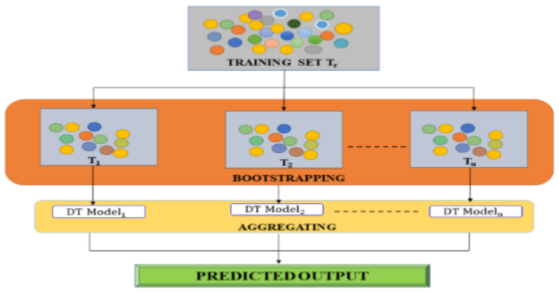

Fig 5. **Flowchart of the bagging method [39]**

is shown in Fig 5 [39].

J. Evaluation

Performance evaluation in this study adopts Stratified 10-Fold Cross-Validation with the Area Under the Curve (AUC) as the primary metric. A stratified 10-fold CV is selected because it offers a well-established trade-off between bias and variance in estimating generalization performance while maintaining a reasonable computational cost across the twelve heterogeneous NASA MDP datasets. Compared with smaller fold settings such as 3-fold or 5-fold, the 10-fold configuration provides more stable performance estimates without excessively increasing run time. Stratification further ensures that each fold approximately preserves the original ratio between defective and non-defective modules, which is crucial under class-imbalanced conditions frequently observed in SDP datasets.

AUC is used as the main evaluation criterion; it is insensitive to class proportions and captures the model's ability to rank defective modules higher than non-defective ones across all possible decision thresholds [40]. This threshold-independent property makes AUC more informative than overall accuracy when the positive (defect) class is in the minority. AUC is computed by measuring the area beneath the Receiver Operating Characteristic (ROC) curve, which depicts the balance between the True Positive Rate (TPR) and the False Positive Rate (FPR). The resulting metric ranges from 0 to 1, with higher values indicating stronger discriminative capability, whereas lower values indicate limited predictive performance [41]. The TPR and FPR metrics follow standard definitions commonly adopted in machine learning research, as expressed in Eq. (6) and Eq. (7) [42].

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

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

Mathematically, AUC is expressed as the integral of the ROC curve, as expressed in Eq. (8) [42].

$$AUC = \int_0^1 TPR(FPR)d(FPR) \tag{8}$$

This formulation provides an overall indicator of how well the model can consistently distinguish between the two classes. The evaluation excludes classifiers trained on the full feature set, as the study focuses on comparing filter-based, wrapper-based, and hybrid feature-selection strategies rather than benchmarking untuned classifier performance. Including a no-feature-selection baseline would introduce confounding factors unrelated to dimensionality reduction and reduce methodological coherence. To complement predictive performance, computational cost is also measured by cross-validation execution time and peak memory usage, providing practical insight into the trade-offs associated with the higher complexity of wrapper and hybrid approaches. For each configuration and dataset, Stratified 10-Fold CV produces the mean of AUC, along with Average Accuracy (ACC), Precision, Recall, and F1 as secondary metrics. Per-dataset scores are averaged to obtain single summary values, and AUC results are further aggregated across the twelve datasets to enable cross-method comparison. The algorithm for the final model evaluation is summarized in the pseudocode presented in Table 7.

III. Results

This study is structured into three primary stages designed to systematically examine how different feature-selection strategies affect the performance exhibited by software-defect prediction models. The first stage employs the filter-based SelectKBest (SKB) method, which identifies the most relevant features using mutual information scores relative to the target label. The second stage applies the wrapper-based Recursive Feature Elimination (RFE) approach, where features with the lowest contribution to model performance are iteratively removed until an optimal subset is obtained. The final stage adopts a hybrid strategy that integrates SKB and RFE, beginning with statistical filtering through SKB, followed by RFE-based refinement to account for inter-feature dependencies within the classification model.

Table 7. **Algorithm 4**

Model Evaluation

```
1. BEGIN
2. Classifiers = {RandomForest, ExtraTrees,
   Bagging(DT)}
3. For each CSV in FOLDER_PATH:
     3.1 Dataset = load_csv(CSV)
     3.2 (X, y) = split_features_and_target(Dataset)
     3.3 X = impute_median(X)
```

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

```
        3.4 cv = StratifiedKFold(n_splits = 10)
        3.5 For each clf in Classifiers:
            a)  Pipe_SKB = [Imputer → SelectKBest(MI,
                k=K_FINAL) → clf]
            b)  Pipe_RFE = [Imputer → RFE(base RF,
                n=K_FINAL, step=0.1) → clf]
            c)  Pipe_Combo = [Imputer →
                SelectKBest(MI,
                k=max(K_FIRST,K_FINAL)) → RFE(base
                RF, n=K_FINAL) → clf]
        3.7 For each Pipeline in {SKB, RFE, Combo}:
            a)  StartTimer() and StartMemoryMonitor()
            b)  Scores = cross_validate(Pipeline, X, y, cv,
                metrics={AUC, ACC, PREC, REC, F1})
            c)  Time_sec = StopTimer()
            d)  PeakMem_MB = StopMemoryMonitor()
            e)  Append {Dataset, clf, method, mean
                (AUC, ACC PREC, REC, F1), Time_sec,
                PeakMem_MB, K_FINAL, K_FIRST} to
                SummaryTable
    4.  Save SummaryTable =
        "SDP_results_summary.csv"
    5.  END
```

This workflow offers a structured foundation for comparing the strengths of each method. Model evaluation is performed through Stratified 10-Fold Cross-Validation to maintain balanced distributions between defective and non-defective instances across folds. The outcomes of various combinations involving feature-selection techniques and tree-based classification algorithms are summarized in Table 8, which reports Area Under the Curve (AUC) metric values for the NASA MDP data collections (CM1-PC5). The first column lists the method model combinations, while the subsequent columns present their corresponding AUC scores. These results serve as the basis for assessing the influence of each feature-selection approach on predictive performance across datasets with varying characteristics.

