Journal of Electronics, Electromedical Engineering, and Medical Informatics
Multidisciplinary: Rapid Review: Open Access Journal **Vol. 5, No. 4, October 2023, pp: 223-230; eISSN: 2656-8632**

RESEARCH ARTICLE **OPEN ACCESS**

Manuscript received July 27, 2023; revised August 20, 2023; accepted September 21, 2023; date of publication October 30, 2023
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeemi.v5i4.315

How to cite: Vina Maulida, Rudy Herteno , Mohammad Reza Faisal , Dwi Kartini , Friska Abadi, "Feature Selection Using Firefly Algorithm with Tree-Based Classification In Software Defect Prediction, vol. 5, no. 4, pp. 223-230, October 2023.

Feature Selection Using Firefly Algorithm with Tree-Based Classification in Software Defect Prediction

Vina Maulida, Rudy Herteno, Mohammad Reza Faisal, Dwi Kartini, Friska Abadi

Fakultas Matematika dan Ilmu Pengetahuan Alam , Universitas Lambung Mangkurat , Indonesia
Corresponding author: rudy.herteno@ulm.ac.id

ABSTRACT Defects that occur in software products are a universal occurrence. Software defect prediction is usually carried out to determine the performance, accuracy, precision and performance of the prediction model or method used in research, using various kinds of datasets. Software defect prediction is one of the Software Engineering studies that is of great concern to researchers. The purpose of this research is to improve the performance produced by the Decision Tree, Random Forest, and Deep Forest classification methods by selecting the Firefly feature in predicting software defects. In addition, it is also to find out a tree-based classification algorithm with Firefly feature selection that can provide better software defect prediction performance. The dataset used in this study is the ReLink dataset which consists of Apache, Safe and Zxing. Then the data is divided into testing data and training data with 10-fold cross validation. Then feature selection is performed using the Firefly Algorithm. Each ReLink dataset will be processed by each tree-based classification algorithm, namely Decision Tree, Random Forest and Deep Forest according to the results of the Firefly feature selection. Performance evaluation uses the AUC value (Area under the ROC Curve). Research was conducted using google collab and the average AUC value generated by Firefly-Decision Tree is 0.66, the average AUC value generated by Firefly-Random Forest is 0.77, and the average AUC value generated by Firefly-Deep Forest is 0.76. The results of this study indicate that the approach using the Firefly algorithm with Random Forest classification gets better results compared to other tree-based algorithms.

INDEX TERMS Software Defect Prediction, Firefly, Decision tree, Random forest, Deep forest

I. INTRODUCTION

Software systems continue to serve important functions in every aspect of our society, the presence of a flaw in such a system can have a major impact on the economy and the general population[1]. Software development projects necessitate a phase of software testing which is of utmost importance and incurs significant costs for investigating the efficacy of the resultant product[2]. A software defect denotes a flaw, error, bug, mistake, fault, or failure within a computer system or program that may result in an unexpected or inaccurate outcome or hinder intended software behavior[3]. To attain high-quality software, the final product must have minimal defects. Early detection of software defects can lead to reduced development costs, rework efforts, and more dependable software[4]. Defect prediction is an exceedingly dynamic domain within software analytics[5]. The utilization of software defect prediction metrics is of parfrequency significance in the development of a prediction model, which has the objective of enhancing software quality by foreseeing a maximal number of software defects[6].

In research conducted by Andini et al[7]. in his research using a tree-based classification with hyperparameter tuning, the average AUC value generated by a Decision Tree is 0.69, while the average AUC value generated by a Random Forest is 0.76 and the average AUC value produced by Deep Forest is 0.79. In another study by Anbu et al. the Firefly optimization method was used to improve software defect prediction performance as feature selection, the final results of the study concluded that the Firefly search algorithm is effective for feature selection problems with the results of classification accuracy SVM-with FS has better accuracy by 4.53% compared to SVM-without FS, by 5.4% compared to KNN without FS, by 11% compared to NB-without FS. In a previous study conducted by Zhou et al. proposes several methods such as Deep Belief Networks

(DBN), Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR) and Support Vector Machine (SVM) in predicting software defects. The data used are NASA, PROMISE, AEEEM and ReLink datasets. Based on the comparison results, for the NASA dataset it can be seen that DPDF has the best performance, AUC increases and the highest value is 92%. For the PROMISE and AEEEM datasets, the DPDF results are also better than the others, with the highest scores of 89% and 86%. And across multiple datasets ReLink, DPDF has not outperformed RF and DBN, the highest score is 75%. Feature selection plays a crucial role in a plethora of applications owing to its indispensability in ensuring generalization, performance, computational efficiency, and feature interpretability[8]. So in this study research will be carried out on the application of feature selection in predicting software defects using the Firefly algorithm for tree-based classification, namely Decision Tree, Random Forest classification, and Deep Forest with the aim of improving the resulting performance.

