Received: 22 November 2024 | Revised: 7 February 2025 | Accepted: 25 February 2025 | Published online: 8 April 2025

Journal of Computational and Cognitive Engineering
2026, Vol. 5(1) 91–102
DOI: 10.47852/bonviewJCCE52024884

RESEARCH ARTICLE

Optimizing Software Fault Prediction Using Atomic Orbital Search for Sustainable Software Development

Somya Rakesh Goyal[1,*] and **Sunil Kumar**[1,*]

[1] *Department of Computer and Communication Engineering, Manipal University Jaipur, India*

Abstract: Software fault prediction (SFP) refers to the early identification of faults that can pose significant challenges to sustainable software development (SSD) goals, primarily through resource waste, increased energy consumption, and extended development cycles. SFP allows testing efforts to focus on fixing issues before they escalate, minimizing wasted time and effort. SFP models facilitate high-quality end-products with reduced environmental impact and optimal resource utilization by enabling organizations to minimize failures and align their software development efforts with sustainability objectives. Machine learning-based predictors are skyrocketing in the SFP domain. This work proposes the optimization of the AdaBoost SFP model through the selection of optimal feature subsets using the atomic orbital search (AOS) algorithm to enhance SSD. The proposed model improves the performance of the AdaBoost-based SFP, having an area under the receiver operating characteristic (ROC) curve (AUC) of 83.56% by 8.85% on average. The impact of AOS on the SFP is assessed by making a comparison with four other optimization algorithms over 10 public datasets, and empirical metrics such as accuracy and AUC are utilized. The experimental findings embark on the usefulness of the proposed SFP model as it brings about an AUC improvement of 6.8%, 2.6%, 5.02%, and 3.6% and an accuracy raise of 7.7%, 4.7%, 5.8%, and 4.2% for whale optimization algorithm, particle swarm optimization, FireFly, and sine cosine, respectively. The statistical validation of the work upholds the proposed SFP model for the improvement in software quality to achieve more sustainable practices in the software industry.

Keywords: software fault prediction (SFP), sustainable software development (SSD), atomic orbital search (AOS), feature selection (FS), area under the receiver operating characteristic (ROC) curve (AUC), sustainable development goal 12 (SDG 12)

1. Introduction

Software fault prediction (SFP) is a crucial aspect of software development that directly impacts quality, resource allocation, and overall sustainability [1]. By effectively predicting and managing defects, software teams can reduce waste, enhance efficiency, and improve the longevity of their products [2]. This focus on defect prediction aligns with the principles of sustainable software development (SSD), which seeks to minimize environmental impact and optimize resource use [3]. SFP plays an important role in sustainability in the following ways [4]: (1) Accurate fault prediction allows teams to allocate resources more effectively. By identifying potential problem areas early, teams can focus their efforts on fixing issues before they escalate, minimizing wasted time and effort. (2) Reducing defects leads to higher-quality software. High-quality products require less maintenance and fewer updates, decreasing the resource consumption associated with post-release fixes. It results in a more maintainable product. (3) Fewer defects resulting in better-quality software lead to less need for additional computing resources, which reduces energy consumption and carbon footprint.

Early fault prediction utilizing learning machines has always had a high influx of research [2]. Historical data from previous projects, including metrics such as code complexity, commit history, and test results, are used to build predictive models, and machine learning (ML) models are deployed to analyze patterns in the data and identify areas with a higher likelihood of defects [5]. Ensembles are effective in handling skewed datasets for SFP [6]. AdaBoost is one of the most prominently used ensembles among researchers in the SFP domain [7].

Software failures badly threaten the software development process and the sustainability goals [8]. Now, as organizations are increasingly prioritizing sustainability, understanding the relationship between software quality and environmental impact becomes crucial. Early fault prediction can save the following situations that may be caused by failures in the future if left undetected [9]: (1) Software failures lead to excessive resource consumption, including time, computing power, and human effort. Fixing defects often requires additional iterations, which can increase energy use and waste resources. (2) Inefficient software that fails to perform as intended may require more server resources, leading to higher energy consumption and a larger carbon footprint. This contradicts sustainability goals aimed at reducing environmental impact. (3) Frequent failures can prolong development cycles, delaying

***Corresponding authors:** Somya Rakesh Goyal, Department of Computer and Communication Engineering, Manipal University Jaipur, India. Email: somya.goyal@jaipur.manipal.edu and Sunil Kumar, Department of Computer and Communication Engineering, Manipal University Jaipur, India. Email: kumar.sunil@jaipur.manipal.edu

product releases. This not only affects productivity but can also lead to missed opportunities for more sustainable solutions, as teams may revert to older, less efficient technologies. (4) Software failures can diminish user trust and satisfaction, prompting organizations to allocate more resources to support and maintenance instead of investing in innovative, sustainable practices.