Across all evaluated configurations, the wrapper-based and hybrid feature-selection methods tend to produce more consistent performance trends on the NASA MDP datasets, while the filter-only SKB method exhibits higher sensitivity to variations among datasets. Although each technique achieves competitive results with the tree-based classifiers, RFE consistently demonstrates marginally superior predictive performance, especially when used in conjunction with the Extra Trees (ET) model. These findings suggest that RFE is particularly adept at identifying feature subsets that complement the inductive behavior of tree-based learning algorithms.

Across several datasets, the three methods maintain comparable predictive performance, as reflected in AUC values ranging from 0.60 to 0.93. Datasets such as PC1 and PC4 exhibit high AUC scores (≥ 0.90), indicating that the models effectively capture feature patterns strongly associated with software defects. Conversely, datasets such as MC2 exhibit noticeably lower AUC values (≤ 0.70), which may stem from class imbalance or weak correlations between the available features and the target variable. These disparities highlight how dataset-specific properties influence the sensitivity of feature-selection methods. Viewed collectively, the experimental outcomes reported in Table 8, the distributional behaviors depicted in Fig. 6, and the average AUC values presented in Fig. 7 clearly indicate that feature-selection exerts a significant

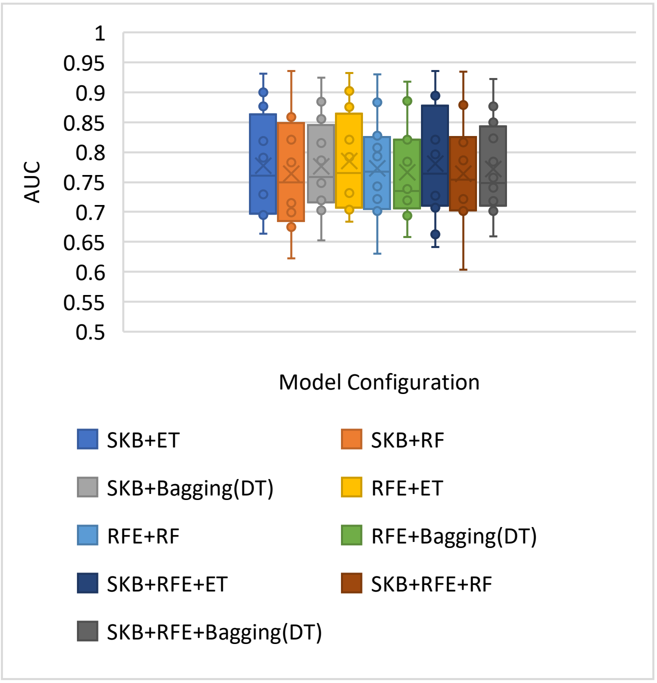

Fig 6. **Boxplot of AUC Scores**

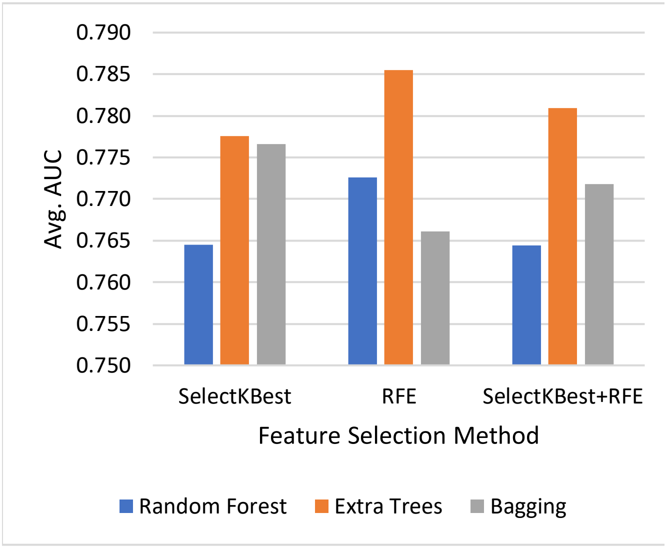

Fig 7. **Average AUC Values**

influence on classification performance in SDP. These findings highlight the critical role of identifying suitable feature subsets to ensure stable and dependable predictive accuracy across datasets that differ in both structural composition and statistical characteristics.

Fig 6 presents the AUC distribution across the 12 NASA MDP datasets. The boxplot highlights clear distinctions in consistency among the nine evaluated configurations, with RFE+Bagging(DT) producing the

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

Table 8. AUC for Tree-Based Classifying Models using Different Feature Selection Methods

Classifier	Dataset											
	CM1	JM1	KC1	KC3	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SKB+ET	0.6944	0.7050	0.7299	0.7050	0.8769	0.6638	0.6950	0.8997	0.8183	0.8208	0.9313	0.7907
SKB+RF	0.6753	0.6996	0.7159	0.7152	0.8585	0.6228	0.6799	0.8709	0.7955	0.8210	**0.9359**	0.7834
SKB+Bagging(DT)	0.7197	0.7023	0.7145	**0.7610**	0.8555	0.6522	0.7483	0.8846	0.7550	0.8151	0.9241	0.7866
RFE+ET	0.6833	**0.7070**	**0.7321**	0.7390	0.8758	**0.7088**	0.7034	0.9019	**0.8291**	0.8209	0.9327	0.7921
RFE+RF	0.7143	0.7022	0.7216	0.7015	0.8279	0.6303	0.7428	0.8833	0.8075	0.8168	0.9298	0.7926
RFE+Bagging(DT)	**0.7314**	0.7021	0.7193	0.7199	0.8215	0.6584	0.7380	0.8853	0.6942	0.8206	0.9177	0.7842
SKB+RFE+ET	0.6630	**0.7070**	**0.7321**	0.7189	**0.8948**	0.6412	0.7270	**0.9070**	0.8210	**0.8269**	0.9354	**0.7962**
SKB+RFE+RF	0.7011	0.7022	0.7216	0.7091	0.8287	0.6034	0.7029	0.8784	0.7878	0.8164	0.9351	0.7859
SKB+RFE+Bagging(DT)	0.7081	0.7021	0.7193	0.7185	0.8501	0.6591	**0.7568**	0.8771	0.7410	0.8232	0.9221	0.7843

