JURNAL INFOTEL
VOL. 17, NO. 1, FEBRUARY 2025, PP. 1–16. DOI:10.20895/INFOTEL.V17I1.1159

RESEARCH ARTICLE

Recursive Feature Elimination Optimization using Shapley Additive Explanations in Software Defect Prediction with LightGBM Classification

Hartati[1], Rudy Herteno[2, *], M. Reza Faisal[3], Fatma Indriani[4], and Friska Abadi[5]

[1,2,3,4,5]Faculty of Mathematics and Natural Sciences, Universitas Lambung Mangkurat, 70124, Indonesia

*Corresponding email: rudy.herteno@ulm.ac.id

Received: March 19, 2024; Revised: July 22, 2024; Accepted: October 07, 2024.

Abstract: Software defect refers to issues where the software does not function properly. The mistakes in the software development process are the reasons for software defects. Software defect prediction is performed to ensure the software is defect-free. Machine learning classification is used to classify defects in software. To improve the classification model, selecting the best features from the dataset is necessary. Recursive Feature Elimination (RFE) is a feature selection method. Shapley Additive Explanations (SHAP) is a method that can optimize feature selection algorithms to produce better results. This research will use the popular boosting algorithm LightGBM as a classifier to predict software defects. Meanwhile, RFE-SHAP will be used for feature selection to select the best subset of features. The results and discussion show that RFE-SHAP feature selection slightly outperforms RFE, with average AUC values of 0.864 and 0.858, respectively. This improvement proves that RFE-SHAP is better than a single RFE for feature selection. As for the T-Test significance result, although employing the feature selection method does not show significant results compared to the model without feature selection, it does yield better results than other research based on the AUC result.

Keywords: feature selection, LightGBM, RFE, SHAP, software defect

1 Introduction

Software quality remains crucial in the current technological era. Software products must be devoid of defects to guarantee their optimal use. If the software product contains defects, it can lead to actual financial losses. Handling software defects can be a time time-consuming process and can result in significant costs. According to [1], errors in writing codes during software development can lead to software defects. To avoid these defects, [2] stated that it is necessary to predict software defects during the software development phase. However, several problems are often encountered in software defect prediction, including high-dimensional datasets.

Feature selection is a solution to reduce the problem of high-dimensional datasets and has been widely used in software defect prediction [3], [4]. One of the most popular feature selection methods is wrapper-based feature selection, as conducted in [5]. Another example of a wrapper feature selection method is Recursive Feature Elimination (RFE) [6]. Recursive Feature Elimination is performed by eliminating some features to generate optimal feature sets [7]. Recursive Feature Elimination has been proven to provide good performance as a feature selection method in some studies, such as in a study conducted by [2]. However, although feature selection methods have proven useful, they can still be optimized to produce better performance [8].

Shapley Additive Explanations (SHAP) provides an interpretive framework for machine learning model performance based on the game theory approach [9]. In SHAP, each feature is assigned a feature importance by using Shapely values [10]. SHAP can be combined with feature selection methods and is proven to provide better performance than using only feature selection without SHAP [8]. One of the feature selection algorithms that can be optimized with SHAP is RFE. The RFE-SHAP combination is proven to be a feature selection model with better performance than the single RFE method [11].

A machine learning classifier approach categorizes the software dataset into defective and non-defective groups [12]. Boosting Algorithms are widely used for classification problems and have proven to perform well. Light Gradient Boosting Machine (LightGBM) is one of the algorithms included in the decision tree-based boosting algorithms that effectively reduce computational costs while maintaining good accuracy [13]. LightGBM's good performance was proven in [14] compared to the CatBoost, XGBoost, and LightGBM methods. LightGBM has been proven to have faster and more accurate results.

Based on the description above, this research proposes the Recursive Feature Elimination (RFE) method optimized with Shapley Additive Explanations (SHAP). LightGBM will be the learning algorithm in RFE and become the final classification model after the feature selection process. In this study, the data set used is NASA MDP. The RFE-SHAP method with the LightGBM classifier has never been implemented on the NASA MDP dataset before.

2 Research Method

2.1 Collect Dataset

In this study, the dataset used is NASA MDP D" obtained from [15] which performs processing on NASA MDP datasets to overcome various problems in previous datasets. The NASA MDP datasets include thirteen datasets. However, this study utilized only five

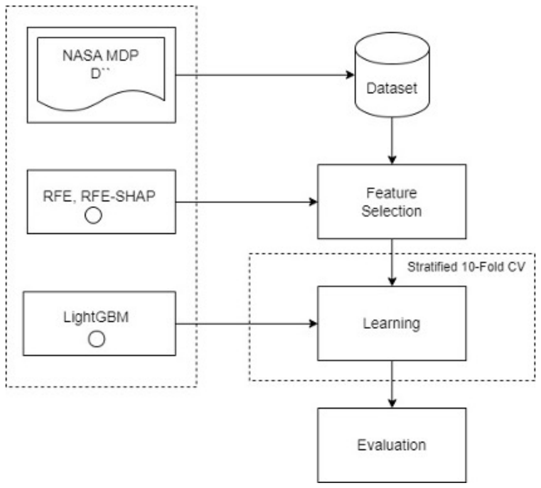

Figure 1: Research methodology.

datasets with 37 identical attributes, which are CM1, MW1, PC, PC3, and PC4, similar to research conducted by [16]. As research by [17], the number of defective and non-defective classes of every dataset conducted in this study are summarized in Table 1.