The defect datasets of the past projects are high dimensional with a few or more irrelevant features that detriment the prediction power of SFP models. The selection of optimal feature subsets enhances the performance of SFP models [10]. However, selecting the feature subset from the significant number of original features in software data requires some sort of exhaustive search, which leads to the NP-hard problem. Metaheuristic algorithms are proven to be a boon in this scenario [11], and the employment of such optimization algorithms, like Genetic Evolution [12], Firefly Search [13], and Lion Optimization [14], for the reduction of high-dimensional datasets, improves the accuracy of SFP models.

To align software quality with sustainability goals, an effective SFP mechanism is required to foster quality. SFP foresees potential software failures, allowing teams to address issues proactively. Feature selection (FS) can enhance the performance of SFP models, hence improving overall software quality [15].

Green Software Engineering and SFP intersect in that both aim to improve software quality. Reducing defects can indirectly contribute to greener software by ensuring that software runs more efficiently and requires less computational power. Software that has fewer defects may require less maintenance, which in turn can reduce resource and energy usage over time. Explicitly, Green Software Engineering emphasizes optimization of resources and energy efficiency to make software more environmentally sustainable [16], and SFP focuses on predicting and identifying defects or bugs in software early in the development process, utilizing statistical and ML models to identify parts of the software most likely to contain defects, enabling developers to prioritize testing and maintenance.

This study proposes a novel model named AB-AOS-SSD based on the AdaBoost (AB) with Atomic Orbital Search (AOS) optimization [17] for SSD. AOS offers a better balance between exploration and exploitation, adapting to dynamic optimization landscapes for improved search and convergence in comparison to other metaheuristic algorithms. Its atomic-based model helps avoid local minimum and premature convergence. By incorporating quantum principles, AOS provides a more sophisticated search mechanism.

To date, we have not identified any studies on AOS deployed in the SFP domain. This model combines the AOS algorithm for feature reduction with the AdaBoost learning algorithm that handles the class imbalance of the dataset. The proposed synergism of AOS and AdaBoost leads to the overall improvement in SFP accuracy. The following research questions (RQs) have been framed to steer the research in a guided direction:

RQ1. How efficient is the SFP model developed using AdaBoost with and without AOS FS?

RQ2. How effective is the SFP model developed using the proposed hybrid model AB-AOS-SSD in comparison to other models from the literature?

RQ3. Does statistical evidence exist in support of the effective performance of the proposed model?

The major contributions of this study are as follows:

1) A hybrid fault predictor based on AdaBoost that integrates the feature reduction mechanism using AOS has been proposed and built.
2) The hybrid model is evaluated on 10 software defect datasets obtained from NASA and Promise repositories to reflect upon the generalizability of our proposed model.
3) Multiple performance metrics – area under ROC curve (AUC) and accuracy – are computed for the validation and analysis of our classification performance of the proposed model.
4) The predictive capability of the proposed model is compared to the other four state-of-the-art hybrid algorithms present in the literature to depict that our model has produced better results across all the datasets.
5) For statistical analysis, we have used the Friedman test to prove the validity of our results.

The structure of the study is as follows: Section 2 discusses the relevant work from current research trends. Section 3 describes the materials and methods used for experimental work. Section 4 reports the results and inferences drawn from the experimental work. Section 5 summarizes the contribution of the study and concludes the work.

2. Literature Review

From the literature relevant to the SFP with metaheuristics to improve the predictions in support of SSD, some latest and relevant studies are handpicked for setting the background tone of the paper and to get into the enhancements achieved by them.

Goyal [12] used genetic evolution with a support vector machine-based SFP. Goyal [13] proposed a FireFly (FF) algorithm for FS and compared the performance of FF-based SFP against the genetic algorithm (GA) and particle swarm optimization (PSO)-based SFPs. FF-based SFP returned better results with statistical validation. Goyal and Bhatia [14] applied a Lion-based optimization algorithm with an Artificial Neural Network-based SFP model.

Rhmann [15] employed a modified PSO for SFP and ended up with promising results. Mafarja et al. [18] presented a fusion of whale optimization and random forests to predict software quality. Das et al. [19] deployed golden jackal optimization algorithms for SFP. Nasser et al. [20] developed a framework for ensemble-based SFP models using advanced FS based on the sine cosine (SC) algorithm. Harikiran et al. [21] developed ensemble models for SFP. Zivkovic et al. [22] developed boosting SFP models with the application of metaheuristics for FS. Potharlanka [23] demonstrated the necessity and positive impact of metaheuristics-based FS for ensemble-based SFP models.