most compact distribution, indicating superior stability. The other wrapper-based and hybrid approaches also maintain relatively consistent patterns, whereas the filter-based (SKB) models exhibit broader variation and heightened sensitivity to dataset-specific characteristics. Overall, the results confirm that wrapper and hybrid feature-selection strategies deliver more robust and stable performance compared to the filter-only approach. Fig 7 offers a comparative overview of the average AUC values across the evaluated feature selection strategies. Among all combinations being assessed, RFE paired with Extra Trees yields the greatest mean AUC with a score of 0.7855, closely trailed by SelectKBest+RFE+Extra Trees (0.7809) and SelectKBest+Extra Trees (0.7776). Despite the narrow gap between these results, all methods demonstrate consistently competitive performance across the NASA MDP datasets. Nonetheless, the leading performance of the RFE+Extra Trees configuration reinforces the observation that wrapper-based iterative elimination is particularly effective in identifying salient feature subsets, thereby improving the model's

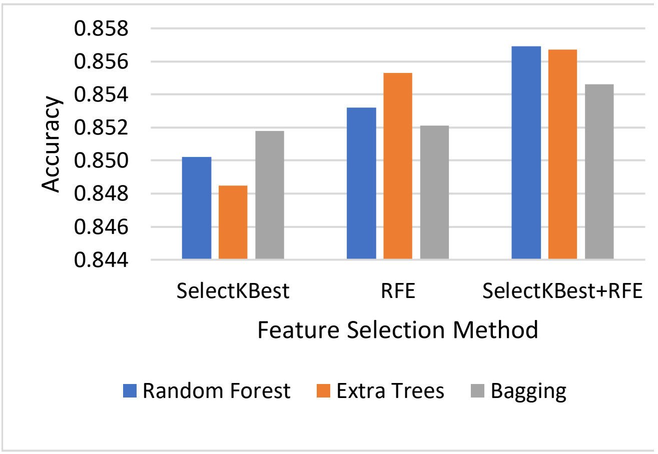

Fig 8. Average Accuracy Values

discriminative capability in distinguishing defective from non-defective software modules. Fig 8 reports the average accuracy achieved by the evaluated feature-selection techniques. Among the tested configurations,

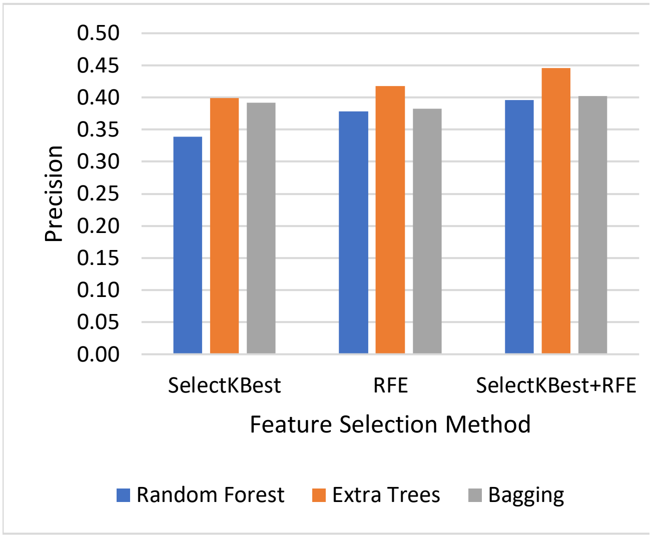

Fig 9. Average Precision Values

SelectKBest+RFE combined with Random Forest attains the highest accuracy (0.8569), followed closely by SelectKBest+RFE with Extra Trees (0.8567) and RFE with Extra Trees (0.8553). Although accuracy is not emphasized as the primary metric due to the inherent class imbalance present in the datasets, the narrow margin between results indicates that the methods exhibit comparable capability in modeling overall predictive patterns. The slight performance edge of the hybrid approach reinforces the notion that integrating filter-based preselection with wrapper-based iterative refinement can yield a more effective and computationally efficient feature-selection process. Fig 9 compares the average precision achieved by the evaluated feature-selection strategies. The hybrid SelectKBest+RFE approach attains the highest precision across all classifiers, providing noticeable improvements relative to both the individual filter-based and wrapper-based methods. Among the classifiers, Extra Trees consistently records the strongest precision performance, reaching 0.4458 when paired with the hybrid strategy. These outcomes demonstrate that integrating filter and wrapper relevance assessments is particularly effective in minimizing false

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

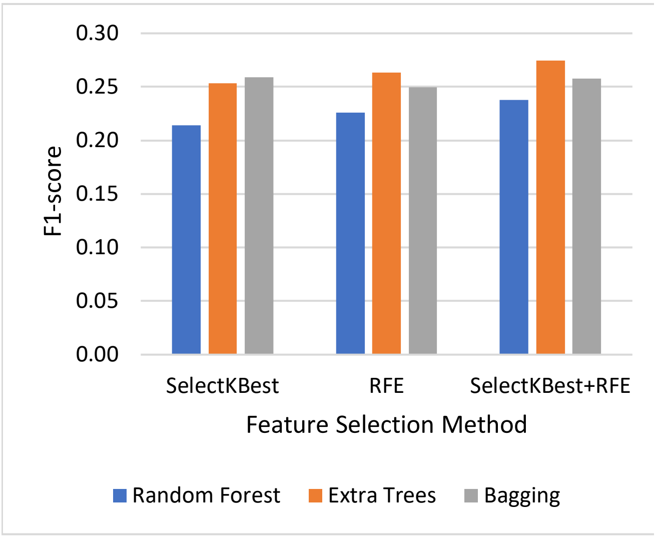

Fig 11. Average F1-score

positives, thereby enhancing the model's confidence in identifying defective modules. The results further show that while RFE offers an improvement over SelectKBest,

