SOFTWARE FAULT PREDICTION USING THE HYBRID FEATURE SELECTION: A STUDY ON THE BUGHUNTER DATASET

Ha Thi Minh Phuong[1], Nguyen Do Anh Nhu[2], Hoang Thi Thanh Ha[2]*

[1]The University of Danang - Vietnam-Korea University of Information and Communication Technology, Vietnam
[2]The University of Danang - University of Economics, Vietnam

*Corresponding author: ha.htt@due.edu.vn

(Received: March 20, 2026; Revised: May 24, 2026; Accepted: May 29, 2026)

DOI: 10.31130/ud-jst.2026.24(6A).165E

Abstract - Software defect prediction is essential for improving software quality and reducing testing costs. However, defect datasets typically contain high-dimensional, imbalanced, and noisy features that negatively affect machine learning performance. This study proposes a hybrid feature selection approach that integrates four statistical filters - Correlation-based Feature Selection, Fisher Score, Laplacian Score, and Gain Ratio - with the Whale Optimization Algorithm (WOA). Experiments were conducted on seven BugHunter datasets containing 96 original features, using Random Forest as the predictive model. The hybrid method reduces the feature set to 19–28 features and achieves the best performance among all tested configurations, with an average Accuracy of 0.720, F1-score of 0.692, and AUC of 0.768. The results demonstrate that combining multi-filter feature selection with meta-heuristic optimization effectively removes irrelevant features, enhances model stability, and improves predictive capability. This hybrid approach shows strong potential for practical applications in software engineering.

Key words - Software defect prediction; feature selection; meta-heuristic optimization; Whale Optimization Algorithm; BugHunter

1. Introduction

As the scale and complexity of software systems continue to grow, the early prediction of fault-prone components (Software Fault Prediction – SFP) has become an essential task in software quality assurance. SFP enables the identification of modules likely to contain defects before the testing phase, thereby helping to concentrate resources (personnel, time, cost) on high-risk areas, enhancing the system's reliability and safety. However, the effectiveness of fault prediction models heavily relies on the quality of the input data. Software fault datasets are often high-dimensional, containing numerous collected metrics, many of which are redundant or not directly related to faults. The presence of these unhelpful features can lead to the "curse of dimensionality," reducing model accuracy and increasing training time. Therefore, feature selection (FS) is a crucial step aimed at reducing data dimensionality, eliminating noise, and focusing on attributes that are truly significant for fault prediction.

Feature selection helps extract an optimal feature subset by retaining attributes highly relevant to the fault label and discarding redundant ones. Consequently, FS can significantly improve the accuracy and performance of machine learning models in SFP. In other words, filtering out the core features from a myriad of potential metrics reduces data noise, prevents overfitting, and speeds up model training. The problem lies in how to determine the optimal feature set when the dataset contains dozens to hundreds of different attributes, many of which may not contribute to fault prediction or might even introduce noise to the model. BugHunter software defect dataset contains many types of source code metrics and various fault information [1]. Incorrectly selecting or overlooking important features in such datasets can negatively impact SFP accuracy.

In this paper, we propose a hybrid multi-layer feature selection method to address the aforementioned problem. Specifically, our approach combines filter techniques and a wrapper method in a complementary manner:

(1) Filter Stage: Initially, four common filter algorithms, including Correlation-based Feature Selection (CFS), Fisher Score, Lap Score, and Gain Ratio, are applied to quickly assess the importance of each attribute. These filters represent diverse criteria: CFS examines the correlation between features and the fault label, Fisher assesses fault discriminability based on statistics, Lap Score leverages the intrinsic structure of the data, and Gain Ratio measures the acquired information adjusted for bias.

(2) Wrapper Stage: Subsequently, a candidate feature set (comprising features highly ranked by the filters) is further optimized through a wrapper algorithm based on the Whale Optimization Algorithm. WOA is a swarm intelligence optimization algorithm inspired by the hunting strategy of humpback whales. This algorithm simulates the whales' encircling and spiral attacking behavior towards prey, allowing for an effective balance between the exploration and exploitation phases of the search space. Thanks to WOA's robust global search capability, our wrapper method can find an optimal feature combination that is often difficult to achieve with greedy or exhaustive search methods.

