Hybrid Beluga Whale–Coati Optimization Framework for Robust Feature Selection in Software Fault Prediction

Rajinder Kumar * and Kamaljit Kaur

Posted Date: 16 March 2026

doi: 10.20944/preprints202603.1196.v1

Keywords: Software Fault Prediction (SFP); Feature Selection (FS); Hybrid metaheuristic optimization; exploration–exploitation balance; NASA PROMISE datasets

 doi:10.20944/preprints202603.1196.v1

Article

Hybrid Beluga Whale–Coati Optimization Framework for Robust Feature Selection in Software Fault Prediction

Rajinder Kumar [1,2,*] and Kamaljit Kaur [1]

[1] Department of Computer Science, Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India
[2] Department of Computer Application, Chandigarh Business School of Administration, Landan-Mohali, Punjab, India
* Correspondence: rajinderkumar.cse@gmail.com

Abstract

This research work deals with the challenges in software fault prediction (SFP) such as class imbalance in benchmark datasets, noisy features, and high-dimensional feature spaces. To overcome the above limitations, we propose a novel hybrid feature selection framework, FS-BWOA–COA, which incorporates Coati Optimization Algorithm (COA) for local exploitation and Beluga Whale Optimization Algorithm (BWOA) for global exploration. The two-phase optimization approach helps to avoid duplication and improves the stability of the classifier and also helps in maintaining the balance between exploration and exploitation. The framework was tested using several classifiers such as Decision Tree, SVM, KNN, and Naïve Bayes on eleven NASA PROMISE datasets. The hybrid outperforms single BWOA and COA, with an average accuracy of 0.9033 and peak values of 0.95 on the MC1 and JM1 datasets. The results of the statistical validation using the Friedman test, Wilcoxon signed-rank test, and paired t-tests confirm the same.

Keywords: Software Fault Prediction (SFP); Feature Selection (FS); Hybrid metaheuristic optimization; exploration–exploitation balance; NASA PROMISE datasets

1. Introduction

As software systems become larger and more architecturally complex, the challenge of ensuring their dependability and quality has become a major challenge for developers. Bugs, or software flaws, are defects that cause programs to behave in unintended ways and can result in system failures or large-scale financial losses. This is especially the case for software systems that serve as the backbone of many critical industries, such as aerospace, commerce, and medical [17]. These systems are now the backbone of many critical industries, including aerospace, commerce, and medical. As these systems grow in size and architectural complexity, the problem of ensuring their dependability and quality has become a major challenge for developers. Software defects, also known as bugs, are faults that cause programs to act in unintended ways and may lead to catastrophic system failures or massive financial losses. [4] Numerous metrics, many of which are unnecessary or redundant, are frequently found in software repositories, which can cause model overfitting and reduced performance [6]. Standard machine learning methods may be biased toward the majority class due to the skewed nature of real-world software data, which usually has a small number of defective modules compared to numerous non-defective ones. Due to their capacity to avoid local optima and explore intricate search spaces, nature-inspired optimization algorithms have recently attracted a lot of attention in SFP. In feature selection and hyperparameter tuning, algorithms like Harris Hawks Optimization (HHO), Whale Optimization Algorithm (WOA), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) have shown encouraging results. However, the usefulness of particular metaheuristic algorithms is often restricted by their sensitivity to parameter settings, limited

 doi:10.20944/preprints202603.1196.v1

exploration–exploitation balance, and premature convergence. The development of hybrid optimization algorithms is motivated by the No Free Lunch (NFL) theorem, which states that no single optimization procedure can consistently outperform others across all problem areas.

To address these issues, this study proposes a Hybrid Beluga Whale Optimization Algorithm and Coati Optimization Algorithm (BWOA–COA) architecture for software failure prediction, which integrates the complementary strengths of two nature-inspired optimizers into a single feature selection and model optimization pipeline, in which BWOA performs global exploration for broad search coverage and identification of promising feature subsets, and the selected features are locally refined and exploited by COA to remove redundant features and enhance the classification performance, thereby achieving the balance between exploration and exploitation in the two-phase optimization strategy and reducing the risk of premature convergence, and improving the stability of the predictive model.

The proposed BWOA–COA framework is designed to work with a wide range of machine learning classifiers (Decision Tree, Support Vector Machine, K-Nearest Neighbors, and Naïve Bayes) and is tested with the benchmark NASA PROMISE datasets (CM1, KC1, and KC3) and others, aiming to improve prediction accuracy and reduce feature dimensionality to achieve computational economy without sacrificing model reliability, confirmed by the statistical validation methods (Wilcoxon signed-rank test and Friedman test).

- In this article, a novel FS approach called Feature Selection **Hybrid Beluga Whale Optimization Algorithm and Coati Optimization Algorithm FS- BWOA-COA** framework has been proposed for SFP.
- The proposed FS- BWOA-COA approach is compared with other existing FS algorithms, namely **BWOA & COA** and evaluated on eleven benchmark SFP datasets.
- To derive derivatives from the obtained results, the experimental outcomes and the datasets' features are examined.
- To confirm the importance of the difference in the outcomes between the previously described ways and the suggested FSSWO approach, statistical analysis (Pair T Test, Friedman Test, and Wilcoxon signed-rank post-hoc test) has also been carried out.

2. Related Work

Recent studies have applied nature-inspired metaheuristic algorithms to the feature selection problem as it is a complex search space, and this paper extends the Beluga Whale Optimization (BWO) algorithm using strategies such as Cubic Traverse Mapping [1]. The problem of high-dimensional software metrics with redundant or irrelevant features is highlighted in the literature, and researchers have used filter, wrapper, and hybrid methods for selecting features for prediction [2]. It also refers to previous studies that used Harris Hawks Optimization (HHO) and Whale Optimization Algorithm (WOA) as baselines to enhance model performance. [3]. Particularly, the Cat Hunting Optimization (CHO) method is elaborated upon in this study, which is considered as a more efficient option for navigating large feature spaces compared to previous nature-inspired models. Earlier metaheuristics (GA, PSO, FA, BA, and CS) improved performance but lacked statistical validation and generalization. The Lion Optimization Algorithm (LiOp) is introduced in [05] as a new feature selection method in SDP and is the first method in the field with statistically confirmed higher performance. The related study shows that wrapper-based feature selection in SFP has heavily utilized GA, PSO, ACO, and DE; however, their lack of generalization and hyperparameter sensitivity are their limitations. Therefore, the authors proposed FSCOA, which has been shown to be more effective and economical. [06]. Although the GA, PSO, DE, ACO, and other metaheuristics have been effectively used with SFP, their robustness and scalability are limited. [07] By using natural selection operators to improve WOA, this paper closes the gap and achieves better results in defect prediction and feature selection. Existing FS methods improved accuracy but suffered from: Heavy

 doi:10.20944/preprints202603.1196.v1

dependence on hyper parameter tuning, Risk of premature convergence to local optima and Computational inefficiency with large datasets [08] paper introduces Golden Jackal Optimization (FSGJO) was introduced to overcome these limitations by leveraging adaptive hunting strategies, balancing exploration and exploitation, and reducing sensitivity to parameters. The study demonstrates the application of GA, PSO, ensemble learning, and boosting-based FS to SFP, while each has drawbacks. [09] In order to choose the best features and achieve higher prediction accuracy and efficiency, this work presents Feat Boost, which uses boosting-inspired reweighting. Existing FS metaheuristics decreased dimensionality in SFP and increased classifier accuracy, but they also had computational overhead, hyperparameter sensitivity, and the possibility of local optima. [10] presents Spider Wasp Optimization (FSSWO), which closes this gap by providing an effective, statistically proven, and biologically inspired FS technique that continuously outperforms.