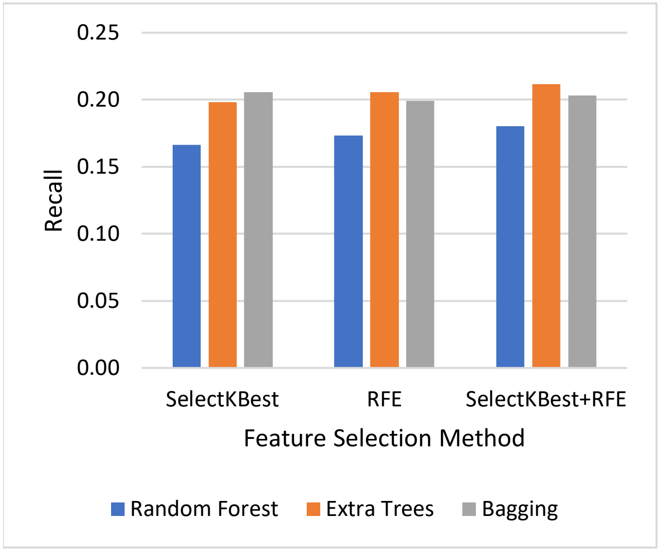

Fig 10. Average Recall Values

the hybrid method yields the most consistent and superior precision overall. Fig. 10 presents the average recall results for all classifier and feature-selection combinations. Although recall values remain modest due to the pronounced class imbalance in the NASA MDP datasets, the hybrid SelectKBest+RFE method consistently yields the strongest performance. Extra Trees achieves the highest recall of 0.2119 under this configuration, with Random Forest and Bagging performing similarly. The performance gains observed from SelectKBest to RFE and then to the hybrid approach indicate that iterative feature elimination helps retain features essential for detecting the minority (defect) class, thereby reducing false negatives. Despite the relatively small differences among methods, the hybrid strategy provides the most balanced capability for recovering defective modules across the datasets.

Table 9. Wilcoxon Signed-Rank Test Comparing RFE+ET Against Other Configurations

Method Comparison	p-Value	Significant ($p < 0.05$)
RFE+ET vs SKB+ET	0.0327	Yes
RFE+ET vs SKB+RFE+ET	0.8457	No
RFE+ET vs RFE+RF	0.1294	No
RFE+ET vs SKB+RF	0.0024	Yes
RFE+ET vs SKB+RFE+RF	0.0161	Yes
RFE+ET vs RFE+Bagging	0.0923	No
RFE+ET vs SKB+Bagging	0.3804	No
RFE+ET vs SKB+RFE+Bagging	0.1294	No

Fig 11 presents the average F1-score for all feature-selection strategies and classifiers. The hybrid SelectKbest+RFE approach achieves the highest performance, with Extra Trees reaching the highest score of 0.2747. These results show that the hybrid method reduces both false positives and false negatives more effectively than the standalone techniques. Although classifier differences are modest, Extra Trees provides the most consistent gains, reinforcing its suitability for feature-selection-based SDP. Overall, the F1 results confirm the benefit of integrating filter and wrapper mechanisms to produce more reliable and discriminative feature subsets.

The results presented in Table 9 show that RFE+ET delivers statistically significant gains over SKB+ET, SKB+RF, and the hybrid SKB+RFE+RF ($p < 0.05$), indicating that its performance advantage is unlikely to be due to random variation. In contrast, its comparisons with SKB+RFE+ET, RFE+RF, and their Bagging-based configurations yield non-significant differences ($p > 0.05$), reflecting largely comparable performance among these methods. Collectively, these findings suggest that the strength of RFE+ET is most evident when evaluated against filter-based techniques and hybrid approaches that employ Random Forest.

Table 10. Average Computational Cost

Method	Runtime (sec)	PeakMem (MB)
SKB	18.01	0.993
RFE	76.42	0.997
SKB+RFE	63.48	1.003

Table 10 summarizes the average computational cost of the evaluated feature-selection methods. SKB remains the most efficient option, exhibiting the lowest runtime and memory usage across all datasets and classifiers. Conversely, RFE and the hybrid SKB+RFE configuration impose considerably higher computational demands due to their iterative elimination processes, with RFE showing the greatest runtime overhead. Memory usage is relatively uniform across methods, indicating that the dominant source of cost differences lies in time complexity rather than memory consumption. Overall, the results highlight an important practical trade-off, although RFE-

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

based strategies can deliver improved predictive performance, they do so at the expense of substantially greater computational cost when compared to the more resource-efficient SKB.

IV. Discussion

The AUC results in Table 8 and Fig 7 show that the RFE+Extra Trees configuration delivers the highest average AUC of 0.7855, followed by SKB+RFE+ET (0.7809) and SKB+ET (0.7776). Dataset-level findings further support this pattern: the SKB+RFE+ET method achieves an AUC of 0.8948 on MC1, with similarly strong performance on PC1 and PC4. In contrast, the smaller performance gaps observed in KC3 and MC2 suggest that the influence of feature-selection strategies varies across datasets. Moreover, the variability analyses in Fig. 6 demonstrate that wrapper and hybrid approaches tend to produce more stable AUC distributions compared to filter-only techniques, with RFE+Bagging(DT) showing the tightest interquartile range.

The effectiveness of RFE+Extra Trees can be explained by the complementary strengths of the two components. Extra Trees employs a high degree of randomization during node splitting, enabling broader exploration of the feature space and reducing the risk of overfitting. RFE subsequently removes low-importance features based on model-derived importance scores, gradually refining the feature subset toward the most discriminative attributes. This synergy enhances noise tolerance and reduces redundancy characteristics, particularly advantageous for the high-dimensional NASA MDP datasets. Although the hybrid SKB+RFE benefits from an initial filter stage before wrapper refinement, its discriminative ability remains slightly below that of pure RFE.

Class imbalance also plays a pivotal role in shaping model performance, especially in datasets like MC1 and PC2. Filter-based methods may inadvertently emphasize majority-class patterns, while RFE can yield unstable importance estimates when defective samples are limited. Although stratified cross-validation and class weighting help alleviate these issues, recall scores remain low, which is expected under severe imbalance. Accuracy results in Fig 8 indicate that the classifiers still capture broad predictive patterns effectively; however, AUC remains the more reliable metric for imbalanced data. The precision scores in Fig. 9 show that the hybrid SKB+RFE configuration achieves the highest average precision, with Extra Trees achieving 0.4458. Consistent recall and F1 patterns in Fig 10 and Fig 11 reinforce the conclusion that the hybrid method enhances minority-class detection, with Extra Trees again demonstrating the most stable performance.