The goal of this research is to find a compact feature subset that carries the highest fault prediction information on the BugHunter dataset, thereby improving the accuracy of the software fault prediction model. By eliminating redundant metrics and retaining only the useful features, the resulting SFP model is both more accurate and has a lower computational cost compared to a model using the entire initial feature set.

The main contributions of this paper include:

(1) Proposing a novel framework that combines multiple feature filters and the WOA wrapper algorithm for the software fault prediction problem.

(2) A wrapper-based feature optimization strategy using the Whale Optimization Algorithm is developed on top of the filtered feature space, enabling effective

exploration of feature interactions and identification of a compact and informative feature subset.

(3) Empirical validation on a real-world fault dataset (BugHunter) to prove that the proposed method significantly improves prediction accuracy compared to traditional feature selection approaches.

2. Related Work

Many studies have focused on applying feature selection techniques to improve the effectiveness of SFP models. Broadly, FS methods are categorized into three main groups: filter, wrapper, and embedded. The filter method evaluates feature usefulness based on statistical or information-theoretic criteria without relying on a machine learning model; the wrapper method directly uses a learning algorithm to assess the quality of the feature set; and the embedded method integrates feature selection right into the model training process.

Applying feature selection has been proven to be clearly effective for the SFP problem: it helps reduce model complexity and increase prediction accuracy by eliminating irrelevant attributes. For instance, the study by Wang et al. investigated software defect prediction under class imbalance conditions and highlighted that noisy and redundant features exacerbate the learning difficulty of defect prediction models [2]. Their study emphasized the necessity of combining appropriate data preprocessing techniques, including feature selection, to mitigate these issues. Similarly, Catal et al. demonstrated that incorporating feature selection into practical fault prediction tools based on traditional classifiers, such as Naïve Bayes, leads to improved prediction accuracy on real-world software projects [3]. These early studies established feature selection as an essential component in SFP pipelines.

Subsequent research has focused on systematically evaluating and comparing different feature selection approaches. Balogun *et al.* performed a comprehensive performance analysis of various feature selection methods from a search-based perspective and reported that search and wrapper-based approaches generally outperform simple filter methods in terms of prediction accuracy and robustness across datasets [4]. Their findings suggest that the choice of feature selection technique has a more pronounced impact on model performance than the choice of classifier in many SFP scenarios.

Despite the advantages of wrapper-based approaches, filter-based feature selection methods have remained widely adopted due to their computational efficiency and scalability. Rathi *et al.* empirically evaluated the combined impact of data sampling techniques and feature selection methods on software fault prediction and reported that filter-based FS can significantly improve prediction performance when appropriately integrated with sampling strategies [5]. Their study highlights the practical appeal of filter methods in handling high-dimensional software metrics, particularly when computational resources are limited.

Beyond simple filter and wrapper strategies, ensemble and optimization-driven feature selection approaches have gained increasing attention. Balogun *et al.* proposed an adaptive ensemble rank-based multi-filter feature selection method (AREMFFS), which combines multiple filter rankings to improve feature stability and selection reliability [6]. Their results demonstrated that aggregating the outputs of multiple filters leads to more stable feature subsets and improved prediction performance compared to single-filter approaches. Ha *et al.* performed an evaluation on the Filter-based FS methods using five learner over NASA dataset. The results showed that Chi-Square and Information Gain achieved the best performance [7].

In parallel, swarm intelligence and meta-heuristic optimization algorithms have been increasingly applied to feature selection problems. Almomani investigated the use of PSO, GWO, FFA, and GA for feature selection and showed that these optimization algorithms are effective in identifying compact and informative feature subsets [7]. Although the study focused on intrusion detection, its conclusions are directly relevant to SFP due to similar challenges in high-dimensional feature spaces and feature redundancy.

Recent studies have further explored hybrid and meta-heuristic-based feature selection frameworks specifically for software fault prediction. Malhotra *et al.* introduced a hybrid multi-filter wrapper feature selection framework combined with a deep neural network and attention mechanism, achieving significant improvements in defect prediction performance [8]. Their work demonstrates the effectiveness of integrating multiple filter methods with wrapper optimization; however, it relies on complex deep learning architectures that increase computational cost.