Table 1: NASA dataset

Dataset	All samples number	Defective samples number	Nondefective samples number
CM1	498	49	449
MW1	253	27	226
PC1	1109	77	1032
PC2	745	16	729
PC4	1287	177	1110

2.2 LightGBM

According to [18], who proposed LightGBM, LightGBM is a method that implements Gradient Boosting Decision Tree (GBDT). LightGBM is a machine-learning model that combines the predictions generated by numerous decision trees. In the context of LightGBM, two specific techniques are employed: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS aims to handle a large number of instances, while EFB seeks to handle a large number of features. LightGBM has the advantages of a fast and more efficient training process, low memory usage, better accuracy results, and the ability to handle large datasets.

According to [19] the results of the trained decision trees are used to predict the residuals of the previous model. Suppose the intention is to construct a LightGBM model, with the variables T representing the number of trees and n denoting the number of dataset

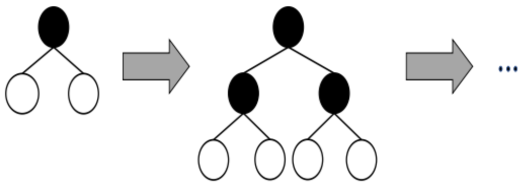

Figure 2: Level-wise approach.

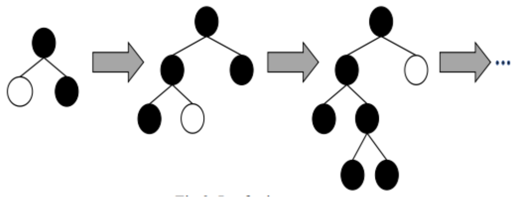

Figure 3: Leaf-wise approach.

samples. The additive training procedure may be represented by Equation 1.

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{t-1} + f_t(x_i) \tag{1}$$

Where $\hat{y}_i^{(t)}$ represents the estimated value of the i-th sample at the t-th iteration and f_t represents the function acquired for the decision tree indexed by t. Throughout the iterations, the current model $\hat{y}_i$ will persist while incorporating a novel function f (or learned residuals) into the model.

LightGBM utilizes a feature importance score, a numerical value that signifies the significance of a feature that has been trained through the implementation of LightGBM. One of LightGBM's feature importance score types is Split, representing the frequency with which each feature is utilized to split the training dataset throughout all trees [20]. As mentioned in [21], LightGBM employs an improved histogram approach. The enhancement of training speed and the efficiency of space consumption can be achieved by exclusively choosing optimal segmentation points within k intervals. The decision tree construction in LightGBM differs from the level-wise strategy by employing the leaf-wise strategy. This strategy enhances the maximum depth limit while ensuring high efficiency to prevent overfitting and reduce training data. The Level-Wise approach and Level-Wise approach are shown in Figure 2 and Figure 3, respectively.

Most decision tree algorithms adopt a level-wise tree growth strategy. However, splitting leaves with a lower splitting gain is redundant and consumes excessive computational resources. On the contrary, tree growth in a leaf-wise strategy involves identifying the leaf that offers the highest splitting gain among all leaves, splitting it, and subsequently iterating this process [22].

2.3 Feature Selection

This research will have two feature selection scenarios. The first uses the Recursive Feature Elimination (RFE) algorithm, and the second uses RFE feature selection optimized with SHAP, namely RFE-SHAP.

Recursive Feature Elimination is a wrapper technique that incorporates a greedy algorithm during its implementation. The wrapper method selects a subset of features from the feature sets used after training the model. The RFE algorithm starts by using all the features and then eliminates features in each iteration until the specified number of features remains. The features selected for elimination are determined based on the evaluation results of the classification [2].

As stated by [23], RFE selects the best feature subset while eliminating based on feature ranking. The procedure for the feature selection process using RFE is as follows:

1. Train a classifier.
2. Compute the ranking criteria for each feature from feature importance.
3. Eliminate the feature with the lowest feature importance.

The RFE procedure will be repeated until the specified number of feature subsets persists. During each iteration, a single feature or multiple features may be eliminated. In terms of computation, removing features one by one will be more expensive. The computational cost will be more efficient if various features are removed simultaneously. However, this may lead to a degradation in the model's performance. Eliminating features one by one will result in ranking each feature in each iteration. On the other hand, eliminating several features at once will result in ranking subset features in each iteration. The feature elimination process used in this research will eliminate one feature in each iteration.