II. METHOD

This research method describes the dataset used, Decision Tree algorithm, Random Forest, Deep Forest, Firefly algorithm, test validation using cross validation, and performance measurement using the evaluation method using AUC. The following is the research procedure that will be carried out. Figure 1 show the flow of this research.

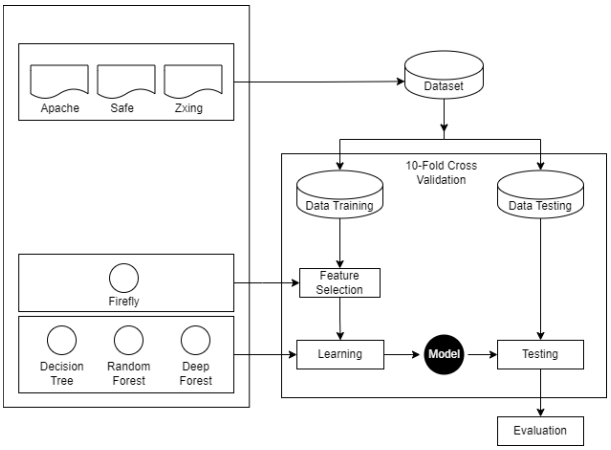


FIGURE 1. Research flow with feature selection

In this study, a flowchart is presented in Figure 1. The initial step involves collecting the ReLink dataset, followed by sharing the data using cross validation. The validation technique adopted in this study is 10-fold validation. To achieve this, each ReLink dataset is partitioned into 10 sections, with 9 sections designated as training data and the remaining section used as test data. Subsequently, feature selection is executed via the Firefly algorithm before classification. The classification phase involves three scenarios, which are Decision Tree, Random Forest, and Deep Forest. The study's evaluation employs the average AUC value. The experimentation was conducted using Python Google Collaboratory.

A. DATA COLLECTION

The dataset used in this study is a software metrics dataset called ReLink, which consists of Apache, Safe, and Zxing data. This dataset can be downloaded at the following link https://github.com/bharlow058/AEEEM-and-otherSDP-datasets/tree/master/dataset/Relink.

TABLE 1 shows the frequency of data that varies in each ReLink dataset, namely Apache with 194 data, Safe with 56 data and Zxing with 399 data. Then explains the ReLink dataset software metrics which are grouped into 2 software metric categories (groups), namely Complexity Metric (CPM) and Count Metric (CTM). The ReLink dataset has the same number of software metrics[7].

TABLE 1
Relink dataset

Matrix Category	Attribute Name	Dataset Original		
		Apache	Safe	Zxing
Complexity Metric(CPM)	AvgCyclomatic	√	√	√
	AvgCyclomaticModified	√	√	√
	AvgCyclomaticStrict	√	√	√
	AvgEssential	√	√	√
	MaxCyclomatic	√	√	√
	MaxCyclomaticModified	√	√	√
	MaxCyclomaticStrict	√	√	√
	RatioCommentToCode	√	√	√
	SumCyclomatic	√	√	√
	SumCyclomaticModified	√	√	√
	SumCyclomaticStrict	√	√	√
	SumEssential	√	√	√
Count Metric(CTM)	AvgLine	√	√	√
	AvgLineBlank	√	√	√
	AvgLineCode	√	√	√
	AvgLineComment	√	√	√
	CountLine	√	√	√
	CountLineBlank	√	√	√
	CountLineCode	√	√	√
	CountLineCodeDecl	√	√	√
	CountLineCodeExe	√	√	√
	CountLineComment	√	√	√
	CountSemicolon	√	√	√
	CountStmt	√	√	√
	CountStmtDecl	√	√	√
	CountStmtExe	√	√	√
	Number of Modules	194	56	399
	Number of Attributes (Feature)	26	26	26
	Number of Classes (Buggy)	98	22	ll8
	Number of Classes (Clean)	96	34	281
	Percentage (Buggy)	50.52%	39.29%	29.57%

B. DATA SHARING

1. CROSS VALIDATION

The reduction of bias in the case of random sampling of datasets is accomplished through the implementation of cross validation[9]. Cross Validation divides the original data into training data and testing data[4]. It consists of randomly dividing the data set into K parts[10]. One part is used to validate the model and the rest to train the classifier. This process is repeated K times, selecting different validation subsets. Cross Validation divides raw data into training data and testing data randomly. Weaknesses that K-Fold Cross Validation has when using unbalanced data where there is a possibility of causing some data to be lost and only testing a few instances so that there are still many untested[11].