Akbar *et al.* proposed integrating hybrid Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO) to enhance feature selection for software defect prediction models based on gradient boosting algorithms [9]. Their results indicate that hybrid swarm-based optimization techniques can further improve feature selection quality and predictive performance. Similarly, Al-Wajih *et al.* proposed a hybrid binary Grey Wolf Optimizer combined with Harris Hawks Optimization (GWO–HHO) for feature selection and demonstrated superior performance compared to standalone optimization algorithms [10]. Ha *et al.* applied a wrapper-based FS, namely Zebra Optimization Algorithm (ZOA) to select the optiomal features for SFP models. Their experimental results showed that the effectiveness of ZOA in reduce redundant features and improve the SFP model's performance [11]. These studies reinforce the potential of combining multiple meta-heuristic strategies to enhance feature selection effectiveness.

Despite these advances, several limitations can be observed in existing studies. Most prior works focus on either single filter-based techniques, ensemble filter methods, or hybrid optimization approaches without systematically integrating multiple complementary filter criteria prior to wrapper optimization. Furthermore, many recent hybrid approaches rely on deep learning or complex ensemble classifiers, which may increase computational cost and limit applicability in practical software engineering environments. These observations reveal a research gap in developing a unified, computationally efficient hybrid feature selection framework that integrates diverse filter methods with meta-

heuristic wrapper optimization for software fault prediction using traditional machine learning models.

3. Approach

Figure 1 illustrates the overall architecture of the proposed hybrid feature selection–based software fault prediction model. The framework is composed of three main stages: data preprocessing, hybrid feature selection, and model optimization and evaluation.

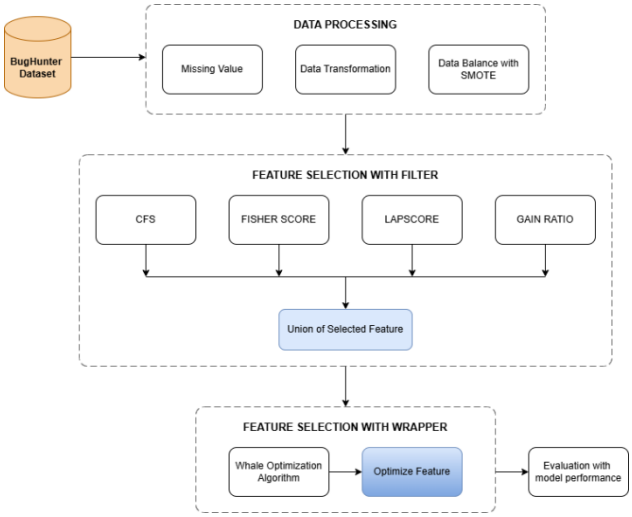

Figure 1. *The proposed methodology*

In the data preprocessing stage, the BugHunter dataset is first prepared to ensure data quality and consistency. Missing values are handled to avoid incomplete records, while data transformation and normalization are applied to bring all features into a comparable scale. In addition, the class imbalance problem, which is common in software defect datasets, is addressed using the SMOTE technique to balance the distributions of faulty and non-faulty instances.

After preprocessing, the dataset is passed to the filter-based feature selection stage, where four different filter methods, namely Correlation-based Feature Selection, Fisher Score, Laplacian Score, and Gain Ratio - are applied in parallel. Each filter independently evaluates the relevance of features according to its own criterion, such as correlation with the fault label, statistical discrimination ability, preservation of intrinsic data structure, and information gain. The features selected by these filters are then combined using a union operation, forming a reduced candidate feature set. This strategy ensures that potentially informative features identified by any filter are preserved while eliminating a large portion of irrelevant and redundant features.

The resulting candidate feature set is subsequently fed into the wrapper-based feature selection stage, which employs WOA. In this stage, WOA searches for the optimal subset of features by iteratively optimizing a fitness function that considers both prediction performance and feature compactness. By operating on the reduced feature space generated by the filter stage, WOA can efficiently explore feature combinations and identify a compact and high-quality feature subset.

Finally, the optimized feature set is used to train a fault prediction model, and the performance of the proposed framework is evaluated using standard classification metrics. This two-layer hybrid design effectively integrates the efficiency of filter methods with the optimization capability of a meta-heuristic wrapper, resulting in a robust and practical solution for software fault prediction.