SHAP (Shapley Additive exPlanations) is a unified framework for interpreting the prediction results of a model. SHAP applies the marginal effects of features to explain the result of every machine learning model. Understanding how the model makes some predictions is very important. It is common to use complex models, such as ensemble methods, to achieve good accuracy results. Complex models are usually difficult to interpret. Therefore, SHAP was designed to overcome this problem. SHAP applies Shapley values to interpret the model. SHAP assigns a weight value to each feature for a particular prediction. SHAP aims to improve the interpretation of machine learning models by generating the partial contribution of each feature to model predictions [10].

According to [21], the fundamental principle underlying the SHAP method concept is rooted in the Shapley value, a well-known concept in combinatorial game theory. The definition of the Shapley value for a feature can be expressed as the difference between the previous contribution of the features set and the contribution made by the new feature. When calculating the Shapley value for feature D, given that S represents the set of features and v(S) is the set contribution, the estimation of the Shapley value can be achieved by employing the Equation 2.

$$\phi_D = v(S \cup \{D\}) - v(S) \tag{2}$$

Due to the excessive complexity of the original model, it cannot be employed as an interpretive model for integration models and neural networks. Consequently, an interpretation model with a more concise definition is utilized to interpret the original model. The contribution of additive features can be expressed mathematically as a linear function, wherein a binary variable is used to represent the independence from the model. This is shown in the Equation 3.

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{3}$$

Where g is the explanatory model, M represents the set of features, ϕ_0 represents the average value of the input data employed in the model's predictive process, and z'_i is a binary value (0 or 1), which signifies the existence or nonexistence of the feature. The determination of the significance of a specific feature can be achieved through the utilization of Equation 4. Initially, a search is conducted to identify all potential combinations of features that exclude the particular feature. Then, the Shapley value is derived by computing the combinations that align with the specific feature. Subsequently, the value is assigned a weight to contribute to the overall total.

$$\phi_j = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!\,(p-|S|-1)!}{p!} \big(\text{val}(S \cup \{x_j\}) - \text{val}(S)\big) \tag{4}$$

Where S is the subset of features used in the model, x is the sample eigenvalue vector to be interpreted, p is the number of features, val(S) represents the output value of the model when S combinations of features are considered, the number of combinations when p features are accounted for as represented by $p!$, after selecting feature j, the remaining number of combinations is $|S|!(p-|S|-1)!$.

Based on the research conducted by [21], the steps of RFE-SHAP are as follows:

1. Train a model.
2. Compute feature weights using Shapley values.
3. Rank features based on the obtained Shapley values and eliminate features with the smallest Shapley values.

2.4 Performance Measure

The evaluation method used in this research is Area Under Curve (AUC). In addition, a paired T-test will be conducted to determine the significance of the RFE and RFE-SHAP feature selection results compared to the model without feature selection.

AUC serves as a metric for assessing the effectiveness of a classifier model in terms of accuracy. AUC is the area under the Receiver Operating Characteristic (ROC) curve [24]. ROC curve is a valuable tool for evaluating and quantifying the discriminative capacity of predictive models. However, it is worth noting that this particular metric has faced criticism about its clinical relevance and the absence of an intuitive interpretation. Despite these concerns, the ROC curve remains a prominent method employed to assess the performance of predictive models [25].

AUC is the preferred metric for evaluating the effectiveness of a classifier when dealing with imbalanced datasets [26]. AUC provides a single numerical value that falls from 0 to 1. The utilization of the ROC-AUC curve serves to distinguish and analyze the trade-off that exists between the true positive rate (TPR) and the false positive rate (FPR) [27]. AUC can be determined by calculating the integral of the curve, thereby determining the total area enclosed. It is important to note that a greater area indicates a higher performance level. The AUC value, when it is equal to 1, serves as an indication of optimal performance. On the other hand, when the AUC value is equal to 0, it indicates the worst possible performance [28]. According to [29], the interpretation guidelines for AUC as a measure are shown in Table 2.

Table 2: Performance interpretation based on AUC

AUC range	Evaluation
AUC = 0.5	Totally random, as good as tossing coin
$0.5 < \text{AUC} < 0.7$	Poor, not much better than a coin toss
$0.7 \leq \text{AUC} < 0.8$	Acceptable
$0.8 \leq \text{AUC} < 0.9$	Excellent
$0.9 \leq \text{AUC}$	Outstanding

Based on the research of [30], the AUC measure is obtained from calculations using ROC curve analysis using Equation 5.

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

To determine the significance of the performance results between the model after feature selection and the model without feature selection is achievable through the use of a T-Test. The T-test is a statistical analysis method that evaluates the presence between the dependent and independent variables. This test serves as a tool to measure the strength and importance of the correlation between the pair of variables. By examining the data and calculating the T-value, the T-test determines whether the observed association is statistically significant or not [31].