C. FEATURE SELECTION

1. FIREFLY ALGORITHM BASED FEATURE SELECTION

Feature selection constitutes a combinational optimization problem[12]. The Firefly algorithm (FA) is a novel population-based meta-heuristic algorithm that exhibits exceptional performance on a multitude of optimization problems[13]. The Firefly Algorithm is algorithm that draws inspiration from the light flashing behavior of the original Firefly[14]. It should be noted that every Firefly has its unique position that is determined by the number that is generated for each of them[15]. Firefly Algorithm for discriminatory features selection of classification and regression models to support the decision-making process using database-based learning methods[16]. It can be posited that the algorithm in question has achieved a remarkable level of success, despite its relatively low cost[17]. This algorithm is inspired by the blinking behavior of a Firefly, a randomly generated solution will be treated as a Firefly, and the brightness assigned depends on its performance in the objective function[14]. The brightness of a Firefly is determined by evaluating the fitness function. For the problem of maximizing brightness, it can be compared with the value of the objective function (fitness function)[18]. The Firefly algorithm exhibits superior capacity to evade trapping in local optima, alongside a marked enhancement in both the speed of convergence and precision of solutions[19].

The attractiveness of the Firefly is determined by its brightness, which is contingent on the light intensity. The calculation of attractiveness for each Firefly is accomplished through the utilization of Equation (1)[14].

$$\beta(r) = \beta_0 e^{-\gamma r2} \quad (1)$$

where variable β0 is utilized to signify the level of appeal at the point where distance (r)=0, and in certain instances, it is regarded as equivalent to the value of one for mathematical computations. Meanwhile, the symbol γ is representative of the degree of light absorption. It should be noted that r denotes the distance between two fireflies, i and j, who are in constant motion from one position to another. It is a well-established fact that the degree of attractiveness between these fireflies is closely linked with the distance that separates them. Therefore, the distance between two fireflies, i and j, is determined using the Euclidean distance law[14]. calculated by the equation(2).

$$rij = ||xi - xj|| = \sqrt{\sum_k^d = 1(Xi,k - Xj,k)^2} \quad (2)$$

where d denotes the dimensions of the given problem, xi,k corresponds to the k-th component of the Firefly position i. After calculating the distance between two fireflies, if Firefly i exhibits a lower luminosity compared to Firefly j, then the resulting attraction between the two occurs when Firefly i moves towards Firefly j. the movement in question is governed by Equation (3)[14], which is stated as follows:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-(\gamma r_{ij}^2)} * \left(x_j^t - x_i^t\right) + \alpha * (rand - 1/2) \quad (3)$$

where t denotes the number of iterations, the coefficient α denotes a stochastic variable governing the magnitude of the random walk, and rand signifies a random number generator that falls within the interval [0,1]. The Firefly with lower luminosity translocates towards the brighter Firefly after considering three factors[14]. The first factor corresponds to the current position of the less luminous Firefly. The second factor denotes the movement towards the brighter Firefly, which is guided by the attraction coefficient β. Finally, the last factor corresponds to a type of random walk that is computed by a random generator multiplied by α.

D. CLASSIFICATION

1. DECISION TREE CLASSIFICATION

A Decision Tree(DT) is a classification technique utilized in data mining that constructs a model in a top-down tree-like fashion, predicated on the attributes intrinsic to a designated data set[20]. The Decision Tree classification method is capable of resolving both binary and multi-class classification problems in data mining classification[21]. As with an ordinary tree, the Decision Tree comprises a root, branches, and leaves, adhering to the same structure[22]. The essence of DT lies in its hierarchal and predictive modeling strategy, wherein the item's observation serves as branches to determine the item's target value in the leaf[23].

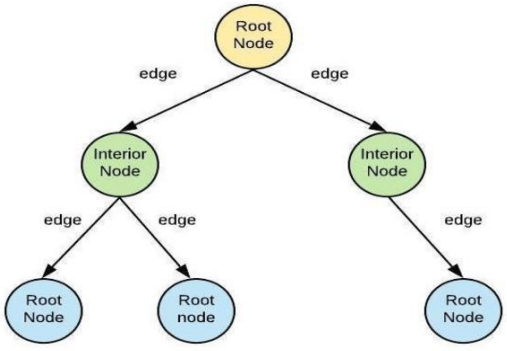


FIGURE 2. Struktur Decision Tree[24]

This implies that it is a coordinated tree through a node called the "root," with no imminent edges, while various

other nodes have only one imminent edge. An inner or exam node is referred to as a center with complex edges. Each additional node is titled as either greeneries or incurable or excellent nodes. The leaf node is linked to the name of the class. The Decision Tree is an integral constituent of the planning set[24]. A Decision Tree of this nature is depicted in FIGURE 2.