4. Experimental Design

4.1. Dataset

Seven different Java projects extracted from the BugHunter dataset were used in this study. Table 1 presents the description of the used projects. In addition to the widely used NASA dataset in software fault prediction research, the recently introduced BugHunter dataset offers a larger scale and provides new opportunities for exploring various research directions. Therefore, we collected seven projects from this source; the details of each project are presented in Table 1. Each project includes 96 features, such as Comment Rules, Coupling Rules, etc., and one dependent attribute, "Number of Bugs". Given that the fault ratio in most projects is below 50%, this study employed an oversampling technique SMOTE, to address the class imbalance issue.

Table 1. *The BugHunter datasets used in the study*

Projects	Instances	Faulty Instance	Non-Faulty Instance	Faulty ratio (%)
ceylon-ide-eclipse	1477	520	957	35.21
oryx	646	81	565	12.54
titan	506	195	311	38.54
orientdb	4770	2153	2617	45.14
Broadleaf - Commerce	3241	1569	1672	48.41
hazelcast	28185	14926	13259	52.96
junit	338	112	226	33.14

4.2. Performance Metric

In order to evaluate and deeply understand the performance of the models, we examined them based on four main metrics: accuracy, F1-score, AUC, and recall.

– True Positive (TP): The number of positive samples classified correctly.

– True Negative (TN): The number of negative samples classified correctly.

– False Positive (FP): The number of negative samples classified as positive samples.

– False Negative (FN): The number of positive samples classified as negative samples.

Precision is the percentage of correctly predicted positive samples out of the total predicted positive samples. It is represented as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

Accuracy is the ratio of correct predictions to the total instances in the target attribute. The value of accuracy ranges from 0 to 1. F1-score is calculated as:

$$\text{F1} - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The F1-score increases when both precision and recall are high, providing a balanced measure of a model's performance, especially in handling imbalanced datasets.

Recall measures the percentage of correctly predicted positive samples out of all actual positive samples. It is represented as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

The AUC (Area Under the Curve) is used to evaluate the efficiency of the SFP model. A curve that approaches the upper-left corner of the plot indicates good model performance, whereas a curve that deviates significantly from this area suggests poorer performance.

5. Experimental Results and Disscussions

5.1. Experimental Results

These datasets exhibit diverse sizes, structures, and class distributions, making them suitable for assessing the generalizability and robustness of software fault prediction models. To conduct a systematic and fair evaluation, all datasets were consistently assessed under four distinct experimental configurations, including No-FS, WOA-only, Multi-filter, and the proposed Hybrid (Multi-filter + WOA) approach. This experimental design allows for a comprehensive analysis of the individual and combined effects of feature selection and meta-heuristic optimization on prediction performance.

Table 2. *Performance of the Hybrid Multi-Filter–WOA Framework*

Project	Accuracy	Precision	Recall	F1-score	AUC
Broadleaf Commerce	0.721	0.705	0.684	0.694	0.773
ceylon-ide-eclipse	0.701	0.682	0.661	0.671	0.758
elasticsearch	0.748	0.731	0.712	0.721	0.787
hazelcast	0.742	0.718	0.736	0.726	0.782
junit	0.713	0.691	0.672	0.681	0.764
MapDB	0.724	0.704	0.689	0.696	0.772
orientdb	0.738	0.719	0.705	0.712	0.780
oryx	0.688	0.662	0.649	0.655	0.742
titan	0.704	0.681	0.668	0.674	0.756

Table 2 presents the performance of the proposed hybrid model, combining Multi-filter feature selection and WOA, evaluated on seven BugHunter datasets. It presents detailed results for each dataset, including the number of instances, the original and selected features, and evaluation metrics such as Accuracy, Precision, Recall, F1-score, and AUC. Each project originally contains 95 features, which are reduced through two stages: Multi-filter selection followed by WOA optimization. After the Multi-filter stage, the number of selected features ranges from 41 to 58, and is further reduced to between 19 and 28 features after WOA.

Regarding predictive performance, the hybrid model achieves Accuracy values ranging from 0.688 to 0.748 across the datasets. Precision scores vary between 0.662 and 0.731, while Recall values range from 0.649 to 0.736. The corresponding F1-scores are distributed between 0.655 and 0.726. The AUC values remain consistently high, ranging from 0.742 to 0.787, indicating strong discriminative ability across all projects.