The type of T-test used in this research is the Paired T-test, and the level of significance (α) chosen is 0.05. For assessment based on t_{count} and t_{table}, if the value of $t_{\text{count}} > t_{\text{table}}$, then the null hypothesis H_0 is rejected and the alternative hypothesis H_1 is accepted. For assessment using P-value, if the P-value $< \alpha$, the null hypothesis H_0 is rejected, and the alternative hypothesis H_1 is accepted. The null hypothesis H_0 indicates the absence of a statistically significant difference between the model with feature selection and the model without feature selection. In contrast, the alternative hypothesis H_1 indicates a statistically significant difference between the model with feature selection and the model without feature selection. One way to get T-Test results can be done using tools such as Excel. The T-test conducted in this research uses Excel.

3 Results

This research uses five NASA MDP D" datasets (CM1, MW1, PC1, PC3, PC4). The conducted experiments within this study will evaluate three models, specifically the model without applying feature selection, the model using the RFE feature selection algorithm, and the model using the RFE-SHAP feature selection algorithm. All trials will use the same classification model, namely LightGBM. The evaluation method used is AUC.

3.1 Analysis Without Feature Selection

It is necessary to know how the model performs without feature selection to see how it performs after feature selection. The model will be built using the LightGBM method for classification. The method employed for validation is Stratified 10-fold Cross Validation. Meanwhile, the evaluation method used is AUC.

In the first step, the dataset was partitioned into distinct subsets consisting of 10 folds, all of equal proportions, achieved through applying the Stratified 10-fold Cross Validation technique. This technique ensured that each fold maintained a balanced distribution of instances across the various classes, thereby enhancing the robustness and reliability of the model evaluation process.

The data partitioning resulted in 10-fold data. Nine folds of the 10-fold data are used as training data, and one fold is used as testing data. After the Stratified 10-fold Cross Validation stage, the dataset will be used for the classification process using the LightGBM method. After the classification process, the next step is to evaluate the result with AUC. The results of the classification and evaluation process without feature selection for 5 NASA MDP D" datasets are shown in Table 3.

Table 3: AUC result for the model without feature selection

Dataset	AUC
CM1	0,795
PC1	0,880
PC3	0,817
PC4	0,941

3.2 Analysis Feature Selection

The feature selection stage conducted in this research consists of two experimental scenarios. The first experiment performed feature selection with RFE, and the second performed feature selection with RFE optimized with SHAP, namely RFE-SHAP.

The first scenario is to perform feature selection with RFE. The first step of RFE involved training the classifier using all features in each dataset without partitioning the data into training and test sets. The classifier utilized in this study is LightGBM. Each feature was ranked based on its importance value, which resulted after fitting the data using LighGBM. LightGBM defaulted to the Split metric, representing the frequency of a feature's uses in splitting nodes throughout the decision tree.

A higher Split value indicates greater feature importance. Subsequently, the feature with the lowest importance value was eliminated, while the remaining features proceeded to the next stage. The subset of the remaining features will be used in the next stage. The remaining subset of features will be trained again using the classifier, this time using the Stratified 10-fold CV validation technique for data partitioning. After that, the evaluation was performed using AUC.

To determine how many feature subsets perform best in RFE feature selection, all processes will be tried on all possible feature subsets in each dataset. The experiment will stop if there are two remaining feature subsets because if only one feature, it cannot be used for classification.

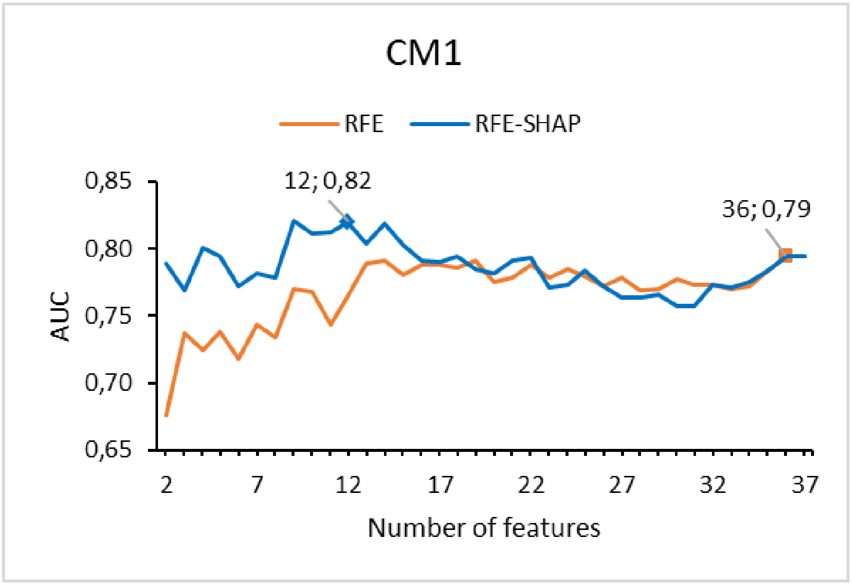

Figure 4: CM1 result.

In the second scenario, feature selection was conducted employing RFE-SHAP. This feature selection process is similar to RFE. Unlike RFE, which evaluates and prioritizes features according to their importance as determined by the learning model, the RFE-SHAP method diverges in its approach by integrating the SHAP technique. This method involves the assignment of specific weightings to each feature within the dataset, employing the Shapley value to derive these weightings. The RFE-SHAP algorithm starts by training the model using all feature subsets. This initial training phase involves using all data samples without partitioning them into training and test sets. After training the model, SHAP is employed to compute weight values for all features. After obtaining the weight values, these weight values are sorted in descending order. The feature with the lowest weight value is removed.