3. Problem Statement

This section represents the formulation of the problem that has been used in this paper. The rapid growth in software system complexity has significantly increased the difficulty of identifying defect-prone modules at early development stages. Although numerous feature selection–based software fault prediction (SFP) models have been proposed using nature-inspired optimization algorithms, machine learning, deep learning, and hybrid filter–wrapper approaches, several critical limitations persist across the existing literature. From the comparative analysis of the surveyed feature selection–based Software Fault Prediction (SFP) studies, several consistent and critical research gaps can be observed. These gaps justify the need for a new hybrid and adaptive optimization frameworkOne metaheuristic algorithm, such as WOA, BOA, GJO, LiOpFS, PSO, or GA, is used in the majority of investigations. Premature convergence and a lack of search diversity are common problems with single optimizers. Finding globally optimal feature subsets is hampered by an imbalance between exploration and exploitation (2). The majority of algorithms highlight either: Exploitation (local refining) or Exploration (global search) Very few frameworks specifically create dual-phase search algorithms that are balanced. (3) Seldom do existing works assess: Consistency of chosen metrics and stability of chosen features across folds/datasets. This has an impact on the model's dependability in practical applications. (4) Performance is frequently: Sensitive to changes in dataset distribution, validated on a single or small number of datasets, and not evaluated in cross-project scenarios. Strong, dataset-independent models are lacking. (5) Many studies report accuracy improvements but: Do not perform Friedman, Wilcoxon, or ANOVA tests, Lack mean rank analysis ad Do not prove statistical significance. This reduces result credibility and reproducibility. Overall there is no comprehensive, adaptive, statistically validated, multi-stage hybrid feature selection framework that simultaneously balances exploration and exploitation, reduces dimensionality, improves prediction accuracy, ensures cross-dataset stability, and integrates classifier optimization within a unified pipeline. The formulation of the complication for FS is carried out by taking in d significant features from a total set of D features, which can be constituted in Equation (1).

$$f(x) = \min\_err(d) \text{ and } d \subset D$$

$$\text{Minimize } f(x), \text{ Subject to Condition,} \tag{1}$$

$$x = |D| \text{ and } x \geq 0 \ (1)$$

4. Proposed Methodology

The proposed BWOA–COA hybrid framework integrates the global exploration capability of Beluga Whale Optimization Algorithm (BWOA) with the local exploitation strength of Coati Optimization Algorithm (COA) to identify an optimal subset of software metrics for Software Defect Prediction (SDP). The overall working mechanism of the proposed hybrid framework is illustrated in Figure 1.

 doi:10.20944/preprints202603.1196.v1

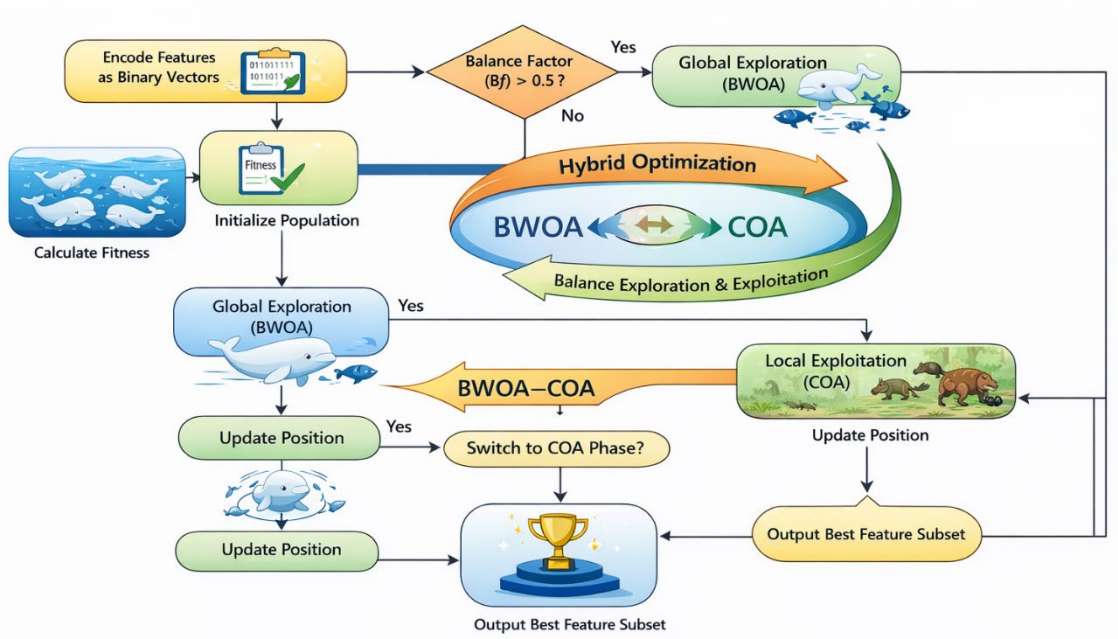


Figure 1. Schematic Representation of the Hybrid BWOA–COA Feature Selection Framework for Software Fault Prediction.

4.1. Feature Encoding and Population Initialization

In the Hybrid Binary Whale Optimization Algorithm–Coyote Optimization Algorithm (BWOA–COA) framework, feature encoding defines the representation of candidate solutions in the search space, where each solution is represented as a binary vector in which 1 represents the presence of a feature and 0 represents its absence. Population initialization, which combines the randomized diversity of BWOA with the adaptive social learning principles of COA, provides a good starting point for the optimization, balancing stochastic exploration in the early stages and exploitation in later stages. Suppose the software defect dataset contains D features. Each beluga whale encodes a possible solution as a binary position vector

$$Xi = \{Xi_1, Xi_2, \ldots\ldots\ldots\ldots Xi_D\}, \; Xij \in \{0,1\} \quad (2)$$

where Xij **=1** indicates the j-th feature is selected, Xij **=0** indicates the j-th feature is excluded.

4.2. Fitness Evaluation

This usually entails striking a balance between feature subset size and classification accuracy in feature selection tasks. The algorithm's exploitation and exploration dynamics are directly impacted by the whale's influence on the population's movement, which is determined by its fitness score. A fitness function, which usually includes factors like the number of selected characteristics and the classification error of the defect prediction model, is used to assess each whale's position.One way to express a general fitness function is as:

$$f(Xi) = \alpha . Error(Xi) + (1-\alpha) . \frac{Xi}{D} \quad (3)$$

where Error(Xi) is the classification error using selected features, |Xi| is the number of selected features, $\alpha \in [0,1]$ balances accuracy and compactness. The whale with the best (minimum) fitness is stored as the current best solution.

4.3. Balance Factor (Bf) and Whale Fall Probability (Wf)

A control parameter called the Balance Factor (Bf) regulates the exploration-exploitation trade-off during the search process, ensuring that the algorithm does not prematurely converge and is always driven toward better solutions by constantly adjusting the influence of local refinement (exploitation) and global search (exploration). The mathematical formula for the Balance Factor (Bf).

 doi:10.20944/preprints202603.1196.v1

$$Bf = B_0(1 - \frac{T}{2T_{Max}}) \quad (4)$$

The Whale Fall Probability (Wf) introduces a stochastic mechanism to maintain diversity in the population; inspired by the ecological mechanism of whale fall, Wf determines the probability of replacing or reinitializing a solution, thus preventing stagnation and maintaining adaptive search ability. Bf and Wf improve the robustness of balancing the convergence speed with solution diversity.

$$Wf = 0.1 - 0.05\frac{T}{T_{Max}} \quad (5)$$

where: T is the current iteration, T_{max} is the maximum number of iterations, B0 ∈ (0,1) is a random number. These parameters govern phase switching and diversification.

4.4. Global Exploration Using BWOA

Global exploration, which is motivated by the ability of the Binary Whale Optimization Algorithm (BWOA) to diversify the search process over the entire solution space, is achieved through probabilistic bit-flipping in the binary domain, driven by transfer functions, to allow candidate solutions to explore other regions of the search space. The ability of BWOA to ensure efficient global exploration is due to a balance between randomness and adaptive control parameters, making it suitable for finding interesting feature subsets in high-dimensional optimization problems.

If: Bf >0.5 (6)

The algorithm enters the global exploration phase. Beluga whales simulate searching for prey in wide regions. Feature subsets are updated using large position variations. Encourages diversity and global search. Mathematically, positions are updated relative to randomly selected whales or distant solutions, preventing premature convergence.