Significance test in Table 9 further validates findings. RFE+Extra Trees achieves statistically significant improvements over SKB+ET, SKB+RF, and SKB+RFE+RF ($p < 0.05$), while performing comparably to SKB+RFE+ET and RFE+RF. Table 10 introduces an additional practical consideration: SKB is the most computationally efficient technique, whereas RFE and SKB+RFE require substantially higher runtimes due to iterative elimination. These results underscore the importance of balancing predictive benefits with computational cost, particularly in environments requiring rapid model updates.

Table 11. Comparison of AUC Findings with Previous Studies

Study	Method	Avg. AUC
Suntoro et al. [10]	AWEIG + AdaCost + NB	0.752
Aryanti et al. [13]	RF + RFE, Boruta, Grid Search + Copeland	0.749
Herteno et al. [43]	RF + Correlation	0.5389
Our	**RFE + Extra Trees**	**0.7855**

Table 11 places the present findings within the context of previous studies on software defect prediction. Suntoro et al. employed an AWEIG combined with AdaCost and Naïve Bayes to address class imbalance, achieving competitive performance under imbalanced conditions. Aryanti et al. proposed a more complex pipeline combining Random Forest with RFE, Boruta, Grid Search, and the Copeland ranking strategy, highlighting the benefits of ensemble feature-selection and hyperparameter optimization. Herteno et al. investigated correlation-based feature selection integrated with Random Forest classifiers, demonstrating the effectiveness of relevance-based filtering in reducing redundant features. Compared to these approaches, the proposed RFE+Extra Trees configuration achieves a higher AUC value (0.7855), indicating improved discriminative capability. This performance advantage stems from Extra Trees' high-randomization splitting strategy coupled with RFE's iterative refinement, which together yield more stable and discriminative feature subsets across a wide range of datasets.

Despite these encouraging results, several limitations must be acknowledged. The models were trained with fixed hyperparameters, and no systematic tuning was performed, which may affect model stability. The evaluation was limited to NASA MDP datasets, and advanced balancing strategies beyond class weighting were not employed, thereby increasing the risk of overfitting on smaller or more imbalanced datasets. Future studies should incorporate systematic hyperparameter optimization, experiments on larger and more diverse datasets, and advanced oversampling methods such as SMOTE or ADASYN. Additionally, exploring evolutionary or deep learning-based feature-selection approaches and including broader model families beyond tree-based classifiers may provide deeper insights into feature importance, scalability, and generalizability across different software development contexts.

From an applied perspective, the strong performance of RFE+Extra Trees offers practical value for software

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

quality assurance workflows. The refined feature subset can be embedded into continuous integration pipelines to automate risk scoring for newly modified code modules, enabling QA teams to prioritize high-risk components and allocate testing resources more efficiently. However, the higher computational cost associated with RFE warrants careful consideration in large-scale or frequently updated systems. Overall, the results indicate that substantial performance improvements can be achieved without relying on complex ranking aggregation mechanisms, laying a strong foundation for developing more efficient and adaptive SDP pipelines.

V. Conclusion

This study conducts a comparative assessment of three feature-selection techniques: SelectKBest (SKB), Recursive Feature Elimination (RFE), and their combined SelectKBest+RFE variant to examine their influence on SDP models using the NASA MDP datasets. In contrast to previous work that primarily emphasizes classifier variations, this study foregrounds the contribution of feature selection to predictive performance across three tree-based models: Random Forest (RF), Extra Trees (ET), and Bagging (Decision Tree/DT). The use of stratified 10-fold cross-validation, together with the AUC metric, ensures consistent and reliable evaluation across datasets with diverse characteristics and varying degrees of class imbalance.

The experimental results indicate that the integration of RFE with the Extra Trees classifier produces the strongest predictive outcomes, attaining an average AUC of 0.7855 and outperforming both the SKB+RFE+ET and SKB+ET configurations. These findings highlight the effectiveness of iterative wrapper-based elimination guided by model-specific importance scores, which facilitates the selection of more stable and informative feature subsets. This advantage is particularly evident when combined with the Extra Trees algorithm's randomized structure. Additionally, the stability demonstrated by the RFE+Bagging (DT) configuration further supports the robustness of wrapper-based methods across heterogeneous datasets.

Beyond predictive accuracy, the computational cost analysis underscores meaningful trade-offs among the examined methods. SKB consistently exhibits the lowest runtime and memory consumption, making it a practical choice when computational resources are limited. In contrast, RFE and the hybrid SKB+RFE approach incur markedly higher runtimes due to their iterative elimination processes. These observations suggest that while wrapper-based strategies can enhance predictive performance, they require substantially greater computational effort, which is an important consideration for deployment in large-scale or real-time software engineering settings.

The study is restricted to the NASA MDP datasets and does not employ class-balancing techniques, or hyperparameter optimization. This design choice enables an isolated assessment of the intrinsic effects of feature-selection methods without confounding influences from parameter tuning or additional preprocessing. Future work is encouraged to incorporate explicit data-level balancing methods such as SMOTE, ADASYN, or Random Under Sampling, along with hyperparameter optimization techniques, including Grid Search or Bayesian Optimization, to address class imbalance better and derive more refined model configurations.

Overall, the findings confirm that pairing RFE with the Extra Trees classifier yields the strongest predictive performance, while the stability of RFE+Bagging(DT) and the efficiency of SKB illustrate meaningful trade-offs between accuracy and computational overhead. These insights provide a foundation for developing more adaptive, efficient, and reliable SDP frameworks in future research.