Metaheuristic algorithms are prominently used for either the hyperparameter tuning of the SFP model or to obtain reduced feature subsets for building the SFP model. In either case, the objective is to improve the prediction power of ML-based SFP models to reduce the chances and hence the adverse impact of software faults on the SDLC schedule, manpower, budget, and environment. The summary of the related work is tabulated in Table 1.

3. Research Methodology

The methodology adopted for experimentation and the principles behind the AB-AOS-SSD model are deliberated with the flow graph of the working model within this section hereunder.

Experiments are carried out using MATLAB R2024 on 13th Gen Intel(R) Core (TM) i5-1345U (1.60 GHz) with 32GB RAM. The proposed model AB-AOS-SSD is the hybridization of AOS-based dimensionality reduction with AdaBoost for SSD by devising an accurate SFP model. The uniqueness of the model lies in the AOS algorithm customized for maximizing the AUC of the AdaBoost SFP model by selecting the optimal feature subset.

The working principle behind the proposed framework is the quantum theory of atoms that covers electron density configuration and energy absorption or emission.

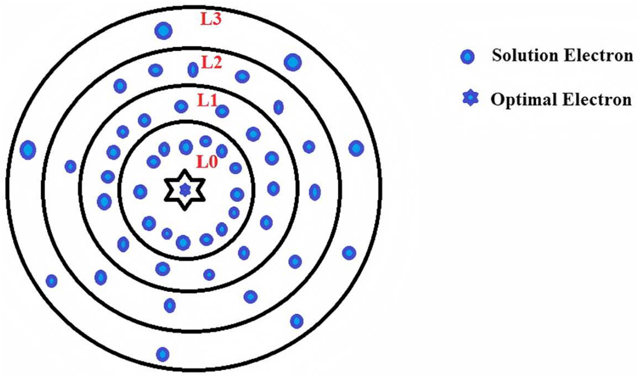
Table 1
Summary of related literature works

S. No.	Literature	Data corpus	Metrics	Metaheuristics	SFP model	Advantages	Limitations
1	Goyal [12]	NASA	AUC	Genetic Evolution (GE)	Support Vector Machine (SVM)	Improved performance of SVM classifier using GE by 16.80% (AUC)	Limited scope due to a lack in comparative analysis with other FS filters
2	Goyal [13]	NASA	AUC	Firefly Optimization (FOFS)	Artificial Neural Networks (ANN)	Improvement in ANN-based SFP by 19.6% (AUC) achieved	No comparison with other FS methods
3	Goyal and Bhatia [14]	NASA	AUC	Lion Optimization (Li-Op-FS)	Artificial Neural Networks (ANN)	Better performance achieved via FS; 19.8% AUC improvement over GE	Generalization of results over datasets lacking
4	Rhmann [15]	NASA	Precision, recall	ACO, PSO	AdaBoost	Optimized the hyperparameters and attained high values for recall and precision of 91.6% and 98.3%	No statistical evidence reported. AUC metric is missing
5	Mafarja et al. [18]	NASA	TPR, TNR, AUC	Binary whale optimization algorithm (BWOA)	Random Forest	Novel BWOA enhanced with gray wolf optimizer + Harris Hawks optimization with F-test Rank 1.81	The model is too complex to be traceable and causes high computational cost
6	Das et al. [19]	NASA	Accuracy	Golden Jackal Optimization	KNN, DT, NB	Better performance achieved via FS, showing 88.9% accuracy	Parameter value setting is complicated
7	Nasser et al. [20]	NASA	AUC, accuracy	Sine cosine algorithm (SCA)	Ensembles (RF, bagging, boosting)	Adaptive SCA with Linear Discriminant Analysis (LDA) proposed showing accuracy of 90.5% and AUC of 87.7%	Generalization of results is lacking as only three datasets are deployed; computationally complex
8	Harikiran et al. [21]	NASA	Accuracy, precision, recall	Whale optimization + FireFly	Ensemble	Feature selection improved the performance of ensemble-based SFP with 91% accuracy	No statistical validation is done
9	Zivkovic et al. [22]	PROMISE	Accuracy	Reptile Search, Shapley	XGBoost	Metaheuristic search for optimization of classifier parameters	Shapley-based local feature importance assessments
10	Potharlanka [23]	PROMISE	AUC, precision	PSO + FA + WOA	Deep-Q	Improved SFP Model with 8.5%, 4.3% raise in accuracy and AUC, respectively	Tuning the hyper-parameter for the model is tedious

The problem formulation is as follows: The search space comprises several solution candidates that are called the population, and this population represents the electrons in the surroundings of a nucleus. The electrons are assumed to exist in the form of a cloud, which is further divided into orbits. Each electron represents a candidate solution represented as X_i, *where* $i = 1, 2, \ldots, m$ (m is the total number of electrons in the cloud). An electron is represented with a binary string of length "d" *denoted as* $< x_i^j >$ *where* $j = 1, 2, \ldots, d$; "d" denotes the dimensionality (i.e., number of features) in the original defect dataset. $x_i^j \in \{0, 1\}$|"0" denotes the exclusion of the jth feature, and "1" denotes the inclusion of the jth feature in X_i.