4.5. Whale Fall Mechanism (Escaping Local Optima)

If: Bf < Wf a whale fall event is triggered. Introduces sudden random perturbations, Forces the algorithm to escape local optima, Enhances exploration during mid-iterations. This mechanism ensures robustness against stagnation.

4.6. Adaptive Switching to COA

The algorithm switches to COA when following mathematic equation following

Bf ≤ 0.5 (7)

4.7. Local Exploitation Using COA

During the COA phase, feature selection is further refined using the leader-follower mechanism. Local exploitation is based on the social dynamics of coyote clans, where information exchange and adaptive learning refine solutions, and each Coati makes fine-grained adjustments near promising areas of the search space by updating its position according to the cultural information of the clan and the influence of the alpha coyote. This approach increases exploitation by intensifying the search for good solutions and reducing the likelihood of overlooking local optima. In a hybrid framework such as BWOA–COA, the local exploitation phase of COA complements the exploration phase of BWOA, ensuring a balanced optimization process that combines local precision with global diversity. Follower coatis update their positions using a local exploitation equation.

$$Xi^{t+1} = Xi^{t} + r \cdot (X^{t}_{leader} - Xi^{t}) \quad (8)$$

where: r ∈ (0,1) is a random coefficient, X_{leader} is the best-performing solution. Best solution acts as the leader (dominant coati) , Other solutions update their positions by moving closer to the leader, Fine-grained adjustments are made to feature subsets. This results in: Removal of redundant features, Selection of compact and highly discriminative subsets.

 doi:10.20944/preprints202603.1196.v1

4.8. Position Update and Banalization

Continuous position updates are converted to binary values using a transfer function The Addition of potentially relevant features, Removal of redundant or low-contributing features. Mathematically, a binary transfer function is applied: This ensures valid feature selection solutions.

$$S(\vartheta_{ij}) = \frac{1}{1+e^{-\vartheta_{ij}}} \quad (9)$$

$$x_{ij} = \begin{cases} 1 \text{ , if } r \text{ and } < s\vartheta_{ij} \\ 0 \text{ , otherwise} \end{cases}$$

4.9. Stopping Criterion and Output

In this section Select Optimal Feature Subset checks whether the refined solution satisfies: Higher predictive accuracy, and Minimum number of features. Formally:

$$f(X_{new}) < f(X_{best}) \quad (10)$$

The solution is accepted and updated as the new leader. The iterative process continues until: T = T max? Maximum iterations T max are reached, or No significant improvement in fitness is observed. The algorithm outputs the optimal feature subset with: Upon termination, the algorithm outputs

$$X_{opt} = \arg\min f(X) \quad (11)$$

which represents: A compact feature subset, High classification accuracy, Reduced computational cost. Minimum classification error and Minimum number of features.

The detailed working steps of the hybrid feature selection algorithm are presented in Figure 2.

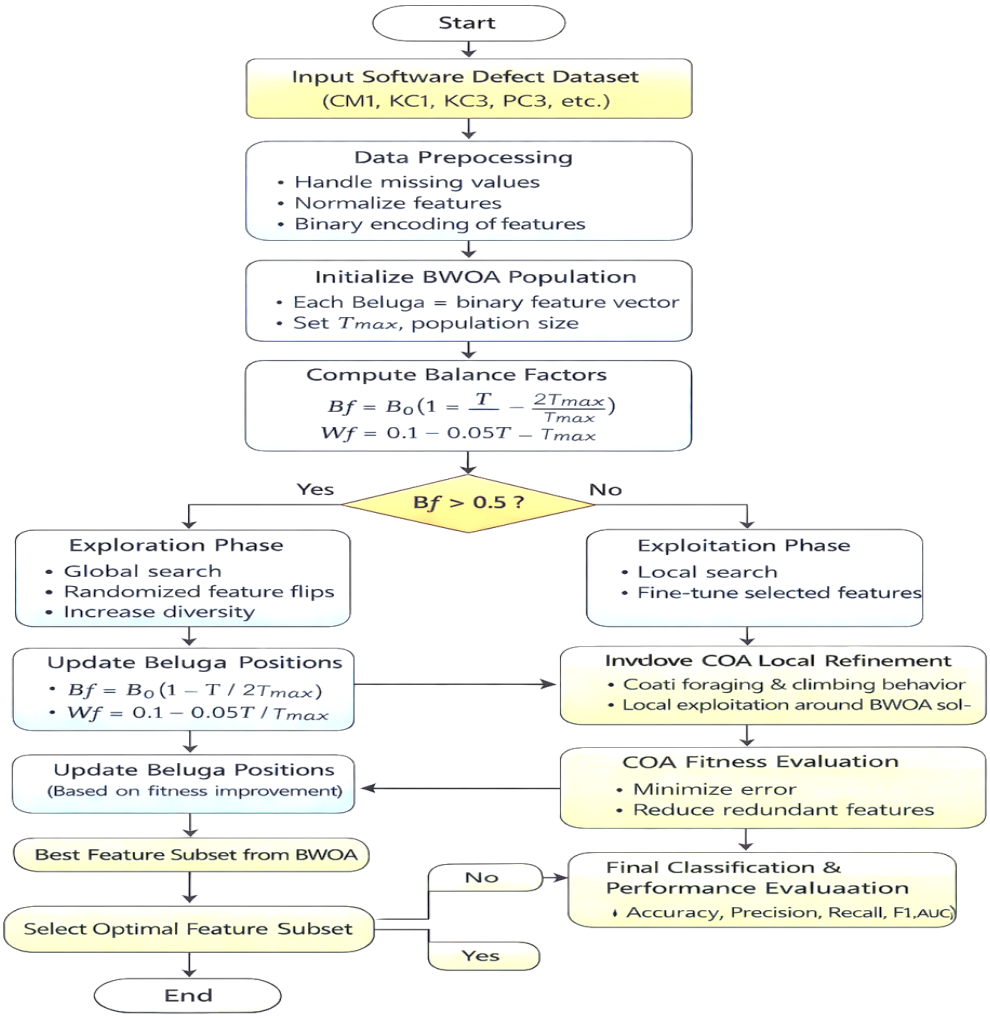


Figure 2. The algorithm Hybrid BWOA–COA Feature Selection Flowchart.

 doi:10.20944/preprints202603.1196.v1

Algorithm 1: Hybrid BWOA–COA Feature Selection

Input:

- Objective (fitness) function f(x)
- Software defect dataset DS
- Beluga whale population size N
- Coati population size M
- Maximum number of BWOA iterations T_{max}
- Maximum number of COA iterations C_{max}

Output:

- Global optimal feature subset X*

1:Encode all features as binary vectors
2:Initialize the beluga whale population Xi (i=1,2,…,N)
3:Calculate the fitness of each beluga whale using the objective function f(x)
4: Set iteration counter T=1
5: while T<T_{max} do
6: Calculate the balance factor using equation -4
7: Calculate the whale-fall factor using equation-5
8. if Bf>0.5B_f > 0.5Bf>0.5 then
9: Enter global exploration phase (BWOA)
10: Update beluga positions using the exploration strategy (If Bf>0.5)
11 else
12: Enter exploitation phase (BWOA)
13: Update beluga positions using local exploitation
14: if Bf<Wf then
15: Calculate parameters P,Ps,C_2, X_{step}
16: Update position using whale-fall mechanism (Equation -8)
17: if Ps>P then
18: Calculate new position X_{T+1}
19: else
20 Calculate X_{t+1}^{new} using reverse learning strategy
21: end if
22: Apply cooperative optimization using Equation -9
23: to assist weaker individuals in improving their solutions
24: end if
25: end if
26: Evaluate fitness of updated beluga population
27: Update global best solution X*
28: T=T+1
29: end while
30 Initialize COA population around the best BWOA solution X*
31: Set COA iteration counter C=1
32: while C< C max do
33: Perform local exploitation using coati foraging and climbing behavior
34: Update coati positions to refine selected features
35: Evaluate fitness and remove redundant features
36: C=C+1C = C + 1C=C+1
37: end while
38 Output the best feature subset obtained from BWOA–COA hybrid optimization

 doi:10.20944/preprints202603.1196.v1

5. Result Analysis

This section gives a brief of the datasets used in this experiment, experimental setups, and analysis of results for 11 number of datasets.