2. RANDOM FOREST CLASSIFICATION

Random Forest(RF) algorithm is a supervised classification algorithm, as indicated by its name, which involves the creation of a forest through a random process. The number of trees within the forest directly affects the accuracy of the outcomes, with larger numbers of trees resulting in greater precision[25]. Random Forest classification is done by obtaining the majority class votes from the individual vote class trees[26]. One important benefit associated with RF relates to the fact that there is no need to prune individual trees, given the presence of multiple trees. However, the disadvantage is that due to the large number of trees, the ability to visualize them effectively is impaired[27]. This method is underpinned by two primary principles: row sampling and voting classifier. The provided records are resampled and then forwarded to the next base learner models for training. Aggregating is the voting classifier concept, where the output for test data is chosen for the class with the highest vote from the base learner models[28]. A generalized model for the Random Forest is depicted in FIGURE 3.

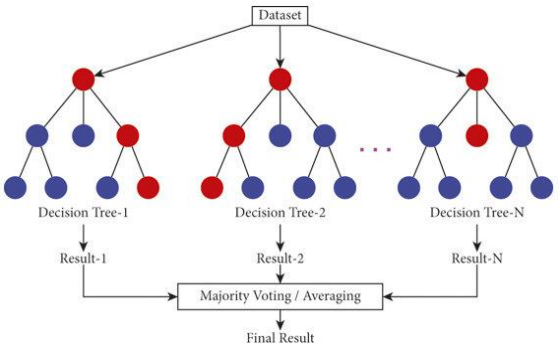


FIGURE 3. Random Forest structure[28]

3. DEEP FOREST CLASSIFICATION

Deep Forest is a new tree based classification algorithm which is an improvement over Random Forest algorithm. Deep Forest is referred to as an alternative Deep Neural Network (DNN), Deep Forest has parts or components, namely a layer-by-layer structure called a cascade forest[29]. A cascading forest is a distribution of classes generated by each tree for each instance[30]. The image presented below illustrates the layered nature of the algorithm, where each layer is stacked one on top of the other. The initial layer obtains input from the original dataset's attributes or features, which are then handled by the Random Forest in the next layer (FIGURE 4). The layer will stop if the process generated Random Forest does not increase or if the output at the given layer decreases. The Deep Forest algorithm will average the results from layer to layer to the final layer of each layer level. The downside is that Deep Forests take longer to process than Random Forests[7].

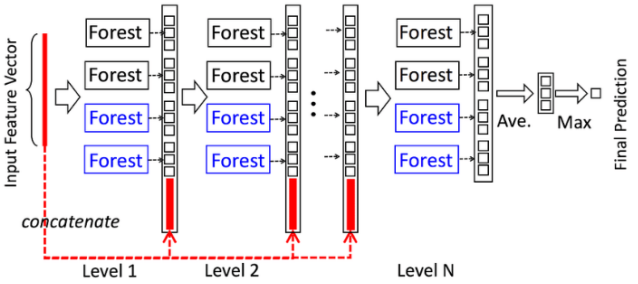


FIGURE 4. The architecture of the cascade forest[29]

E. EVALUATION OF RESULT

The features of this study were taken from 3 ReLink datasets obtained from the github repository, each of which has 26 features. Feature selection is an important step in data analysis, because the right features will improve the classification performance of the model. In this study, feature selection was performed using the Firefly Algorithm to improve feature selection efficiency and improve the accuracy of the tree-based classification model. Firefly is used to find the best feature combination that gives higher AUC performance than the classification model. In this study, 10 trials were carried out to find out the average AUC value obtained. After implementing Firefly, the final results show a comparison of the AUC between models that use a combination of classification and selection of Firefly features and models that use classification and hyperparameter tuning, so that it can be seen whether the implementation of the Firefly feature provides an increase in AUC performance in classification on prediction of software defects. Evaluation of the classification performance of the Decision Tree, Random Forest and Deep Forest models for each ReLink dataset uses the AUC (Area under the ROC Curve) value.

The AUC represents the area under the ROC curve and has been recommended for improving cross-study comparability. Its potential for significantly enhancing convergence across empirical experiments in software defect prediction lies in its ability to disentangle predictive performance from operating conditions, thereby serving as a general measure of predictiveness[31].

III. RESULT

The TABLE 2 shows the performance of a tree-based classification algorithm with the Firefly search feature on the Apache dataset.

TABLE 2
Table of AUC results with Firefly feature selection on the Apache, Safe and Zxing datasets

Datasets	Average Features	Avarage AUC Value		
		Decision Tree (DT)	Random Forest (RF)	Deep Forest (DF)
Apache	11.5	0.658	0.773	0.756
Safe	12.3	0.7323	0.8388	0.8382

Datasets	Average Features	Avarage AUC Value		
		Decision Tree (DT)	Random Forest (RF)	Deep Forest (DF)
Zxing	12.1	0.5904	0.7138	0.679

The TABLE 3 shows the number of times a feature appears in 10 trials using Firefly feature selection on the Apache, Safe and Zxing datasets.