The remaining subset of features will be retrained using the classifier, and this time, the Stratified 10-fold CV validation technique will be implemented. After that, the model will be evaluated using AUC. This process is iteratively applied to all conceivable feature subsets within each dataset to obtain the optimal evaluation results. The experiment will stop if the remaining subset of features amounts to two, as utilizing a feature subset of one would render it unsuitable for classification. The results of RFE and RFE-SHAP experiments on 5 NASA MDP D“ datasets are visually presented in Figure 4, 5, 6, 7, and 8.

4 Discussion

The outcomes of RFE and RFE-SHAP will be compared with models without feature selection. The evaluation will be conducted based on the utilization of the number of features used and the AUC results of each model. Firstly, the number of features acquired after

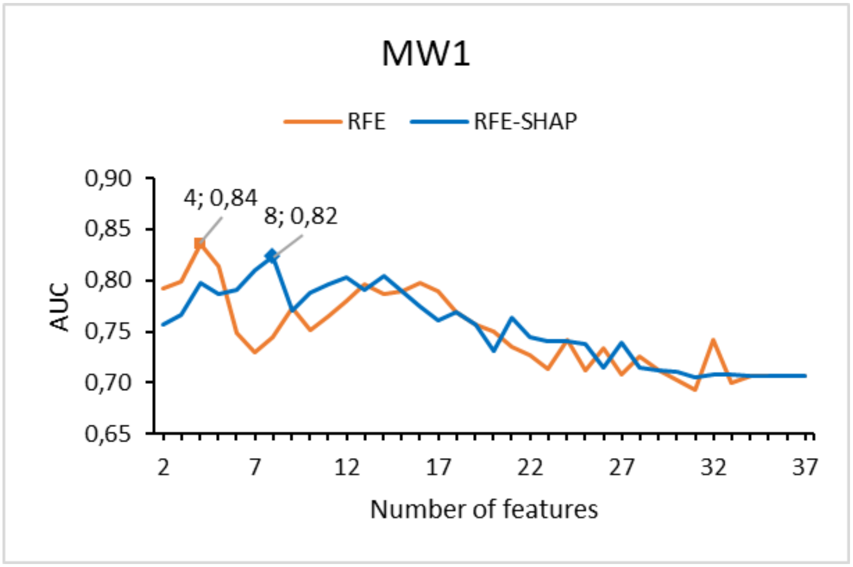

Figure 5: MW1 result.

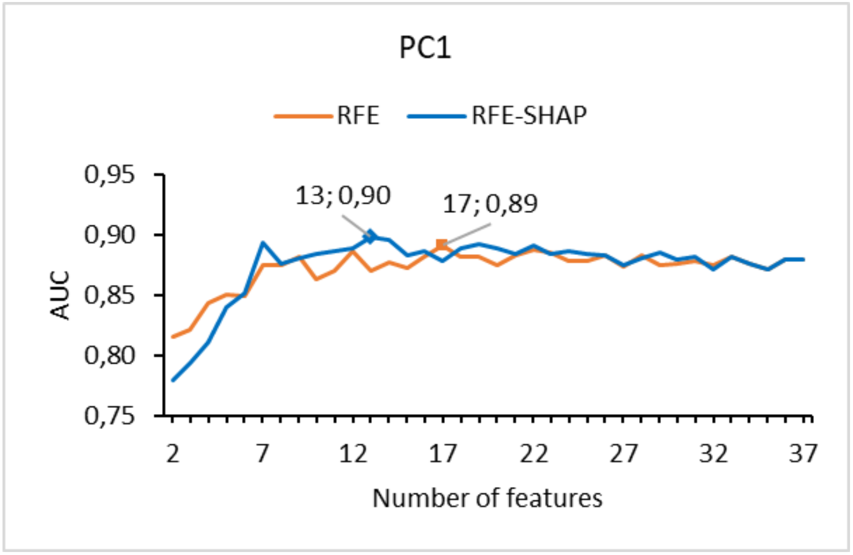

Figure 6: PC1 result.