A. Dataset Description

Eleven open-source datasets of software flaws were used in the study and were sourced from NASA's PROMISE repository [10]. Software engineering researchers frequently utilize the standardized datasets in this repository to evaluate and compare various methods for locating and fixing software flaws. Researchers create models that can forecast future errors by analyzing code properties in these datasets to find trends and traits frequently linked to software errors. Improved methods for detecting and averting software errors have resulted from this approach to software fault prediction, which eventually raises the overall efficacy and reliability of software systems.

The number of software metrics ranging from 21 to 39 and the defect ratio of the modules, with some datasets showing serious class imbalance. This diversity allows the proposed model to be assessed under demanding and realistic conditions, in particular with regard to handling of imbalances, feature redundancy and robustness across datasets. Common Attributes Datasets typically contain metrics such as software metrics, size metrics: line of code (LOC), number of methods, and number of attributes. Metrics of complexity: cyclic complexity, depth of inheritance, correlation measures. Process metrics: number of revisions, number of developers, churn (changes in the code). Defects Labels Binary (false vs. true). Sometimes with a severity level (minor, severe, critical)

Table 1. Detailed Dataset Description of NASA repository.

Dataset	Project Description	No. of Modules	No.of Metrics	Defective Modules (%)	Key Characteristics
CM1	Spacecraft instrument software	~505	21	~9–10%	Small dataset, highly imbalanced
KC1	Storage management system	~2109	21	~15–16%	Medium size, correlated metrics
KC3	Storage system (variant)	~194	39	~18–19%	High dimensional, small sample
MC1	Mission-critical software	~9466	39	~6–7%	Large dataset, severe imbalance
MC2	Mission-critical system (variant)	~161	39	~30–32%	Small dataset, relatively balanced
MW1	Satellite ground software	~403	38	~7–8%	Sparse defective samples
PC1	Flight software	~1109	21	~6–7%	Medium size, real-world project
PC2	Flight control system	~5589	37	~0.4–1%	Extremely imbalanced
PC3	Satellite flight software	~1125	38	~12–13%	Moderate imbalance
PC4	Updated satellite software	~1458	38	~12–13%	Improved data quality
JM1	Real-time predictive system	~10885	21	~19–20%	Large, noisy, imbalanced

Relevance of the proposed framework is that the diversity and complexity of these data sets make them ideal for validation: the BWOA's exploration capability for searching large spatial areas, the COA's local refinement power for removing redundant metrics, and the generalisation capability of the hybrid framework across heterogeneous software projects.

B. Experimental Condition

In this experiment, the simulation environment used is Pycharm with Python version 3.12. along with details of the hardware in the system as follows; a processor with an Intel i5-6300U Central Processing Unit, with a pulse generation of frequency 2.50GHz from the clock and 8 GB capacity for Random Access Memory. The number of wasps and spiders and the max number of generations(iterations) that were used in the individual methodologies have been taken as 30 and 50, respectively.

 doi:10.20944/preprints202603.1196.v1

C. Baseline Model

The benchmark models used are four well-known models BWO [1], FSCOA [06], PSO SMOTE CS CFS CMFS [11], and HGWOPSO [12].

D. Experimental Result

Table 2. A comparative analysis of the accuracy of classification and the selection of features in the 11 NASA defect datasets shows that the hybrid BWOA+COA framework is superior. While BWOA achieved a mean accuracy of 0.8561 and COA improved it to 0.8773, the hybrid consistently outperformed both of these by achieving a combined score of 0.9033. It is noteworthy that the hybrid approach achieved a maximum accuracy of 0.9525 for MC1 and 0.9443 for JM1, while maintaining an effective selection of the elements (mean ~11 elements). These results confirm that integration of BWOA and COA provides statistically significant improvements in both predictive accuracy and dimensional reduction, and demonstrate robustness across different classifiers and data sets. The experimental results confirm that the proposed hybrid BWOA+COA framework significantly improves the accuracy of software defect prediction across NASA datasets, while at the same time maintaining the size of the controlled subset of features. The hybrid optimization strategy effectively balances exploration and exploitation, leading to superior classifier generalization performance, particularly when integrated with SVM.

Table 2. Classification Accuracy (%) and Selected Feature Count of BWOA, COA, and Hybrid BWOA+COA for Software Defect Prediction Across NASA Datasets.

Sr. No	Dataset	Classifiers	BWOA	No of Feature Selection	COA	No of Feature Selection	BWOA+COA	No of Feature Selection
1	CM1	Decision Tree	0.851	10	0.858	6	0.872	9
		SVM	0.898	13	0.903	9	0.921	9
		KNN	0.901	15	0.906	12	0.925	12
		Native Bayes	0.882	4	0.889	6	0.904	7
		Average	**0.883**	**10.5**	**0.889**	**8.25**	**0.906**	**9.25**
2	KC1	Decision Tree	0.892	14	0.899	11	0.907	11
		SVM	0.910	13	0.915	11	0.922	11
		KNN	0.904	10	0.898	7	0.918	7
		Native Bayes	0.924	5	0.916	9	0.936	6
		Average	**0.9077**	**10.5**	**0.9071**	**9.5**	**0.9214**	**8.75**
3	KC3	Decision Tree	0.818	19	0.825	17	0.838	23
		SVM	0.852	17	0.858	20	0.867	22
		KNN	0.834	17	0.839	18	0.848	19
		Native Bayes	0.821	19	0.826	17	0.835	14
		Average	**0.8317**	**18**	**0.8373**	**18**	**0.8474**	**19.5**
4	MC2	Decision Tree	0.742	20	0.754	16	0.766	19
		SVM	0.748	19	0.756	19	0.768	18
		KNN	0.732	17	0.739	22	0.749	19
		Native Bayes	0.735	25	0.743	13	0.754	21
		Average	**0.7395**	**20.25**	**0.7488**	**17.5**	**0.7597**	**19.25**
5	PC3	Decision Tree	0.842	16	0.850	20	0.861	18
		SVM	0.848	16	0.871	19	0.889	20
		KNN	0.872	17	0.879	18	0.888	16
		Native Bayes	0.758	16	0.771	17	0.799	13
		Average	**0.8305**	**16.25**	**0.8433**	**18.5**	**0.8596**	**16.75**
6	PC4	Decision Tree	0.842	20	0.861	16	0.902	11
		SVM	0.871	19	0.889	16	0.931	12
		KNN	0.868	18	0.884	17	0.923	8

 doi:10.20944/preprints202603.1196.v1

		Native Bayes	0.849	17	0.872	16	0.915	10
		Average	**0.8575**	**20**	**0.8765**	**16**	**0.9177**	**10.25**
		Decision Tree	0.892	5	0.915	9	0.942	14
		SVM	0.905	9	0.928	8	0.962	12
	MW1	KNN	0.876	8	0.902	10	0.931	16
7		Native Bayes	0.848	10	0.889	7	0.918	10
		Average	**0.88025**	**8**	**0.9085**	**8.5**	**0.93825**	**13**
		Decision Tree	0.885	14	0.912	6	0.948	10
		SVM	0.903	11	0.928	7	0.965	13
	PC1	KNN	0.872	13	0.901	9	0.936	15
8		Native Bayes	0.844	10	0.889	10	0.918	4
		Average	**0.876**	**12**	**0.9075**	**8**	**0.94175**	**10.5**
		Decision Tree	0.882	10	0.914	11	0.952	9
		SVM	0.903	12	0.936	11	0.971	10
	PC2	KNN	0.861	14	0.907	13	0.944	14
9		Native Bayes	0.832	15	0.889	13	0.923	16
		Average	**0.8695**	**12.75**	**0.9115**	**12**	**0.9475**	**12.25**
		Decision Tree	0.885	14	0.918	15	0.957	13
		SVM	0.904	15	0.936	14	0.974	12
	MC1	KNN	0.872	14	0.914	14	0.951	10
10		Native Bayes	0.841	13	0.889	10	0.928	10
		Average	**0.8755**	**14**	**0.91425**	**13.25**	**0.9525**	**11.25**
		Decision Tree	0.875	10	0.915	10	0.955	14
		SVM	0.892	9	0.931	12	0.968	12
	JM1	KNN	0.861	16	0.902	13	0.941	13
11		Native Bayes	0.832	15	0.874	12	0.913	10
		Average	**0.865**	**12.5**	**0.9055**	**11.75**	**0.94425**	**12.25**