Acknowledgement

The authors would like to express sincere gratitude to the Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, for the invaluable support and resources provided throughout this research. The facilities, academic environment, and encouragement from members of the study program have significantly contributed to the completion of this work. This study would not have been possible without the institution's commitment to advancing research and innovation in software defect prediction.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors

Data Availability

The datasets analyzed in this study are publicly available from the NASA Metrics Data Program (MDP) and can be accessed at: https://github.com/klainfo/NASADefectDataset/tree/master. No new datasets were generated during the current study.

Author Contribution

Rahmayanti was responsible for drafting the manuscript, conducting the literature review, implementing the experimental framework, and performing the analysis and interpretation of the results. Rudy Herteno contributed by preparing and curating the datasets, providing methodological guidance, and supervising the overall research process. Setyo Wahyu Saputro contributed to the development and validation of the classification models and provided technical feedback to improve the experimental design. Triando Hamonangan Saragih and Friska Abadi contributed through a critical review of the manuscript, evaluation of the research methodology, and constructive suggestions to improve clarity, structure, and academic rigor. All authors reviewed and approved the final version of the manuscript and agreed to be responsible for all aspects of the work, ensuring integrity and accuracy.

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

Declarations

Ethical Approval

This study did not involve human participants or animal subjects. Therefore, ethical approval was not required.

Consent for Publication Participants

Not applicable.

Competing Interests

The authors declare no competing interests.

References

[1] F. Matloob, S. Aftab, M. Ahmad, M. A. Khan, A. Fatima, M. Iqbal, W. M. Alruwaili, and N. S. Elmitwally, "Software Defect Prediction Using Supervised Machine Learning Techniques: A Systematic Literature Review", *Intelligent Automation & Soft Computing*, vol. 29, no. 2, pp. 403-421, Jun. 2021, doi: 10.32604/iasc.2021.017562.

[2] L. Q. Chen, C. Wang, and S. L. Song, "Software Defect Prediction Based on Nested-Stacking and Heterogeneous Feature Selection", *Complex & Intelligent Systems*, vol. 8, pp. 3333-3348, Feb. 2022, doi: 10.1007/s40747-022-00676-y.

[3] A. O. Balogun, S. Basri, S. J. Abdulkadir, and A. S. Hashim, "Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach", *Applied Sciences*, vol. 9, no. 13, pp. 1-20, Jul. 2019, doi: 10.3390/app9132764.

[4] N. Krishnaveni, and V. Radha, "Feature Selection Algorithms for Data Mining Classification: A Survey", *Indian Journal of Science and Technology*, vol. 12, no. 6, pp. 1-11, Feb. 2019, doi: 10.17485/ijst/2018/v12i6/139581.

[5] N. Pudjihartono, T. Fadason, A. W. Kempa-Liehr, and J. M. O'Sullivan, "A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction", *Front. Bioinform*, vol. 2, Art. no. 927312, 2022, doi: 10.3389/fbinf.2022.927312.

[6] Y. Li, C. Y. Chen, and W. W. Wasserman, "Deep Feature Selection: Theory and Application to Identify Enhancers and Promoters", *Conference Paper*, pp. 1-13, 2015, doi: 10.13140/2.1.3673.6327.

[7] N. A. A. Khleel, and K. Nehez, "A New Approach to Software Defect Prediction Based on Convolutional Neural Network and Bidirectional Long Short-Term Memory", *Production Systems and Information Engineering*, vol. 10, no. 3, pp. 1-15, Nov. 2022, doi: 10.32968/psaie.2022.3.1.

[8] X. He, K. Zhao, and X. Chu, "AutoML: A Survey of the State-of-the-Art", *Knowledge-Based Systems*, vol. 212, no. 106622, pp. 1-35, Jan. 2021, doi: 10.1016/j.knosys.2020.106622.

[9] T. Sharma, A. Jatain, S. Bhaskar, and K. Pabreja, "An Empirical Analysis of Feature Selection Techniques for Software Defect Prediction", *Journal of Autonomous Intelligence*, vol. 7, no. 3, pp. 1-17, 2024, doi: 10.32629/jai.v7i3.1097.

[10] J. Suntoro, F. W. Christanto, and H. Indriyawati, "Software Defect Prediction Using AWEIG+ADACOST Bayesian Algorithm for Handling High Dimensional Data and Class Imbalance Problem", *Int. Journal of Information Technology and Business*, vol. 5, no. 1, pp. 27-32, Nov. 2022, doi: 10.24246/ijiteb.512018.27-32.

[11] A. O. Balogun, S. Basri, S. Mahamad, L. F. Capretz, A. A. Imam, M. A. Almomani, V. E. Adeyemo, and G. Kumar, "A Novel Rank Aggregation-Based Hybrid Multifilter Wrapper Feature Selection Method in Software Defect Prediction", *Computational Intelligence and Neuroscience*, vo. 2021, no. 1, Nov. 2021, doi: 10.1155/2021/5069016.

[12] V. Maulida, R. Herteno, D. Kartini, F. Abadi, and M. R. Faisal, "Feature Selection Using Firefly Algorithm With Tree-Based Classification In Software Defect Prediction ", *j.electron.electromedical.eng.med.inform*, vol. 5, no. 4, pp. 223-230, Aug. 2023, doi: 10.35882/jeeemi.v5i4.315.

[13] A. K. Aryanti, R. Herteno, F. Indriani, R. A. Nugroho, and M. Muliadi, "Implementation of Copeland Method on Wrapper-Based Feature Selection Using Random Forest For Software Defect Prediction", *ijeeemi*, vol. 7, no. 1, pp. 90–101, Feb. 2025, doi: 10.35882/2pgffc67.

[14] A. S. Nugraha, M. R. Faisal, F. Abadi, R. A. Nugroho, and R. Herteno, "DEEP NEURAL NETWORK ON SOFTWARE DEFECT PREDICTION", *JDSSE*, vol. 2, no. 02, pp. 82-89, Sep. 2021.

[15] F. Matloob, T. M. Ghazal, N. Taleb, S. Aftab, M. Ahmad, M. A. Khan, S. Abbas, and T. R. Soomro, "Software Defect Prediction Using Ensemble Learning: A Systematic Literature Review", *IEEE Access*, pp. 98754-98771, Jul. 2021, doi: 10.1109/ACCESS.2021.3095559.