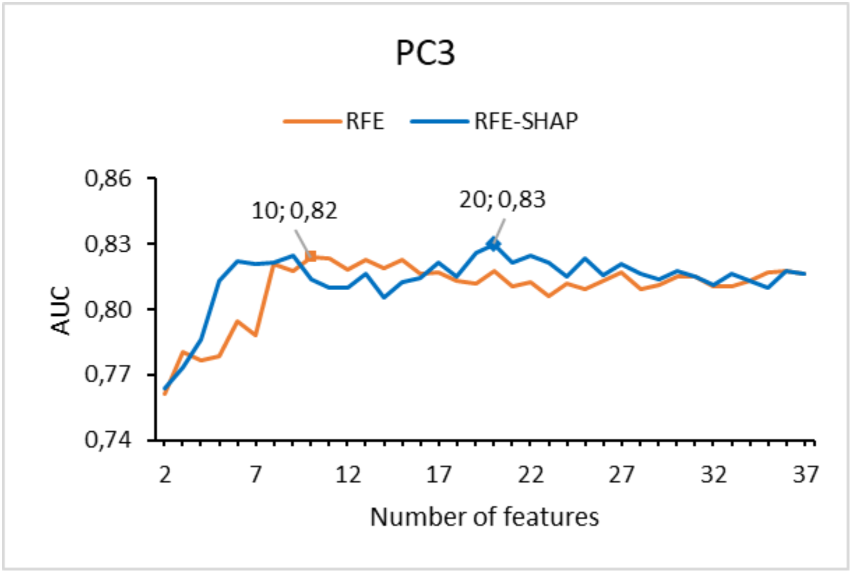

Figure 7: PC3 result.

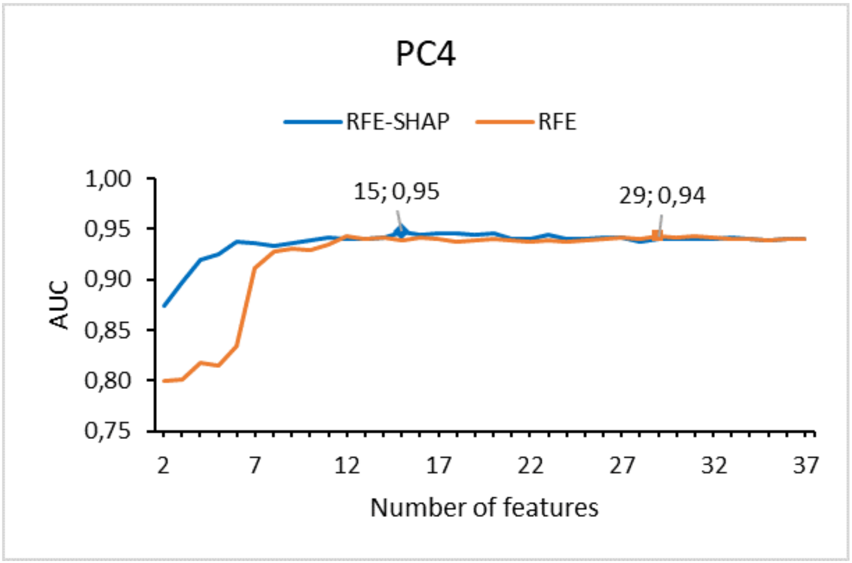

Figure 8: PC4 result.

applying RFE and RFE-SHAP feature selection will be compared. The results are shown in Table 4. The numbers in bold indicate the highest values.

Table 4: Comparison of the number of features used before and after feature selection

Dataset	Without feature selection	RFE	RFE-SHAP
CM1	37	36	**12**
MW1	37	4	8
PC1	37	17	**13**
PC3	37	10	20
PC4	37	29	**15**

The AUC evaluation results for the model without feature selection will be compared with the model using RFE and RFE-SHAP feature selection. The results are shown in Table 5. The numbers in bold indicate the highest values

Table 5: AUC Results

Dataset	Without feature selection	RFE	RFE-SHAP
CM1	0,795	0,795	**0,821**
MW1	0,706	0,836	0,823
PC1	0,880	0,892	**0,899**
PC3	0,817	0,824	**0,830**
PC4	0,941	0,943	**0,948**
Average	**0,827**	**0,858**	**0,864**

Figure 9 represents the graphical form of the different AUC results for all datasets based on Table 5. It can be seen that 4 out of 5 datasets in the RFE-SHAP method have the best outcomes compared to other models.

The proposed method, RFE-SHAP, performs best compared to the model without feature selection and RFE feature selection. Whether the model exhibits a significant change or not after feature selection can be determined by conducting a T-Test. Therefore, a T-Test will be performed to evaluate the significance of AUC results for models without feature selection and that use RFE and RFE-SHAP feature selection. The RFE vs the model without feature selection T-Test results are shown in Table 6. For the RFE-SHAP vs. the model without feature selection, T-test results are shown in Table 7.