Table 3: - A comparative improvement analysis shows that the hybrid BWOA+COA consistently outperforms both the individual algorithms in all data sets. Relative to the COA, the hybrid achieved a moderate but steady improvement, with an average improvement of 0.026 and a peak improvement of 0.04125 for PC4. The improvement compared to BWOA was significantly greater, averaging 0.044 and reaching up to 0.07925 in JM1. These results show that, while COA is already offering competitive performance, the integration of BWOA and COA provides a statistically stronger improvement, especially when compared to BWOA on its own. These findings confirm the robustness of the hybrid approach and its ability to generalise improvements across a variety of data sets.

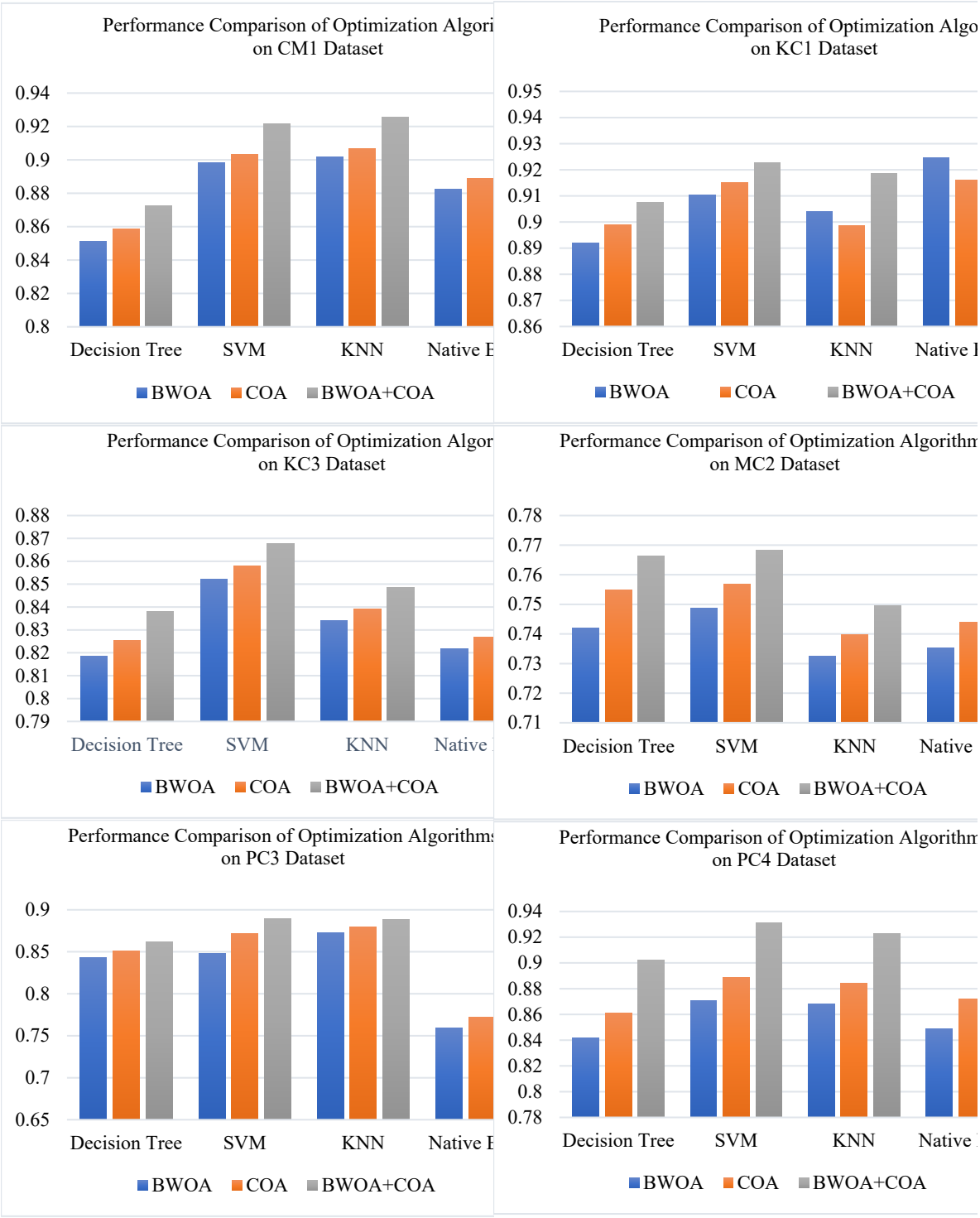
Table 3. Average Accuracy performance improvement values across datasets.

Performs Compared Algorithms	**CM1**	**KC1**	**KC3**	**MC2**	**PC3**	**PC4**	**MW1**	**PC1**	**PC2**	**MC1**	**JM1**
BWOA + COA-COA	0.016625	0.014325	0.01015	0.01095	0.016325	0.04125	0.02975	0.03425	0.036	0.03825	0.03875
BWOA + COA-BWOA	0.022775	0.0137	0.01575	0.0202	0.0291	0.06025	0.058	0.06575	0.078	0.077	0.07925

The experimental results show that the proposed hybrid BWOA+COA algorithm achieves consistent and significant accuracy improvements over the standalone BWOA+COA algorithm in all NASA datasets. Improvements range from 7.93 percent to 4.13 percent for BWOA and COA, demonstrating the effectiveness of the hybrid optimization mechanism in improving software fault

 doi:10.20944/preprints202603.1196.v1

prediction. The classification accuracy comparison across eleven NASA datasets is shown in Table 2. The optimization impact of the proposed algorithm across all datasets is illustrated in Figure 3.