Each electron has a state energy E_i associated with it, which is the objective function for the SFP optimization problem. E_i is the negated AUC of AdaBoost for that particular feature subset represented by the binary string of X_i. As per the atomic theory, the electron with the lowest energy has the highest probability density and is considered the best solution. The SFP problem is a maximization problem, so the AUC is negated to fit into the atomic theory. (The higher the magnitude of AUC, the lower the energy of the candidate electron.) As per the atomic theory, the electrons in the close vicinity of the nucleus have lower energy and higher probability of availability, as shown in Figure 1.

Figure 1
Electron probability density around nucleus within atom



The model with N orbits has electrons or population (say solutions) X^k, where k denotes the respective layer [24]. X^k can be represented, as shown in Equation (1).

$$X^k = \begin{bmatrix} X_1^k \\ X_2^k \\ \cdot \\ \cdot \\ \cdot \\ X_i^k \\ \cdot \\ \cdot \\ X_p^k \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^j & \dots & x_1^d \\ x_2^1 & x_2^2 & \dots & x_2^j & \dots & x_2^d \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ x_i^1 & x_i^2 & \dots & x_i^j & \dots & x_i^d \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ x_p^1 & x_p^2 & \dots & x_p^j & \dots & x_p^d \end{bmatrix} \tag{1}$$

where X_i^k is the ith electron s.t. $1 \le i \le p$ and p is the total number of electrons at the kth orbit. $1 \le k \le n$ denotes the orbits. An individual electron X_i^k represents a candidate solution that can be mathematically shown as a position vector, where x_i^j denotes the jth feature of the ith electron and $1 \le j \le d$ and d denotes the total number of dimensions (say features) in the original dataset. x_i^j can have the value of 1 or 0 for inclusion or exclusion of the respective jth feature in the ith candidate solution electron.

The objective function that is the energy levels of electrons at level-k can be modeled as a vector shown in Equation (2), where E_i^k is the energy value of ith electron at kth level, that is, negated magnitude of the AUC obtained from AdaBoost SFP model built with that electron (feature subset).

The original defect dataset is fed to the AOS algorithm for FS, where the AUC of AdaBoost model for candidate electron is the objective function. The reduced feature subset is used for the final predictions of fault-prone modules leading to SSD.

The model's uniqueness lies in the quantum atomic theory of how the energy levels affect the movement of electrons among the atomic orbits.

The movement of electrons to achieve a stable state is mapped as an optimization problem that has not yet been practiced in SSD using SFP optimization.

$$E^k = \begin{bmatrix} E_1^k \\ E_2^k \\ \cdot \\ \cdot \\ \cdot \\ E_i^k \\ \cdot \\ \cdot \\ E_p^k \end{bmatrix} = (-) \begin{bmatrix} AUC_1^k \\ AUC_2^k \\ \cdot \\ \cdot \\ \cdot \\ AUC_i^k \\ \cdot \\ \cdot \\ AUC_p^k \end{bmatrix} \tag{2}$$

The mathematical equations for computing the binding energy and binding state at kth level (or orbit) are given as Equation (3a) and (3b), respectively:

$$BE^k = \sum_{i=1}^{p} E_i^k \Big/ p \tag{3a}$$

$$BS^k = \sum_{i=1}^{p} E_i^k \Big/ p \tag{3b}$$

where $1 \le i \le p$ (population at kth level) and $1 \le k \le n$ (orbits).

Similarly, the mathematical equations for computing the binding energy and binding state of the atom are given as Equation (3c) and (3d), respectively:

$$BE = \sum_{i=1}^{m} X_i \Big/ m \tag{3c}$$

$$BS = \sum_{i=1}^{m} X_i \Big/ m \tag{3d}$$

where $\sum_{i=1}^{m} E_i \Big/ m$ (electrons in an atom, say, the search space).