Table 6: RFE vs. Without feature selection AUC T-Test result

Paired T-test	
df	4
t_{count}	1.215
t_{table}	2.776
P-value	0.291
α	0.05

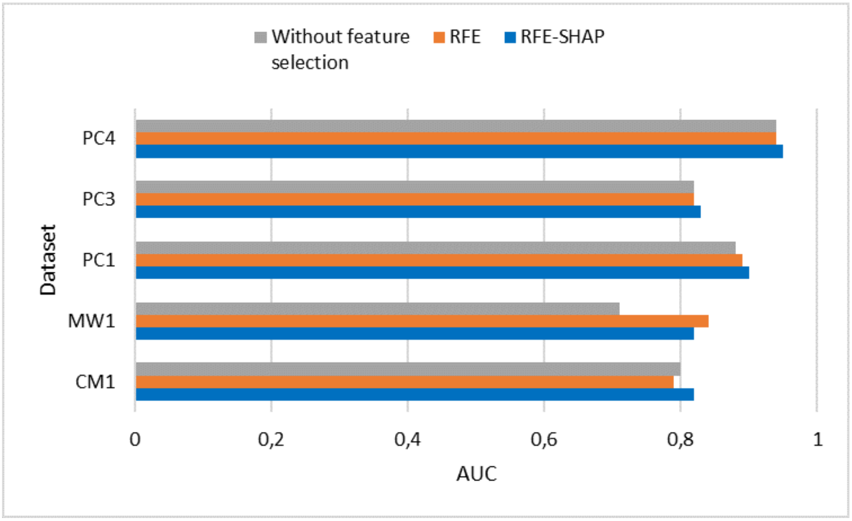

Figure 9: AUC result.

Table 7: RFE-SHAP vs. Without feature selection AUC T-Test result

Paired T-test	
df	4
t_{count}	1.788
t_{table}	2.776
P-value	0.148
α	0.05

It can be seen that $t_{\text{count}} < t_{\text{table}}$ for the results in Table 6 and Table 7. Because $t_{\text{count}} < t_{\text{table}}$, the null hypothesis H_0 is accepted and the alternative hypothesis H_1 is rejected in both table. Meanwhile, it can also be seen that the P-value $> \alpha = 0.05$ in both tables, which means that H_0 is accepted. H_0 is accepted, indicating no significant difference between the RFE results and the model without feature selection.

Table 8 compares the AUC performance of the proposed method and other studies in software defect prediction proposed by [4] and [5]. Research conducted by [4] proposed Particle Swarm Optimization (PSO) for feature selection with Random Under Sampling (RUS) and SMOTE to handle imbalanced data. The research also used ensemble bagging with the Naïve Bayes Classifier. In addition, research by [5] proposed an enhanced wrapper-based feature selection (EWFS) with Naïve Bayes and Decision Tree classifier.

It can be seen that the proposed model, RFE-SHAP feature selection, has higher results than the enhanced wrapper feature selection in research conducted by [5]. The proposed model also outperforms the combination of feature selection and RUS or SMOTE sampling methods in [4]. Therefore, even though there is no imbalanced data handling in this research, the results still outperform those of other studies. The bold values in Table 6 show the highest score among other results.

Table 8: Comparison of the AUC results of the proposed method with other studies

Dataset	Proposed Model	[4]		[5]	
		PSO + SMOTE + BG + NB	EWFS + NB	EWFS + DT	PSO + RUS + BG + NB
CM1	0,821	0,733	0,722	0,509	0,741
MW1	0,823	0,806	0,756	0,536	0,776
PC1	0,899	0,84	0,826	0,726	0,869
PC3	0,830	-	0,806	0,687	-
PC4	0,948	0,844	0,845	0,878	0,82

5 Conclusion

Based on the conducted research, it can be concluded that the performance results of Recursive Feature Elimination (RFE) with LightGBM classification on 5 NASA MDP D" datasets have an average AUC of 0.858. Meanwhile, the performance results of Recursive Feature Elimination optimized by Shapley Additive Explanations (RFE-SHAP) with LightGBM classification on 5 NASA MDP D" datasets have an average AUC of 0.864. RFE-SHAP used a fewer number of features than RFE in 3 out of 5 datasets; CM1, PC1, and PC4. All the results prove that RFE-SHAP performs slightly better than RFE while using fewer features. Even though the T-Test showed that both RFE and RFE-SHAP have no significant result compared to the model without feature selection, RFE-SHAP has outperformed other research based on AUC results comparison.

References

[1] M. K. Thota, F. H. Shajin, P. Rajesh, *et al.*, "Survey on software defect prediction techniques," *International Journal of Applied Science and Engineering*, vol. 17, no. 4, pp. 331–344, 2020.

[2] S. Mehta and K. S. Patnaik, "Improved prediction of software defects using ensemble machine learning techniques," *Neural Comput. Appl.*, vol. 33, pp. 10551–10562, Aug. 2021.

[3] A. O. Balogun, S. Basri, S. Mahamad, S. J. Abdulkadir, M. A. Almomani, V. E. Adeyemo, Q. Al-Tashi, H. A. Mojeed, A. A. Imam, and A. O. Bajeh, "Impact of feature selection methods on the predictive performance of software defect prediction models: An extensive empirical study," *Symmetry (Basel)*, vol. 12, p. 1147, July 2020.

[4] A. Suryadi, "Integration of feature selection with data level approach for software defect prediction," *SinkrOn*, vol. 4, p. 51, Sept. 2019.

[5] A. O. Balogun, S. Basri, L. F. Capretz, S. Mahamad, A. A. Imam, M. A. Almomani, V. E. Adeyemo, A. K. Alazzawi, A. O. Bajeh, and G. Kumar, "Software defect prediction using wrapper feature selection based on dynamic re-ranking strategy," *Symmetry (Basel)*, vol. 13, p. 2166, Nov. 2021.