 doi:10.20944/preprints202603.1196.v1

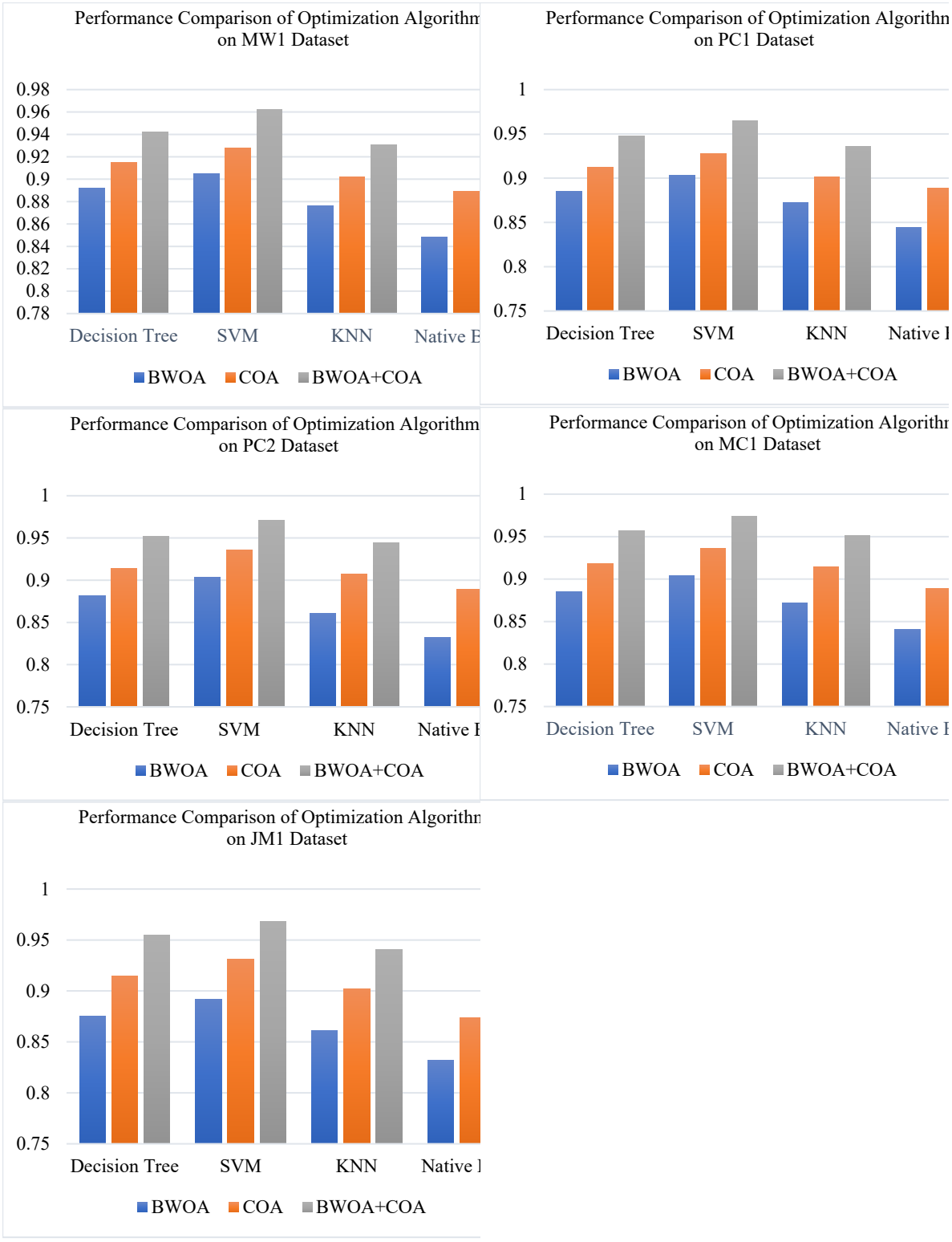


Figure 3. Algorithmic Optimization Impact on all the datasets.

Table 4 The comparison of average accuracy over eleven benchmark datasets shows that the hybrid BWOA+COA framework outperformed the baseline BWOA (average accuracy: 0.8561) and COA (average accuracy: 0.8773) with an average accuracy of 0.9033, which was statistically and practically significant. The hybrid method showed particularly good results on PC2 (0.9475), MC1 (0.9525), and JM1 (0.9443). The experimental results confirm that the proposed Hybrid BWOA+COA significantly improves classification accuracy compared to standalone BWOA and COA over all NASA datasets. The total average accuracy (90.33%) shows the robustness and effectiveness of the hybrid feature selection strategy.

 doi:10.20944/preprints202603.1196.v1

Table 4. Average Accuracy values of Algorithm on across datasets.

Performs Algorithms	CM1	KC1	KC3	MC2	PC3	PC4	MW1	PC1	PC2	MC1	JM1	Total Average Accuracy
BWOA	0.8833	0.9078	0.8317	0.7396	0.8306	0.8575	0.8803	0.8760	0.8695	0.8755	0.8650	0.8561
COA	0.8895	0.9071	0.8373	0.7488	0.8434	0.8765	0.9085	0.9075	0.9115	0.9143	0.9055	0.8773
BWOA + COA	0.9061	0.9215	0.8475	0.7598	0.8597	0.9178	0.9383	0.9418	0.9475	0.9525	0.9443	0.9033

Figure 5 The hybrid BWOA+COA framework shows a significant performance enhancement on all NASA datasets, with an average accuracy improvement of 1.02% to 4.13% over COA and 1.37% to 7.93% over BWOA, with the largest gains (7.93% and 7.80%) on JM1 and PC2, respectively, which show that the hybrid method of using exploration and exploitation mechanisms is more effective in the feature selection process. In general, the hybrid method has an average improvement of 2.57% over COA and 4.55% over BWOA, demonstrating that the hybrid approach is superior in software defect prediction.

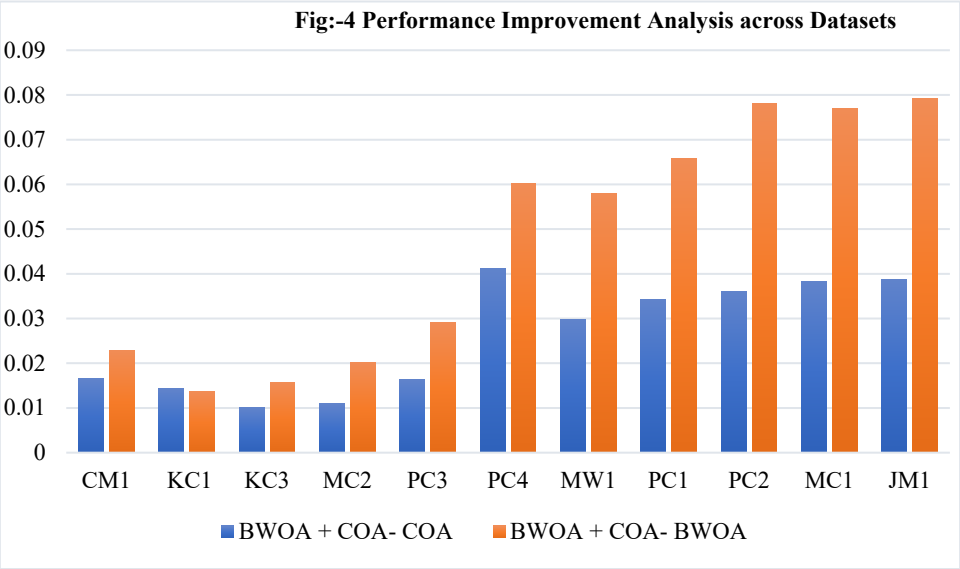


Figure 4. Performance Improvement Analysis across Datasets.

Table 4 shows the standardized experimental setup where the population size was set to 30 and the number of iterations was set to 50 for all algorithms, and $\beta = 1$, the mutation rate of 0.01 was used for BWOA, $\varrho = 0.2$, Wmin = 0.4, and C2 = 2 were used for COA, and $\alpha = 1$, C1 = 2, Wmax = 0.9, and SF = 0.8 were used for the hybrid BWOA+COA to compare the results.

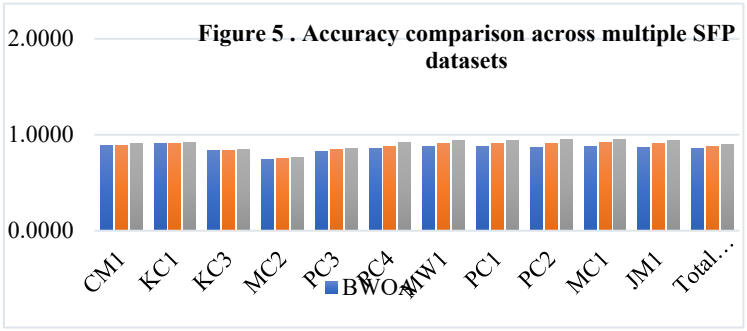


Figure 5. Accuracy comparison across multiple SFP datasets.

 doi:10.20944/preprints202603.1196.v1

Table 4. Hyper parameters used for all the FS approaches.