The emission and absorption of energy result in an update in the position of an electron that can be modeled as Equations (4) and (5), respectively:

$$X_{i+1}^k = X_i^k + [(\alpha \times (\beta \times LE - \gamma \times BS))/k] \tag{4}$$

$$X_{i+1}^k = X_i^k + (\alpha \times (\beta \times LE - \gamma \times BS)) \tag{5}$$

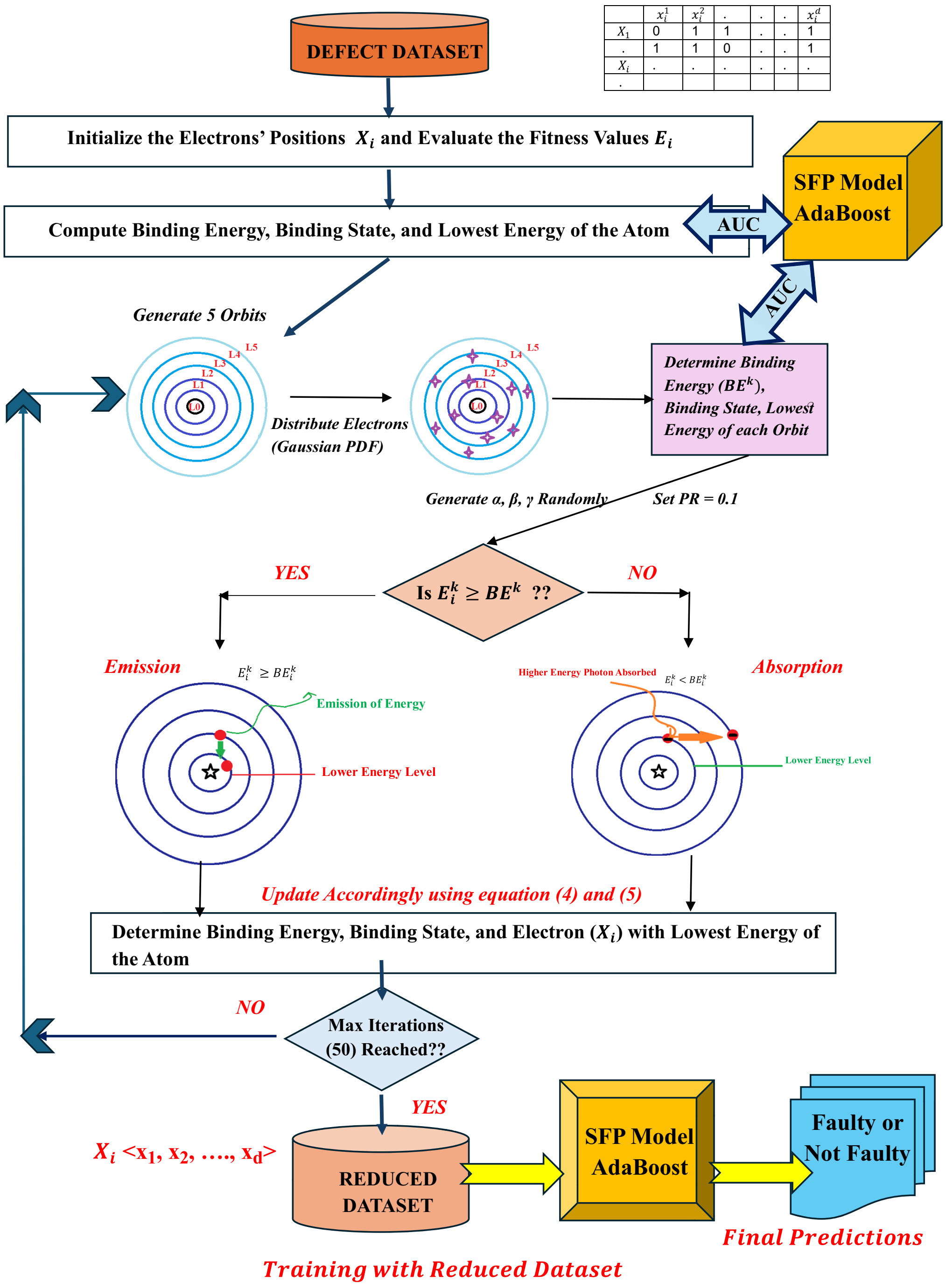
where $1 \le i \le p$ (population at kth level), $1 \le k \le n$ (orbits), and α, β, and γ are random numbers from [0,1].

3.1. Research design

The work of the proposed model AB-AOS-SDD is based on the following algorithm. The flow graph of the proposed working model is shown in Figure 2.