TABLE 3
The number of features that appear in the dataset Apache, Safe and Zxing

No.	Features	Apache	Safe	Zxing	Average
1	AvgCyclomatic	3	4	3	3.3
2	AvgCyclomaticModified	2	3	2	2.3
3	AvgCyclomaticStrict	1	4	5	3.3
4	AvgEssential	5	4	6	5
5	MaxCyclomatic	2	3	3	2.6
6	MaxCyclomaticModified	3	6	4	4.3
7	MaxCyclomaticStrict	6	4	3	4.3
8	RatioCommentToCode	7	4	6	5.6
9	SumCyclomatic	7	6	5	6
10	SumCyclomaticModified	5	6	6	5.6
11	SumCyclomaticStrict	5	5	5	5
12	SumEssential	5	3	3	3.6
13	AvgLine	6	4	4	4.6
14	AvgLineBlank	3	7	3	4.3
15	AvgLineCode	2	5	8	5
16	AvgLineComment	4	5	5	4.6
17	CountLine	7	5	7	6.3
18	CountLineBlank	4	4	7	5
19	CountLineCode	5	7	4	5.3
20	CountLineCodeDecl	5	4	4	4.3
21	CountLineCodeExe	8	4	5	5.6
22	CountLineComment	4	3	2	3
23	CountSemicolon	1	7	4	4
24	CountStmt	4	7	3	4.6
25	CountStmtDecl	5	4	7	5.3
26	CountStmtExe	6	4	7	5.6

IV. DISCUSSION

In this study, a total of 10 trials were carried out to obtain the average value. The results of the software defect prediction assessment of the three ReLink datasets on the area under the curve (AUC) values obtained from the ten experiments conducted are presented in Tables 2, 4, and 6. Due to the random selection of the Firefly feature according to the best intensity, the selected features change with each trial, resulting in varying AUC values, some higher and some lower. It should be noted that the AUC values obtained from each experiment are different, with the optimal average number of features used being 12 features.

TABLE 3, TABLE 4, and TABLE 5 describe the frequency of characteristic selection via the utilization of the Firefly algorithm on the 26 features of the ReLink dataset, organized according to their respective degree of implementation. Among the plethora of garnered findings, the Matrix Category Count Metric (CTM) emerges as the most frequently employed feature. TABLE 4 and FIGURE 5 show the average AUC values achieved across all ReLink datasets and the most frequently used feature sets.

The feature selection carried out by the fireflies on all tree-based classification algorithms has proven successful in elevating software defect prediction performance compared to prior studies that employed hyperparameter tuning. This is evidenced by the superior average performance of each proposed method, as shown in Table 10, relative to previous methodologies.

TABLE 4
Comparison of AUC results with previous studies

Datasets	Previous research methods (AUC)			The proposed research method(AUC)		
	DT[[7]	RF[7]	DF[7]	DT	RF	DF
Apache	0.76	0.76	0.75	0.66	0.77	0.76
Saf	0.63	0.73	0.73	0.73	0.84	0.84
Zxing	0.64	0.67	0.70	0.59	0.71	0.68
Average	**0.66**	**0.72**	**0.73**	**0.66**	**0.77**	**0.76**

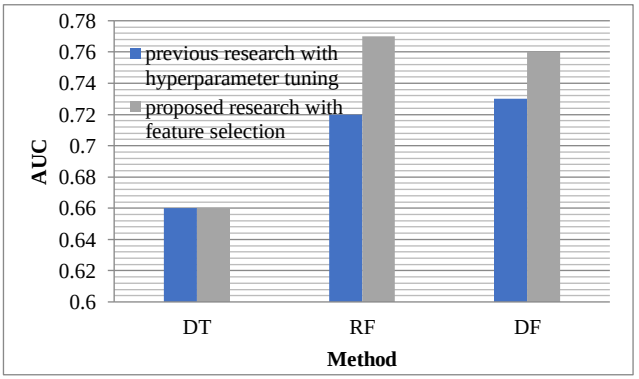


FIGURE 5. Graph of AUC performance comparison with previous studies

Based on the data presented in TABLE 4 and FIGURE 6, a comparison is made between the Decision Tree algorithm that utilizes the proposed Firefly feature selection method and the Decision Tree algorithm with hyperparameter settings, as previously studied. Interestingly, both methods produce an equivalent AUC value of 0.66. In contrast, the proposed Random Forest algorithm with Firefly feature selection shows an improvement of 5% compared to the Random Forest algorithm with hyperparameter tuning, as examined in previous studies. Likewise, the Deep Forest algorithm using the proposed Firefly feature selection method produces a 3% increase when compared to the Deep Forest algorithm with hyperparameter settings in previous studies.