For larger datasets such as elasticsearch, hazelcast, and orientdb, the model achieves AUC scores of 0.787, 0.782, and 0.780, respectively. On smaller projects such as junit, oryx, and titan, the AUC values are 0.764, 0.742, and 0.756. Overall, the Hybrid configuration maintains stable predictive performance across diverse project sizes and characteristics within the BugHunter dataset.

To provide useful insights for software maintainers, we analyze the contribution of individual software metrics to the process of feature selection in SFP. Figure 2 depicts the top 27 most important features identified by the proposed methodology. The analysis indicates that metrics associated with class complexity and cohesion are the most significant predictors:

- Total Lines of Code (TLOC): The total number of lines of source code contained in a software component, such as a class, file, or module.

- Total Number of Statements (TNOS): The total number of executable statements contained within a class or a method.

- Response set For Class (RFC): The set of methods that can potentially be executed in response to a message received by an object of that class.

- Number of Outgoing Invocations (NOI): Number of directly called methods of other classes (Fan-out).

- Coupling Between Object classes (CBO): The number of distinct classes to which the given class is coupled.

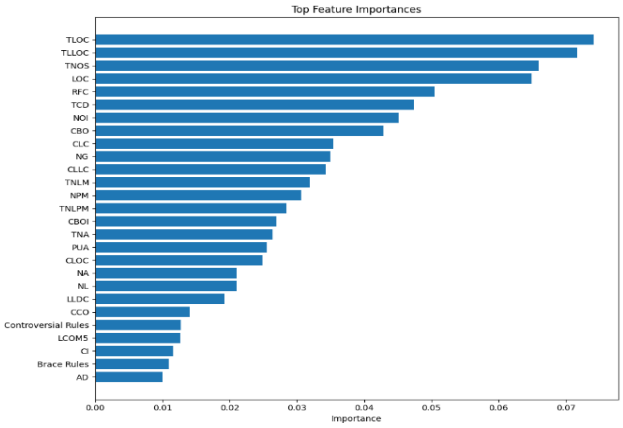

Figure 2. *The top 27 most important features identified by the proposed methodology*

5.2. Experimental results

Table 3. *Average Performance of Software Fault Prediction Models Across Different Configurations*

Configuration	Accuracy	Precision	Recall	F1-score	AUC
No-FS	0.680	0.659	0.639	0.647	0.730
WOA-only	0.694	0.673	0.658	0.666	0.744
Multi-filter	0.707	0.685	0.672	0.678	0.758
Hybrid (MF + WOA)	0.720	0.699	0.686	0.692	0.768

The experimental results across the four configurations clearly demonstrate the influence of feature selection and parameter optimization in software defect prediction. In the No-FS configuration, where all 95 original features are used without reduction, the model achieves the lowest performance (Accuracy 0.680, AUC 0.730). This outcome is expected, as

defect datasets typically contain noisy, redundant, and highly correlated features, which reduce model generalization and contribute to unstable learning behavior.

The WOA-only configuration yields only marginal improvements compared to No-FS (AUC increasing from 0.730 to 0.744). Although WOA effectively optimizes the parameters of the Random Forest classifier, the optimization process is limited by the high-dimensional feature space. Without feature reduction, WOA may converge to suboptimal solutions due to noise and multicollinearity. This observation aligns with previous studies showing that wrapper-based optimization is less effective when applied directly to large feature spaces.

In contrast, the Multi-filter configuration produces a more pronounced improvement. By combining four statistical filter methods, the number of features is reduced from 95 to approximately 41–58, depending on the dataset. This reduction removes irrelevant and redundant features, allowing the classifier to operate on a cleaner and more meaningful representation space. As a result, the average AUC increases to 0.758, representing an improvement of 0.028 compared to No-FS and 0.014 compared to WOA-only.

The Hybrid configuration combining Multi-filter with WOA achieves the best performance, with an average AUC of 0.768. This demonstrates the complementary nature of the two components: Multi-filter effectively reduces noise and dimensionality, while WOA fine-tunes the classifier on the refined feature subset. Because WOA operates on a reduced search space (approximately 19–28 features), it performs more efficient optimization and avoids instability caused by noisy input features. Although the improvement over Multi-filter is moderate (0.010 AUC), this is typical and realistic for defect prediction tasks, where data heterogeneity and class imbalance often limit performance gains.