[6] A. Habibi, M. R. Delavar, M. S. Sadeghian, B. Nazari, and S. Pirasteh, "A hybrid of ensemble machine learning models with RFE and boruta wrapper-based algorithms for

flash flood susceptibility assessment," *Int. J. Appl. Earth Obs. Geoinf.*, vol. 122, p. 103401, Aug. 2023.

[7] C. A. Ramezan, "Transferability of recursive feature elimination (RFE)-derived feature sets for support vector machine land cover classification," *Remote Sens. (Basel)*, vol. 14, p. 6218, Dec. 2022.

[8] A. Gramegna and P. Giudici, "Shapley feature selection," *FinTech*, vol. 1, pp. 72–80, Feb. 2022.

[9] S. Mangalathu, S.-H. Hwang, and J.-S. Jeon, "Failure mode and effects analysis of RC members based on machine-learning-based SHapley additive explanations (SHAP) approach," *Eng. Struct.*, vol. 219, p. 110927, Sept. 2020.

[10] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," May 2017.

[11] P. Rodríguez, A. Villanueva, L. Dombrovskaia, and J. P. Valenzuela, "A methodology to design, develop, and evaluate machine learning models for predicting dropout in school systems: the case of chile," *Educ. Inf. Technol.*, vol. 28, pp. 1–47, Jan. 2023.

[12] A. I. M. L. Artificial Intelligence, E. Dada, D. Oyewola, S. Joseph, A. Dauda, S. Bassi, and A. Baba, "Ensemble machine learning model for software defect prediction," vol. 2, pp. 11–21, 07 2021.

[13] H. Jafarzadeh, M. Mahdianpari, E. Gill, F. Mohammadimanesh, and S. Homayouni, "Bagging and boosting ensemble classifiers for classification of multispectral, hyperspectral and PolSAR data: A comparative evaluation," *Remote Sens. (Basel)*, vol. 13, p. 4405, Nov. 2021.

[14] E. Al Daoud, "Comparison between xgboost, lightgbm and catboost using a home credit dataset," *International Journal of Computer and Information Engineering*, vol. 13, no. 1, pp. 6–10, 2019.

[15] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," *IEEE Trans. Softw. Eng.*, vol. 39, pp. 1208–1215, Sept. 2013.

[16] W. Ustyannie, E. Setyaningsih, and C. Iswahyudi, "Optimization of software defects prediction in imbalanced class using a combination of resampling methods with support vector machine and logistic regression," *J. Infotel*, vol. 13, pp. 176–184, Dec. 2021.

[17] M. S. Alkhasawneh, "Software defect prediction through neural network and feature selections," *Appl. Comput. Intell. Soft Comput.*, vol. 2022, pp. 1–16, Sept. 2022.

[18] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[19] T. Chen, J. Xu, H. Ying, X. Chen, R. Feng, X. Fang, H. Gao, and J. Wu, "Prediction of extubation failure for intensive care unit patients using light gradient boosting machine," *IEEE Access*, vol. 7, pp. 150960–150968, 2019.

[20] C. Tang, N. Luktarhan, and Y. Zhao, "An efficient intrusion detection method based on LightGBM and autoencoder," *Symmetry (Basel)*, vol. 12, p. 1458, Sept. 2020.

[21] D. Meng, H. Dai, Q. Sun, Y. Xu, and T. Shi, "Novel wireless sensor network intrusion detection method based on lightgbm model.," *IAENG International Journal of Applied Mathematics*, vol. 52, no. 4, 2022.

[22] Y. Zhang, C. Zhu, and Q. Wang, "LightGBM-based model for metro passenger volume forecasting," *IET Intell. Transp. Syst.*, vol. 14, pp. 1815–1823, Dec. 2020.

[23] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, pp. 389–422, 2002.

[24] R. S. Wahono, N. S. Herman, and S. Ahmad, "A comparison framework of classification models for software defect prediction," *Adv. Sci. Lett.*, vol. 20, pp. 1945–1950, Oct. 2014.

[25] A. C. J. W. Janssens and F. K. Martens, "Reflection on modern methods: Revisiting the area under the ROC curve," *Int. J. Epidemiol.*, vol. 49, pp. 1397–1403, Aug. 2020.

[26] T. Yang and Y. Ying, "AUC maximization in the era of big data and AI: A survey," *ACM Comput. Surv.*, vol. 55, pp. 1–37, Aug. 2023.

[27] R. Malhotra and K. Khan, "A study on software defect prediction using feature extraction techniques," in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, IEEE, June 2020.

[28] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," in *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*, (New York, NY, USA), ACM, Nov. 2020.

[29] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. Wiley Series in Probability and Statistics, Hoboken, NJ: Wiley-Blackwell, 3 ed., Mar. 2013.

[30] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, (New York, NY, USA), ACM, May 2014.

[31] F. Gorunescu, *Data Mining*. Intelligent systems reference library, Berlin, Germany: Springer, 2011 ed., June 2011.