[16] 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, Jan. 2020, doi: 10.1002/spe.2784.

[17] 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.

[18] H. Alsawalqah, N. Hijazi, M. Eshtay, H. Faris, A. A. Radaideh, I. Aljarah, and Y. Alshamaileh, "Software Defect Prediction Using Heterogeneous Ensemble Classification Based on Segmented Patterns", *Appl. Sci.*, vol. 10, no. 5, pp. 1-25, Mar. 2020, doi: 10.3390/app10051745.

[19] A. Ghavidel, P. Pazos, R. Del Aguila Suarez, and A. Atashi, "Predicting the Need for Cardiovascular

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

Surgery: A Comparative Study of Machine Learning Models", *j.electron.electromedical.eng.med.inform*, vol. 6, no. 2, pp. 92-106, Feb. 2024, doi: 10.35882/jeeemi.v6i2.359.

[20] D. P. H. Gray, "Software Defect Prediction Using Static Code Metrics: Formulating a Methodology", *Ph.D. dissertation, Univ. of Hertfordshire, Hatfield, UK*, Dec. 2012. [Online]. Available: https://uhra.herts.ac.uk/id/eprint/16494/1/04079420%20Gray%20David%20final%20PhD%20submission.pdf

[21] K. Marzuki, L. G. Rady Putra, H. Hairani, L. Z. A. Mardedi, and J. X. Guterres, "Performance Improvement of The Random Forest Method Based on Smote-Tomek Link on Lombok Tourism Analysis Sentiment", *BITe*, vol. 5, no. 2, pp. 151–158, Jan. 2024, doi: 10.30812/bite.v5i2.3166.

[22] 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", *j.electron.electromedical.eng.med.inform*, vol. 6, no. 3, pp. 276-288, May 2024, doi: 10.35882/jeeemi.v6i3.453.

[23] J. Kaliappan, A. R. Bagepalli, S. Almal, R. Mishra, Y.-C. Hu, and K. Srinivasan, "Impact of Cross-Validation on Machine Learning Models for Early Detection of Intrauterine Fetal Demise", *Diagnostics*, vol. 13, no. 10, pp. 1-22, May 2023, doi: 10.3390/diagnostics13101692.

[24] X. Duan, "Automatic Identification of Conodont Species Using Fine-Grained Convolutional Neural Networks", *Front. Earth Sci.* vol. 10, pp. 1-15, Jan. 2023, doi: 10.3389/feart.2022.1046327.

[25] S. Szeghalmy, and A. Fazekas, "A Comparative Study of the Use of Stratified Cross-Validation and Distribution-Balanced Stratified Cross-Validation in Imbalanced Learning", *Sensors*, vol. *23*, no. 4, pp. 1-27, Feb. 2023, doi: 10.3390/s23042333.

[26] N. R. Abid-Althaqafi and H. A. Alsalamah, "The Effect of Feature Selection on the Accuracy of X-Platform User Credibility Detection with Supervised Machine Learning", *Electronics*, vol. 13, no. 1, pp. 1-28, Jan. 2024, doi: 10.3390/electronics13010205.

[27] J. R. Vergara and P. A. Estevez, "A Review of Feature Selection Methods Based on Mutual Information", *Neural Comput & Applic*, vol. 24, no. 1, pp. 1-12, Jan. 2014, doi: 10.1007/s00521-013-1368-0.

[28] N. Papaioannou, G. Myllis, A. Tsimpiris, and V. Vrana, "The Role of Mutual Information Estimator Choice in Feature Selection: An Empirical Study on mRMR", *Information*, vol. 16, no. 9, pp. 1-25, Aug. 2025, doi: 10.3390/info16090724.

[29] O. Bulut, B. Tan, E. Mazzullo, and A. Syed, "Benchmarking Variants of Recursive Feature Elimination: Insights from Predictive Tasks in Education and Healthcare", *Information*, vol. 16, no. 4, pp. 1-21, Jun. 2025, doi: 10.3390/info16060476.

[30] T. A. Pham and V. Q. Tran, "Developing Random Forest Hybridization Models for Estimating the Axial Bearing Capacity of Pile", *PLoS ONE*, vol. 17, no. 3, Mar, 2022, doi: 10.1371/journal.pone.0265747.

[31] F. Tang, and H. Ishwaran, "Random Forest Missing Data Algorithms", *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 10, no. 6, pp. 363-377, Jun. 2017. doi: 10.1002/sam.11348.

[32] R. Supriyadi, W. Gata, N. Maulidah, and A. Fauzi, "Penerapan Algoritma Random Forest Untuk Menentukan Kualitas Anggur Merah", *E-Bisnis*, vol. 13, no. 2, pp. 67–75, Nov. 2020, doi: 10.51903/e-bisnis.v13i2.247.

[33] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely Randomized Trees", *Machine Learning*, vol. 63, no. 1, pp. 3-42, Mar. 2006, doi: 10.1007/s10994-006-6226-1.

[34] W. N. Hidayatullah, R. Herteno, M. R. Faisal, R. A. Nugroho, S. W. Saputro, and Z. B. Akhtar, "A Comparative Analysis of Polynomial-fit-SMOTE Variations with Tree-Based Classifiers on Software Defect Prediction", *j.electron.electromedical.eng.med.inform*, vol. 6, no. 3, pp. 289-301, Jul. 2024, doi: 10.35882/jeeemi.v6i3.455.

[35] Y. Lou, Y. Ye, Y. Yang, W. Zou, G. Wang, M. Strong, S. Upadhyaya, and C. Payne, "Individualized empirical baselines for evaluating the energy performance of existing buildings", *Science and Technology for the Built Environment*, vol. 29, no. 1, pp. 19-33, Oct. 2022, doi: 10.1080/23744731.2022.2134680.

[36] L. Breiman, "Bagging Predictors", *Machine Learning*, vol. 24, no. 2, pp. 123-140, Aug. 1996, doi: 10.1007/BF00058655.