The NASA and Promise datasets are being considered for experimentation. These datasets are appropriate for experimentation as the literature has used these datasets for benchmarking various SFP models. NASA datasets have defect data from NASA projects, and the Promise repository comprises data from industry projects. Availability and acceptance are two major reasons for selecting these datasets, and they are large enough to carry out the experiments. Further, it allows us to compare our results with existing work carried out using the same datasets. Holdout validation with an 80–20 ratio is adopted. It serves well in this case as the available datasets are large enough. On a trial-and-error basis, the authors observed that hold-out is adequate for this experimentation without adding any additional computational overhead, as the case of cross validation, which does not add any substantial value. It is quicker and bears lesser computational cost. Furthermore, it is in direct association with SSD with reduced computational cost.

Figure 2
Flow graph of AB-AOS-SSD algorithm


DEFECT DATASET
Initialize the Electrons' Positions X_i and Evaluate the Fitness Values E_i
Compute Binding Energy, Binding State, and Lowest Energy of the Atom
AUC
SFP Model AdaBoost
AUC
Generate 5 Orbits
L0 L1 L2 L3 L4 L5
Distribute Electrons (Gaussian PDF)
Determine Binding Energy (BE^k), Binding State, Lowest Energy of each Orbit
Generate α, β, γ Randomly
Set PR = 0.1
Is $E_i^k \geq BE^k$??
YES
NO
Emission
$E_i^k \geq BE_i^k$
Emission of Energy
Lower Energy Level
Absorption
Higher Energy Photon Absorbed
$E_i^k < BE_i^k$
Lower Energy Level
Update Accordingly using equation (4) and (5)
Determine Binding Energy, Binding State, and Electron (X_i) with Lowest Energy of the Atom
Max Iterations (50) Reached??
NO
YES
X_i <x_1, x_2, …., x_d>
REDUCED DATASET
SFP Model AdaBoost
Faulty or Not Faulty
Final Predictions
Training with Reduced Dataset

	x_i^1	x_i^2	.	.	.	x_i^d
X_1	0	1	1	.	.	1
.	1	1	0	.	.	1
X_i	.	.	.	.	.	.
.						

The training set is pre-processed with AOS, and then the reduced dataset is used to build and train the AdaBoost ensemble-based SFP model. Adaboost is an ensemble learning technique that combines weak learners to form a strong classifier, making it a powerful tool for improving classification performance and yielding high accuracy. AdaBoost has an implicit ability to handle high-dimensional datasets and to focus on important features. Further, it is computationally efficient and easy to tune and resistant to overfitting [22]. Later, the testing set is used to make the final predictions about the fault-prone modules.

The use of metaheuristics in comparison to other available methods for FS (e.g., filter, wrapper, embedded methods) offers multiple benefits that are as follows: (1) facilitates the excellent handling of complex relationships among the features, unlike filter methods that are capable of learning only linear relationships; (2) allows faster processing in comparison to wrapper methods and less prone to overfitting; and (3) offers higher scalability, flexibility, and model independence in comparison to embedded methods [19].

Algorithm: AB-AOS-SDD Algorithm

Input: Original Defect Dataset; ADABoost Model; Parameter-Value Settings: Base:= C4.5; Number of estimators:=180; Learning Rate:=0.01; Grid Search; Hold-Out Validation; Ratio:=80–20; Number of Solutions:=25; Max_Iterations:=50; n:=5; m:=25; p:=5;
Output: AUC and accuracy of ADABoost optimized with AOS for SDD
Steps:

```
1.  Determine the initial position $X_i$ of the candidate solution (electrons) in cloud with $i = 1, 2, \ldots, m$;
2.  Evaluate the fitness of electrons $E_i$ by computing $-AUC_i$ of AdaBoost SFP for that respective solution;
3.  Compute binding energy (BE) and binding state (BS) of the atom;
4.  Identify the electron with the lowest energy level;
5.  while (iteration_number<Max_Iterations)
6.      Generate n orbits;
7.      Sort the electrons in ascending order;
8.      Distribute the electrons in orbits with a Gaussian PDF;
9.      for k:=1 to n
10.         Compute binding energy $BE^k$ and binding state $BS^k$ at kth orbit;
11.         Identify the electron with the lowest energy level in the kth orbit;
12.         for i:=1 to p
13.             Generate $\alpha$, $\beta$, and $\gamma$ random numbers from[0,1];
14.             Set PR:=0.1; // Photon Ratio
15.             Test if $E_i^k \geq BE^k$
16.                 update $X_{i+1}^k = X_i^k + [(\alpha \times (\beta \times LE - \gamma \times BS))/k]$;
17.             else
18.                 update $X_{i+1}^k = X_i^k + (\alpha \times (\beta \times LE - \gamma \times BS))$;
19.             end if;
20.         end for;
21.     end for;
22.     Update binding energy (BE) and binding state (BS) of the atom;
23.     Update the electron with the lowest energy level;
24. end while;
25. Assess the electron with the lowest energy level for the optimized feature subset;
26. Train the AdaBoost SFP with the reduced training set;
27. Test the AdaBoost SFP with the reduced test bench;
28. return the ROC, AUC, and accuracy for the optimized SFP model as Output;
```