In previous research. the Decision Tree parameter was set to the default value or without selecting the Firefly

feature. resulting in an average AUC value of 0.66. The Random Forest parameter is set to the default value or without the Firefly feature selected. resulting in an average AUC value of 0.72. And the Deep Forest parameter is set to the default value or without the Firefly selection feature resulting in an average AUC value of 0.73. In this study. the Decision Tree parameters with Firefly feature selection produced an average AUC value of 0.66. Setting the Random Forest parameter by selecting the Firefly feature produces an average AUC value of 0.77. And the Deep Forest parameter is set by selecting the Firefly feature so that it produces an average AUC value of 0.76.

Table 5
Comparison of AUC results with other research methods

Datasets	Previous research methods (AUC)			The proposed research method (AUC)		
	NB [29]	LR [29]	SVM[29]	DT	RF	DF
Apache	0.74	0.70	0.76	0.66	0.77	0.76
Safe	0.69	0.67	0.69	0.73	0.84	0.84
Zxing	0.61	0.57	0.66	0.59	0.71	0.68
Average	**0.68**	**0.64**	**0.70**	**0.66**	**0.77**	**0.76**

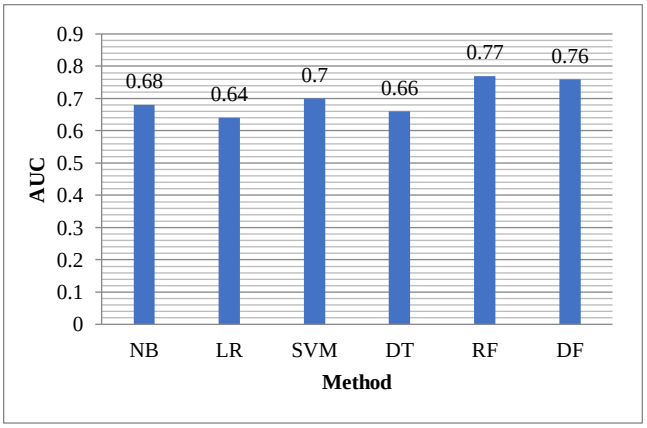


FIGURE 6. Comparison of AUC results with other research methods

TABLE 5 and FIGURE 6 present a comprehensive analysis of the results compared to previous investigations using various methodologies. It is clear that the results of this study outperform those of its predecessors. In particular, the average AUC value of the Random Forest and Deep Forest classification technique using the Firefly feature selection method outperforms the average AUC value of other methodologies.

V. CONCLUSION

This study aims to predict software defects in the ReLink dataset through the application of Decision Tree, Random Forest, and Deep Forest tree-based classification with Firefly feature selection. The performance of these models varies, as evidenced by the comparison results in experimental trials. Specifically, Firefly's feature selection was found to improve AUC performance when compared to previous studies using hyperparameter tuning for tree-based classification. In addition, Firefly feature selection combined with tree-based classification outperformed previous studies using the Naïve Bayes (NB) method, as well as Logistic Regression (LR) and Support Vector Machine (SVM). Overall, these findings highlight the potential benefits of using Firefly feature selection with tree-based classification to perform well in predicting software crashes.

The findings of the research indicate that the application of the Firefly feature selection in conjunction with Random Forest classification yields superior performance in comparison to feature selection utilizing other classifications based on trees. This is evidenced by an average AUC value of 0.77, an average feature usage of 12 out of 26 features, and the most frequently occurring feature belonging to the Count Metric category. Thus, the results suggest that the features categorized under CountMetric are the most effective. In future studies, the tree-based algorithm will be tested with the firefly selection feature on other datasets that have a higher score ratio. The goal is to find out better algorithm performance in predicting software defects. Another further research is experimenting with firefly features with other classification algorithms in predicting software defects. The aim is to find out the search for firefly features with a classification algorithm that is expected to get a better performance value.

ACKNOWLEDGMENT

We would like to thank all the lecturers and staff of the Computer Science study program, Faculty of Mathematics and Natural Sciences, University of Lambung Mangkurat who have provided the necessary resources and helped complete this research.