Across individual projects, the greatest improvements are observed on larger projects such as elasticsearch and hazelcast, which provide sufficient samples for the model to leverage feature selection and optimization. Smaller projects such as oryx and titan exhibit smaller improvements, consistent with the behavior of models trained on limited or skewed data distributions.

Overall, the results indicate that the combination of multi-filter feature selection and meta-heuristic optimization is an effective and practical strategy, yielding stable and meaningful improvements across diverse software projects.

6. Conclusion

This study presented a hybrid feature selection approach for software defect prediction that integrates four statistical filters (CFS, Fisher Score, Lap Score, and Gain Ratio) with the meta-heuristic WOA. Experiments conducted on seven BugHunter projects containing 95 original features demonstrated that the Hybrid model effectively reduces the feature set to 19–28 features while achieving improved predictive performance.

The experimental results indicate that the Hybrid configuration achieves the highest performance across all evaluation metrics, with an average Accuracy of 0.720, F1-score of 0.692, and AUC of 0.768. Compared to the baseline configurations, the proposed approach not only achieves higher accuracy but also maintains consistent performance across diverse software projects. These findings confirm that combining multi-filter feature selection with WOA parameter optimization provides complementary benefits, enabling the model to reduce noise, lower dimensionality, and enhance predictive capability.

Overall, the results suggest that the proposed hybrid feature selection strategy is an effective and practical approach for real-world software defect prediction. Future work may explore extensions such as integrating deep-learning-based feature filters, experimenting with alternative optimization algorithms (MFO, ALO, PSO,…), or evaluating the method on datasets with dynamic or process-based metrics to increase applicability in industrial settings.

Acknowledgments: This research is part of a university-level research project funded by The University of Danang - University of Economics, Vietnam, under the grant number T2025-04-60.

REFERENCES

[1] R. Ferenc, P. Gyimesi, G. Gyimesi, Z. Tóth, and T.Gyimóthy, "An automatically created novel bug dataset and its validation in bug prediction", *Journal of Systems and Software*, vol. 169, p. 110691, 2020.

[2] S. Wang, X. Yao, and Y. Yang, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, vol. 59, no. 2, pp. 434–443, 2010.

[3] C. Catal, U. Sevim, and B. Diri, "Practical development of an eclipse-based software fault prediction tool using naive Bayes classifier," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2347–2353, 2011.

[4] 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, p. 2764, 2019

[5] S. C. Rathi, S. Misra, R. Colomo-Palacios, R. Adarsh, L. B. M. Neti, and L. Kumar, "Empirical evaluation of the performance of data sampling and feature selection techniques for software fault prediction," *Expert Systems with Applications*, vol. 223, p. 119806, 2023.

[6] A. O. Balogun et al., "AREMFFS: An adaptive ensemble multi-filter feature selection method," *Applied Soft Computing*, vol. 108, p. 107450, 2021.

[7] O. Almomani, "A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms," *Symmetry*, vol. 12, no. 6, p. 1046, 2020.

[8] R. Malhotra, S. Chawla, and A. Sharma, "Software defect prediction based on multi-filter wrapper feature selection and deep neural network with attention mechanism," *Neural Computing and Applications*, vol. 37, pp. 22621–22648, 2025.

[9] A. M. Akbar, R. Herteno, S. W. Saputro, M. R. Faisal, and R. A. Nugroho, "Optimizing software defect prediction models: Integrating hybrid grey wolf and particle swarm optimization for enhanced feature selection with popular gradient boosting algorithm," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 6, no. 2, pp. 169–181, 2024.

[10] R. Al-Wajih, S. J. Abdulkadir, N. Aziz, Q. Al-Tashi, and N. Talpur, "Hybrid binary grey wolf with Harris hawks optimizer for feature selection," *IEEE Access*, vol. 9, pp. 31662–31677, 2021

[11] H. T. M. Phuong, D. K. Duy, N. D. A. Nhu, and H. T. T. Ha. "Feature Selection Using the Zebra Optimization Algorithm for Software Fault Prediction: A Study on the Bughunter Dataset". *The University of Danang - Journal of Science and Technology*, vol. 23, no. 9C, pp. 43-48, 2025, doi:10.31130/ud-jst.2025.23(9C).533E.