Multiple metrics are recorded for performance evaluation over all 10 datasets, including SO2, ROC, AUC, and accuracy. The recorded results are further empirically compared with state-of-the-art algorithms, namely, FireFly (FF), whale optimization algorithm (WOA), particle swarm optimization(PSO), and sine cosine (SC). The selection of these algorithms for comparison with AOS is grounded in their prominence and effectiveness in the field of SFP. These algorithms represent different nature-inspired approaches, each with its own strengths. The comparison allows for a detailed evaluation of AOS's performance relative to state-of-the-art optimization algorithms, shedding light on its potential advantages for improved FS in SFP models. Parameter settings of all the algorithms used in experimentation are given in Table 2.

Table 2
Parameter settings for SFP model

Classification Algorithm AdaBoost	Base = C4.5; Number of estimators = 180; Learning Rate = 0.01; Grid Search; Hold-Out Validation; Ratio = 80–20;
Metaheuristic Algorithm (Dimension Reduction)	Number of Solutions = 25; Max. Number of Iterations = 50;
	n = 5; α, β, and γ are random numbers from [0,1]; PR = 0.1;
FireFly (FF)	thres = 0.5; alpha = 1; beta = 1; gamma = 1; theta = 0.97;
Whale Optimization Algorithm (WOA)	b = 1;
Particle Swarm Optimization(PSO)	thres = 0.5; c1 = 2; c2 = 2; w = 0.9; Vmax = (ub – lb) / 2;
Sine Cosine (SC)	thres = 0.5; alpha = 2;

4. Results and Discussions

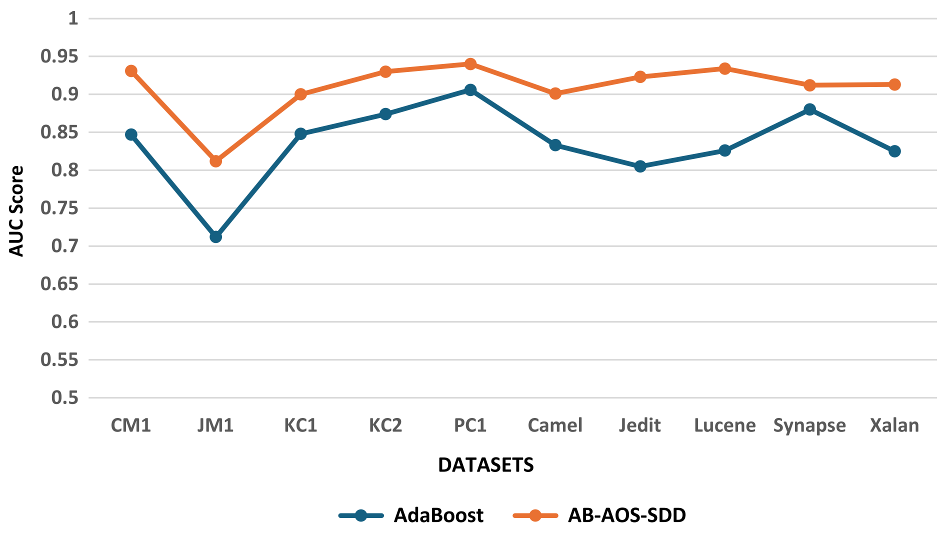
After conducting the experiments, the observations and inferences are reported under this section. The discussions address the RQs of this work.

4.1. Impact of AB-AOS-SDD on software fault prediction

Addressing RQ#1: How efficient is the SFP model developed using AdaBoost with and without AOS FS?

To measure the impact of the AOS on the performance of the AdaBoost SFP model, the experiments are repeated twice: first, AdaBoost SFP without AOS-based FS and, second, AdaBoost SFP with AOS (AB-AOS-SDD). During the experimentation, AUC score was recorded over all 10 datasets. The results are plotted in Figure 3. AdaBoost without FS is showing AUC of 84.7% for CM1, 71.2% for JM1, 84.8% for KC1, 87.4% for KC2, 83.3% for Camel, 80.5% for JEdit, 82.6% Lucene, 88.0% for Synapse, and 82.5% for

Figure 3
AUC of AdaBoost SFP with and without AOS



Xalan, whereas AB-AOS-SDD-based SFP has AUC of 93.1% for CM1, 81.2% for JM1, 90.0% for KC1, 93.0% for KC2, 94.0% for Camel, 92.3% for JEdit, 93.4% Lucene, 91.2% for Synapse, and 91.3% for Xalan.