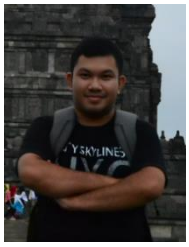
REFERENCES

[1] H. K. Dam, J. Grundy, T. Kim, and C. Kim, "A deep tree-based model for software defect prediction".
[2] F. Yucalar, A. Ozcift, E. Borandag, and D. Kilinc, "Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability," *Eng. Sci. Technol. an Int. J.*, vol. 23, no. 4, pp. 938–950, 2020, doi: 10.1016/j.jestch.2019.10.005.
[3] M. S. Rawat and S. K. Dubey, "Software defect prediction models for quality improvement: A literature study," *Int. J. Comput. Sci. Issues*, vol. 9, no. 5 5–2, pp. 288–296, 2018.
[4] J. Ren, K. Qin, Y. Ma, and G. Luo, "Survey on Software Defect Prediction Using Machine Learning Techniques," *J. Appl. Math.*, vol. 3, no. 12, pp. 2319–7064, 2018, doi: 10.1155/2014/785435.
[5] J. Grundy, T. Kim, and C. Kim, "Lessons learned from using a deep tree-based model for software defect prediction in practice," no. May, pp. 26–27, 2019.
[6] Z. Li, X. Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," *IET Softw.*, vol. 12, no. 3, pp. 161–175, 2018, doi: 10.1049/iet-sen.2017.0148.
[7] E. Andini, M. R. Faisal, and R. Herteno, "Software Defect Prediction Performance Improvement With Hyperparameter Tuning In Deep Forest Classification Algorithm," vol. 5, no. 2, pp. 119–127, 2022.
[8] N. Gayatri, S. Nickolas, and A. V Reddy, "Feature Selection Using Decision Tree Induction in Class level Metrics Dataset for Software Defect Predictions," *World Congr. Eng. Comput. Sci. Vols 1 2*, vol. I, pp. 124–129, 2020.
[9] S. Ghosh, A. Rana, and V. Kansal, "A Nonlinear Manifold Detection based Model for Software Defect Prediction," *Procedia Comput.*

Sci., vol. 132, pp. 581–594, 2018, doi: 10.1016/j.procs.2018.05.012.
[10] H. Wei, C. Hu, S. Chen, Y. Xue, and Q. Zhang, "Establishing a software defect prediction model via effective dimension reduction," *Inf. Sci. (Ny).*, vol. 477, pp. 399–409, 2019, doi: 10.1016/j.ins.2018.10.056.
[11] B. Kovalerchuk, "Enhancement of Cross Validation Using Hybrid Visual and Analytical Means with Shannon Function," *Stud. Comput. Intell.*, vol. 835, pp. 517–543, 2020, doi: 10.1007/978-3-030-31041-7_29.
[12] H. Xu, S. Yu, J. Chen, and X. Zuo, "An Improved Firefly Algorithm for Feature Selection in Classification," *Wirel. Pers. Commun.*, 2018, doi: 10.1007/s11277-018-5309-1.
[13] J. Wang, "A novel firefly algorithm for portfolio optimization problem," *IAENG Int. J. Appl. Math.*, vol. 49, no. 1, 2019.
[14] E. M. Mashhour, E. M. F. El Houby, K. T. Wassif, and A. I. Salah, "Feature selection approach based on firefly algorithm and chi-square," *Int. J. Electr. Comput. Eng.*, vol. 8, no. 4, pp. 2338–2350, 2018, doi: 10.11591/ijece.v8i4.pp2338-2350.
[15] R. F. Najeeb and B. N. Dhannoon, "A feature selection approach using binary Firefly Algorithm for network intrusion detection system A FEATURE SELECTION APPROACH USING BINARY FIREFLY ALGORITHM FOR NETWORK INTRUSION DETECTION SYSTEM," no. March, 2018.
[16] L. Zhang, K. Mistry, C. P. Lim, and S. C. Neoh, "Feature selection using firefly optimization for classification and regression models," *Decis. Support Syst.*, vol. 106, pp. 64–85, 2018, doi: 10.1016/j.dss.2017.12.001.
[17] A. H. Damia, "Automated Test Data Generation Using a Combination of Firefly Algorithm and Asexual Reproduction Optimization Algorithm," 2020.
[18] S. Larabi Marie-Sainte and N. Alalyani, "Firefly Algorithm based Feature Selection for Arabic Text Classification," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 32, no. 3, pp. 320–328, 2020, doi: 10.1016/j.jksuci.2018.06.004.
[19] T. Fan, J. Wang, M. Feng, X. Zhang, J. Wang, and R. Wu, "Application of multi-objective firefly algorithm based on archive learning in robot path planning," *Int. J. Intell. Inf. Database Syst.*, vol. 12, no. 3, pp. 199–211, 2019, doi: 10.1504/IJIIDS.2019.102939.
[20] A. K. Hamoud, A. S. Hashim, and W. A. Awadh, "Predicting Student Performance in Higher Education Institutions Using Decision Tree Analysis," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 5, no. 2, p. 26, 2018, doi: 10.9781/ijimai.2018.02.004.
[21] M. A. Febriantono, S. H. Pramono, Rahmadwati, and G. Naghdy, "Classification of multiclass imbalanced data using cost-sensitive decision tree c5.0," *IAES Int. J. Artif. Intell.*, vol. 9, no. 1, pp. 65–72, 2020, doi: 10.11591/ijai.v9.i1.pp65-72.
[22] H. H. Patel and P. Prajapati, "Study and Analysis of Decision Tree Based Classification Algorithms," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 10, pp. 74–78, 2018, doi: 10.26438/ijcse/v6i10.7478.
[23] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, "Software Bug Prediction using machine learning approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 78–83, 2018, doi: 10.14569/IJACSA.2018.090212.
[24] F. M. J. M. Shamrat *et al.*, "CART Decision Tree Algorithms".
[25] R. Rama Devi and M. Abualkibash, "Intrusion Detection System Classification Using Different Machine Learning Algorithms on KDD-99 and NSL-KDD Datasets - A Review Paper," *Int. J. Comput. Sci. Inf. Technol.*, vol. 11, no. 03, pp. 65–80, 2019, doi: 10.5121/ijcsit.2019.11306.
[26] H. Tyralis, G. Papacharalampous, and A. Langousis, "A brief review of random forests for water scientists and practitioners and their recent history in water resources," *Water (Switzerland)*, vol. 11, no. 5, 2019, doi: 10.3390/w11050910.
[27] A. E. Maxwell, T. A. Warner, and F. Fang, "Implementation of machine-learning classification in remote sensing: An applied review," *Int. J. Remote Sens.*, vol. 39, no. 9, pp. 2784–2817, 2018, doi: 10.1080/01431161.2018.1433343.
[28] H. B. Kibria and A. Matin, "The severity prediction of the binary and multi-class cardiovascular disease − A machine learning-based fusion approach," *Comput. Biol. Chem.*, vol. 98, no. March, 2022, doi: 10.1016/j.compbiolchem.2022.107672.
[29] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, "Improving defect prediction with deep forest," *Inf. Softw. Technol.*, vol. 114, no. July 2018, pp. 204–216, 2019, doi: 10.1016/j.infsof.2019.07.003.
[30] L. V. Utkin, M. S. Kovalev, and A. A. Meldo, "A deep forest classifier with weights of class probability distribution subsets," *Knowledge-Based Syst.*, vol. 173, pp. 15–27, 2019, doi: 10.1016/j.knosys.2019.02.022.
[31] R. S. Wahono and N. Suryana, "Combining particle swarm optimization based feature selection and bagging technique for software defect prediction," *Int. J. Softw. Eng. its Appl.*, vol. 7, no. 5, pp. 153–166, 2013, doi: 10.14257/ijseia.2013.7.5.16.