[37] X. Wu, and J. Wang, "Application of Bagging, Boosting and Stacking Ensemble and EasyEnsemble Methods for Landslide Susceptibility Mapping in the Three Gorges Reservoir Area of China", *Int. J. Environ. Res. Public Health*, vol. 20, no. 6, pp. 1-18, Mar. 2023, doi: 10.3390/ijerph20064977.

[38] S. M. H. Kabir, M. T. Rahman, and A. H. Mridul, "Software Defect Prediction Using Traditional Machine Learning and Ensemble Learning Algorithms", *SWT*, vol. 1, pp. 1-16, May. 2025, doi: 10.47852/bonviewSWT52025645.

[39] V. K. R. R. Satuluri and V. Kumar, "Precision Insulin Delivery: Predictive Modelling for Bolus Insulin Injection in Real-Time", *IJACSA*, vol. 15, no. 2, pp. 292-302, Jan. 2024, doi: 10.14569/IJACSA.2024.0150231.

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

[40] A. D. Putri, F. Sholekhah, E. Dadynata, L. Efrizoni, R. Rahmaddeni, and N. Sapina, "Penerapan Algoritma Decision Tree C4.5 untuk Memprediksi Tingkat Kelangsungan Hidup Pasien Kanker Tiroid: The Application of C4.5 Decision Tree Algorithm for Predicting the Survival Rate of Thyroid Cancer Patients", *MALCOM*, vol. 4, no. 4, pp. 1485-1495, Sep. 2024, doi: 10.57152/malcom.v4i4.1532.

[41] A. Alazba, and H. Aljamaan, "Software Defect Prediction Using Stacking Generalization of Optimized Tree-Based Ensembles", *Appl. Sci.*, vol. 12, no. 9, pp. 1-20, Apr. 2022. Doi 10.3390/app12094577.

[42] B. Zhou, H. Zhao, Y. Wen, G. Ding, Y. Xing, X. Lin, and L. Xiao, "Software Defect Prediction Based on Semantic Views of Metrics: Clustering Analysis and Model Performance Analysis", *Comput. Mater. Contin.*, vol. 84, no. 3, pp. 5201–5221, Jul. 2025, doi: 10.32604/cmc.2025.065726.

[43] R. Herteno, M. R. Faisal, R. A. Nugroho, F. Abadi, and S. W. Saputro, "Agregasi Peringkat Berdasarkan Feature Filter Rangking Dalam Cross-Project Software Defects", *SINTECH Journal*, vol. 8, no. 1, pp. 1–11, Apr. 2025, doi: 10.31598/sintechjournal.v8i1.1763.

AUTHOR BIOGRAPHY

Rahmayanti is an undergraduate student in Computer Science at Lambung Mangkurat University. Her academic interests encompass software defect prediction, feature selection strategies, and machine-learning-based classification models. Her current work primarily investigates improvements in predictive performance through comparative evaluations of filter, wrapper, and hybrid feature selection approaches. Throughout her studies, she has actively participated in student organizations and diverse academic initiatives, thereby strengthening her technical competence, analytical reasoning, and collaborative abilities. She remains committed to contributing practical and research-oriented advancements within the broader domain of software engineering. She can be contacted at email: 2211016120010@mhs.ulm.ac.id.

Rudy Herteno earned his bachelor's degree in Computer Science from Lambung Mangkurat University in 2011. Following his graduation, he worked as a software developer for several years, gaining extensive professional experience in designing and implementing software systems. During this period, he contributed to the development of various software solutions, particularly those intended to support operational needs within local government institutions. In 2017, he completed his master's degree in Informatics at STMIK Amikom University. He currently serves as a lecturer in the Computer Science program at Lambung Mangkurat University. His research interests encompass software engineering, software defect prediction, and deep learning, with a focus on enhancing software quality, improving error detection mechanisms, and advancing artificial intelligence-driven solutions. He can be contacted at email: rudy.herteno@ulm.ac.id.

Setyo Wahyu Saputro is a lecturer in the Computer Science Department, Faculty of Mathematics and Natural Science, Lambung Mangkurat University in Banjarbaru. He earned his bachelor's degree in Computer Science from Lambung Mangkurat University in 2011 and subsequently completed his master's degree in Informatics at STMIK Amikom University in 2016. Since 2017, he has been actively engaged as an information technology practitioner and consultant, serving as a project manager and systems analyst for various government and private sector initiatives across South Kalimantan. His research interests encompass software engineering, human-computer interaction, and applications of artificial intelligence. He can be contacted at the email address setyo.saputro@ulm.ac.id.

Triando Hamonangan Saragih is a lecturer in the Computer Science Department, Faculty of Mathematics and Natural Science, Lambung Mangkurat University in Banjarbaru, where he is actively engaged in academic and research activities, with a strong focus on the broad field of Data Science. He earned his bachelor's degree in Informatics from Brawijaya University, Malang, in 2016, an accomplishment that laid the foundation for his subsequent academic trajectory. He later pursued and completed a master's degree in Computer Science at the same institution in 2018, further strengthening his expertise. His current research interests remain centered on Data Science and its associated analytical methodologies. He can be contacted at email: triando.saragih@ulm.ac.id.

Friska Abadi earned his bachelor's degree in Computer Science from Lambung Mangkurat University, Banjarbaru, Indonesia, in 2011, and subsequently completed his Master's degree in Informatics at STMIK Amikom Yogyakarta in 2016. He is currently serving as a lecturer in the Department of Computer Science, Faculty of Mathematics and Natural Sciences, Lambung Mangkurat University. His academic and research interests focus on data mining and software engineering, particularly in the application of computational methods for data analysis and software quality improvement. In addition to his teaching responsibilities, he is actively involved in academic supervision and research activities within the department. He has contributed to various academic publications and participated in technology-related

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294

research and community service programs. He can be contacted at email: friska.abadi@ulm.ac.id.

Corresponding author: Rudy Herteno, rudy.herteno@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Banjarbaru, Indonesia.
Digital Object Identifier (DOI): https://doi.org/10.35882/ijeeemi.v8i1.294