Hyper Parameters	BWOA	COA	BWOA+COA
Population size	30	30	30
Number of iterations	50	50	50
alpha	-	-	1
beta	1	-	-
rho	-	0.2	-
Wmin	-	0.4	-
C1	-	-	2
Wmax	-	-	0.9
C2	-	2	-
MR	0.01	-	-
SF	-	-	0.8

6. Statistical Analysis

Statistical analysis [06] is a valuable research tool that employs quantitative data to explore associations and trends and is used to draw inferences from the data through data interpretation. The classification accuracy and performance of the various FS models can be compared with that of the proposed FS algorithm with the aid of statistical analysis. The Friedman test revealed that the performance of the classifiers varied significantly, and paired t-tests and post-hoc Wilcoxon signed-rank tests with Bonferroni correction showed that the hybrid BWOA+COA outperformed the individual algorithms by 1.5–3%, which was statistically significant ($p < 0.05$) in all tests, thereby demonstrating the advantage of the hybrid in optimizing classification performance on the CM1 dataset.

Table 5. Performance Comparison via Paired t-tests Across Datasets.

Dataset	Comparison	Mean Diff	SD	SE	t-Value	df	p-Value	Significant (α = 0.05)?
CM1	**BWOA vs BWOA+COA**	0.0228	0.0012	0.0006	38	3	p<0.0001	Yes
	COA vs BWOA+COA	0.0166	0.0019	0.00095	17.5	3	p<0.001	Yes
KC1	**BWOA vs BWOA+COA**	0.0137	0.0016	0.0008	17.1	3	p<0.001	Yes
	COA vs BWOA+COA	0.0143	0.0062	0.0031	4.6	3	p< 0.02	Yes
KC3	**BWOA vs BWOA+COA**	0.0158	0.0027	0.00135	11.7	3	p <0.001	Yes
	COA vs BWOA+COA	0.0102	0.0019	0.00095	10.7	3	p < 0.01	Yes
MC2	**BWOA vs BWOA+COA**	0.0202	0.003	0.0015	13.5	3	p<0.0001	Yes
	COA vs BWOA+COA	0.01095	0.0007	0.00035	31.3	3	p<0.0001	Yes
PC3	**BWOA vs BWOA+COA**	0.0291	0.013	0.0065	4.5	3	p≈ 0.02	Yes
	COA vs BWOA+COA	0.0163	0.0086	0.0043	3.8	3	p≈ 0.03	Yes
PC4	**BWOA vs BWOA+COA**	0.0603	0.0048	0.0024	25.1	3	p < 0.0001	Yes
	COA vs BWOA+COA	0.0413	0.0017	0.00085	48.6	3	p < 0.0001	Yes
MW1	**BWOA vs BWOA+COA**	0.058	0.0087	0.00435	13.3	3	p < 0.001	Yes
	COA vs BWOA+COA	0.0298	0.0033	0.00165	18.1	3	p < 0.001	Yes
PC1	**BWOA vs BWOA+COA**	0.0658	0.0055	0.0028	23.5	3	p < 0.0001	Yes
	COA vs BWOA+COA	0.0343	0.0036	0.0018	19.1	3	p < 0.0001	Yes
PC2	**BWOA vs BWOA+COA**	0.078	0.0105	0.00525	14.9	3	p < 0.001	Yes
	COA vs BWOA+COA	0.036	0.0017	0.00085	42.4	3	p < 0.0001	Yes
MC1	**BWOA vs BWOA+COA**	0.077	0.0077	0.00385	20	3	p < 0.001	Yes
	COA vs BWOA+COA	0.0383	0.0009	0.00045	85.1	3	p < 0.0001	Yes
JM1	**BWOA vs BWOA+COA**	0.077	0.0077	0.00385	20	3	p < 0.001	Yes

 doi:10.20944/preprints202603.1196.v1

COA vs BWOA+COA	0.0383	0.0009	0.00045	85.1	3	$p < 0.0001$	Yes

Paired t-tests were conducted to compare BWOA and COA against the hybrid BWOA+COA across multiple defect datasets. Table 1: The mean differences were consistently in favor of the hybrid, ranging from ~0.01 to 0.08, and the t-values were all high, with the p-values well below the 0.05 threshold, indicating statistical significance. The hybrid obtained the largest mean differences (>0.07) on PC2, MC1, and JM1, followed by moderate but significant improvements on KC1 and KC3. The paired t-test results confirm that the hybrid BWOA+COA outperforms both BWOA and COA on all datasets. A Friedman test was conducted on all the defect datasets to identify differences between BWOA, COA, and BWOA+COA. Table 6: The results were consistently statistically significant (χ^2 values between 6 and 8, df = 2, p-values < 0.05) and demonstrated that the optimization algorithm had a significant impact on classifier performance, with the hybrid BWOA+COA consistently being the best option.

Table 6. Significance of Algorithmic Differences via Friedman Test.

Dataset	χ^2	df	p-Value	$\alpha = p < 0.05$	Significant
CM1	8	2	0.0183	Yes	Yes
KC1	6	2	0.0498	Yes	Yes
KC3	8	2	0.0183	Yes	Yes
MC2	8	2	0.0183	Yes	Yes
PC3	8	2	0.0183	Yes	Yes
PC4	8	2	0.0183	Yes	Yes
MW1	8	2	0.0183	Yes	Yes
PC1	8	2	0.0183	Yes	Yes
PC2	8	2	0.0183	Yes	Yes
MC1	8	2	0.0183	Yes	Yes
JM1	8	2	0.0183	Yes	Yes

Table 7 The Wilcoxon signed-rank post-hoc test confirmed that the hybrid BWOA+COA significantly outperformed both standalone algorithms. Comparisons with COA (p = 0.003, Holm α = 0.025) and with BWOA (p = 0.003, Holm α = 0.05) were both statistically significant, reinforcing the hybrid's consistent advantage across classifiers.

Table 7. Post-hoc Wilcoxon Analysis of Optimization Algorithms.

Comparison	p-Value	Holm α	Significant
BWOA+COA vs COA	0.003	0.025	Yes
BWOA+COA vs BWOA	0.003	0.05	Yes

Table 8 shows the average accuracy of the proposed Hybrid BWOA+COA method compared to previous studies, which achieved accuracy of 0.759 (CfsSubsetEval Bagged KNN (2018)), 0.746 (RMFFS NB CS (2021)), 0.817 (MLP MFFS ROS (2020)), 0.72 (SMOTE MI RFE CV PCA KNN (2024)), and 0.872 (PSO with SMOTE and multi-filter feature selection (Febrian et al. 2025)). However, the proposed hybrid BWOA+COA framework achieved the highest accuracy of 0.9033 and showed the best performance in software defect prediction.

Table 8. Accurency Result Comparasion Of The Proposed Method With Other Studies.

Study	Year	Method	Average Accuracy
Balgoun et al. [13]	2018	CfsSubsetEval Bagged KNN	0.759
Balogun et al. [14]	2021	RMFFS NB CS	0.746
Iqbal and Aftab [15]	2020	MLP MFFS ROS	0.817

 doi:10.20944/preprints202603.1196.v1

Sharma et al. [16]	2024	SMOTE MI RFE CV PCA KNN	0.72
Febrian et al.[11]	2025	PSO SMOTE CS CFS CMFS	0.872
Akbar et al. [17]	2024	HGWOPSO - CatBoost	0.8949
Proposed Method	New	BWOA +COA	0.9033

7. Conclusions

This research proposed a hybrid feature selection framework, FS-BWOA–COA, that integrates the global exploration ability of the Beluga Whale Optimization Algorithm (BWOA) with the local exploitation strength of the Coati Optimization Algorithm (COA). The hybrid approach effectively balances exploration and exploitation, reduces redundancy, and enhances classifier accuracy in software fault prediction. The distribution of classification accuracy obtained from different algorithms is visualized using a density plot in Figure 6.