BIOGRAPHY

Vina Maulida originated in Amuntai, Hulu Sungai Utara, South Kalimantan. Since 2018, she has pursued her academic endeavors as a student of Computer Science Department at Universitas Lambung Mangkurt. Her current area of research lies within the realm of software engineering. Additionally, his final assignment involves research centered on predicting defects in software. The goal of this research effort is to predict defects in software.



Rudy Herteno, was born in Banjarmasin, South Kalimantan. After graduating from high school, he pursued his undergraduate studies in the Computer Science Department at Lambung Mangkurat University and graduated in 2011. After completing his undergraduate program, he worked as a software developer to gather experience for several years. He developed a lot of software, especially for local governments. In 2017, He completed his master's degree in Informatics from STMIK Amikom University. Currently, he is a lecturer in the Faculty of Mathematics and Natural Science at Lambung Mangkurat University. His research interests include software engineering, software defect prediction, and deep learning.

Mohammad Reza Faisal was born in Banjarmasin. Following his graduation from high school, he pursued his undergraduate studies in the Informatics department at Pasundan University in 1995, and later majored in Physics at Bandung Institute of Technology in 1997. After completing his bachelor's program, he gained experience as a training trainer in the field of information technology and software development. Since 2008, he has been a lecturer in computer science at Universitas Lambung Mangkurat, while also pursuing his master's program in Informatics at Bandung Institute of Technology in 2010. In 2015, he furthered his education by pursuing a doctoral degree in Bioinformatics at Kanazawa University, Japan. To this day, he continues his work as a lecturer in Computer Science at Universitas Lambung Mangakurat. His research interests encompass Data Science, Software Engineering, and Bioinformatic.

Dwi Kartini received her bachelor's and master's degrees in computer science from the Faculty of Computer Science, Putra Indonesia University "YPTK" Padang, Indonesia. Her research interests include the applications of Artificial Intelligence and Data Mining. She is an assistant professor in the Department of Computer Science, Faculty of Mathematics and Natural Sciences, Lambung Mangkurat University in Banjarbaru, Indonesia.

Friska Abadi finished his bachelor's degree in Computer Science from Universitas Lambung Mangakurat in 2011. Subsequently, in 2016, he obtained her master's degree from the Department of Informatics at STIMIK Amikom, Yogyakarta. Following that, he joined Universitas Lambung Mangakurat as a lecturer in Computer Science. Currently, he holds the position of head of the software engineering laboratory. Him current area of research revolves around software engineering.