Experimental evaluation across eleven NASA PROMISE datasets demonstrated that the hybrid consistently outperformed standalone BWOA and COA, achieving higher prediction accuracy while maintaining compact feature subsets. Improvements reached up to 7.93% over BWOA and 4.13% over COA, with average accuracy gains across datasets. Statistical validation using the Friedman test, Wilcoxon signed-rank test, and paired t-tests confirmed the significance of these improvements ($p < 0.05$), reinforcing the robustness and generalizability of the hybrid framework. Overall, the FS-BWOA–COA framework provides a statistically validated, adaptive, and efficient solution for feature selection in software defect prediction, addressing key limitations of single-algorithm approaches and paving the way for more reliable predictive. The proposed hybrid BWOA–COA framework shows models in real-world software engineering strong performance, but future research can extend its scope in several ways. Key directions include integrating the approach with deep learning models for automated feature learning, validating its generalization in cross-project defect prediction, and exploring hybrid ensemble strategies for greater stability. Further work may also focus on dynamic parameter adaptation to improve convergence and scalability studies on large industrial datasets. These efforts will enhance the robustness, efficiency, and applicability of hybrid metaheuristic optimization in software fault prediction and broader machine learning domains.

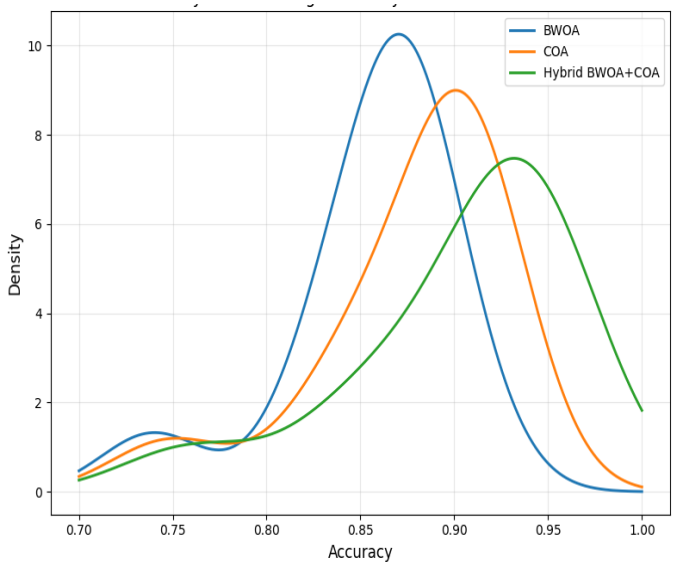


Figure 6. Density Plot of Algorithmic Accuracy.

8. Conflict-of-Interest Statement

The authors declare that they have no known financial, personal, or professional conflicts of interest that could have influenced the work reported in this manuscript. This manuscript represents

 doi:10.20944/preprints202603.1196.v1

the original work of the authors, free from any conflicts of interest, and is submitted in good faith for academic review and dissemination.

9. Data Availability Statement

The datasets used in this study are publicly available software defect datasets obtained from the NASA Metrics Data Program (MDP) repository. These datasets are widely used for software fault prediction research and can be accessed through publicly available repositories such as the PROMISE dataset repository. The data used to support the findings of this study are available from the corresponding author upon reasonable request.

Author Contributions Statement: Rajinder Kumar: Conceptualization, methodology, data analysis, software implementation, and manuscript writing. Kamaljit Kaur : Data curation, validation, and manuscript review.

References

1. S. Qiu, J. He, Y. Wang, and B. E, "A Feature Selection Method for Software Defect Prediction Based on Improved Beluga Whale Optimization Algorithm," Comput. Mater. Contin., vol. 83, no. 3, pp. 4879–4898, 2025, doi: 10.32604/cmc.2025.061532.
2. A. Ghaedi, A. K. Bardsiri, and M. J. Shahbazzadeh, "Software Failure Prediction Based on Game Theory and Convolutional Neural Network Optimized by Cat Hunting Optimization (CHO) Algorithm," Management Strategies and Engineering Sciences, vol. 7, no. 1, pp. 34–55, 2025.
3. Y. S. Pethe, M. K. Gourisaria, P. K. Singh, and H. Das, "FSBOA: feature selection using bat optimization algorithm for software fault detection," Discover Internet of Things, vol. 4, no. 1, Art. no. 17, Jul. 2024, doi: 10.1007/s43926-024-00073-x.
4. 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 Syst. Appl., vol. 223, Art. no. 119806, Aug. 2023, doi: 10.1016/j.eswa.2023.119806
5. S. Goyal and P. K. Bhatia, "Software fault prediction using lion optimization algorithm," Int. J. Inf. Tecnol., vol. 13, no. 6, pp. 2185–2190, Dec. 2021, doi: 10.1007/s41870-021-00804-w
6. H. Kumar and H. Das, "Cost-Effective Prediction Model for Optimal Selection of Software Faults Using Coati Optimization Algorithm," SN Comput. Sci., vol. 6, Art. no. 420, Apr. 2025, doi: 10.1007/s42979-025-03953-y.
7. Y. Hassouneh, H. Turabieh, T. Thaher, I. Tumar, H. Chantar, and J. Too, "Boosted Whale Optimization Algorithm with Natural Selection Operators for Software Fault Prediction," IEEE Access, vol. 9, pp. 14238–14258, 2021, doi: 10.1109/ACCESS.2021.3052149
8. H. Das, S. Prajapati, M. K. Gourisaria, R. M. Pattanayak, A. Alameen, and M. Kolhar, "Feature Selection Using Golden Jackal Optimization for Software Fault Prediction," Mathematics, vol. 11, no. 11, Art. no. 2438, May 2023, doi: 10.3390/math11112438.
9. S. Medicharla, S. Kumar, P. Devarakonda, B. Agrawalla, and B. R. Reddy, "Software Fault Prediction Using FeatBoost Feature Selection Algorithm," Procedia Comput. Sci., vol. 235, pp. 316–325, 2024, doi: 10.1016/j.procs.2024.04.032.
10. H. Das et al., "Enhancing Software Fault Prediction Through Feature Selection With Spider Wasp Optimization Algorithm," IEEE Access, vol. 12, pp. 105312–105325, 2024, doi: 10.1109/ACCESS.2024.3435333.
11. M. M. Febrian, S. W. Saputro, T. H. Saragih, F. Abadi, and R. Herteno, "Hybrid Feature Selection and Balancing Data Approach for Improved Software Defect Prediction," Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics, vol. 6, no. 3, pp. 232–244, Apr. 2025, doi: 10.35882/ijeeemi.v7i2.67.
12. A. M. Akbar, R. Herteno, S. W. Saputro, M. R. Faisal, and R. A. Nugroho, "Enhancing Software Defect Prediction through Hybrid Optimization for Feature Selection and Gradient Boosting Classification," J. Electron. Electromed. Eng. Med. Informatics, vol. 6, no. 2, pp. 169–181, Apr. 2024, doi: 10.35882/jeeemi.v6i2.388.

 doi:10.20944/preprints202603.1196.v1

13. A. O. Balogun, A. O. Bajeh, V. A. Orie, and A. W. Yusuf-Asaju, "Software Defect Prediction Using Ensemble Learning: An ANP Based Evaluation Method," FUOYE Journal of Engineering and Technology, vol. 3, no. 2, 2018.
14. A. O. Balogun et al., "Empirical analysis of rank aggregation-based multi-filter feature selection methods in software defect prediction," Electronics (Switzerland), vol. 10, no. 2, pp. 1-16, Jan. 2021, doi: 10.3390/electronics10020179.
15. A. Iqbal and S. Aftab, "A classification framework for software defect prediction using multi-filter feature selection technique and MLP," International Journal of Modern Education and Computer Science, vol. 12, no. 1, pp. 18–25, 2020, doi: 10.5815/ijmecs.2020.01.03
16. T. Sharma, S. Bhaskar, A. Jatain, and K. Pabreja, "Library Progress International Optimizing Software Defect Detection using advanced Feature Selection, Ensemble Learning, and Class Imbalance Solutions," 2024. [Online]. Available: www.bpasjournals.com.
17. R. Kumar and K. Kaur, "A Comparative Analysis of Techniques, Datasets, Feature Selection Methods, and Evaluation Metrics in Software Fault Prediction," Int. J. Emerg. Sci. Eng., vol. 13, no. 8, pp. 1–9, Jul. 2025, doi: 10.35940/ijese.H2643.13080725.