On average, AdaBoost has 82.5% AUC, and the proposed AB-AOS-SDD has 91.3% AUC over the selected 10 datasets. It shows that the AB-AOS-SDD-based SFP model has an 8.8% better AUC score than the simple AdaBoost SFP model. It is inferred that ***the AB-AOS-SDD is more effective than the simple AdaBoost SFP model***.

4.2. Comparison of AB-AOS-SDD with models from literature

Addressing RQ#2: How effective is the SFP model developed using the proposed hybrid model AB-AOS-SSD in comparison to the other models from the literature?

To address this question, four FS techniques have been chosen for hybridization with AdaBoost Classifier, namely, WOA [18], FF [13], PSO [15], and SC [20]. The convergence of all these four models and AB-AOS-SDD is plotted in Figure 4. For FF, PSO, and

Figure 4
Convergence of AB-AOS-SDD and competing models

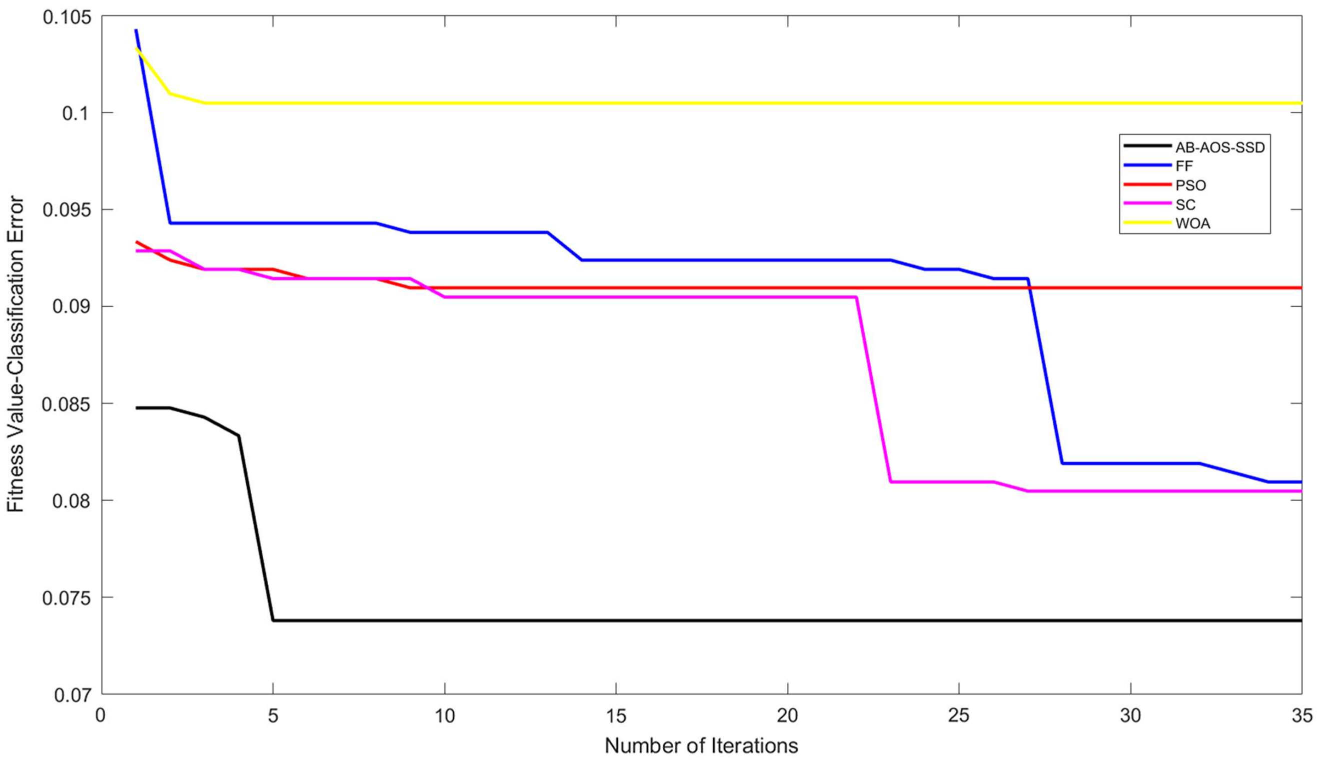
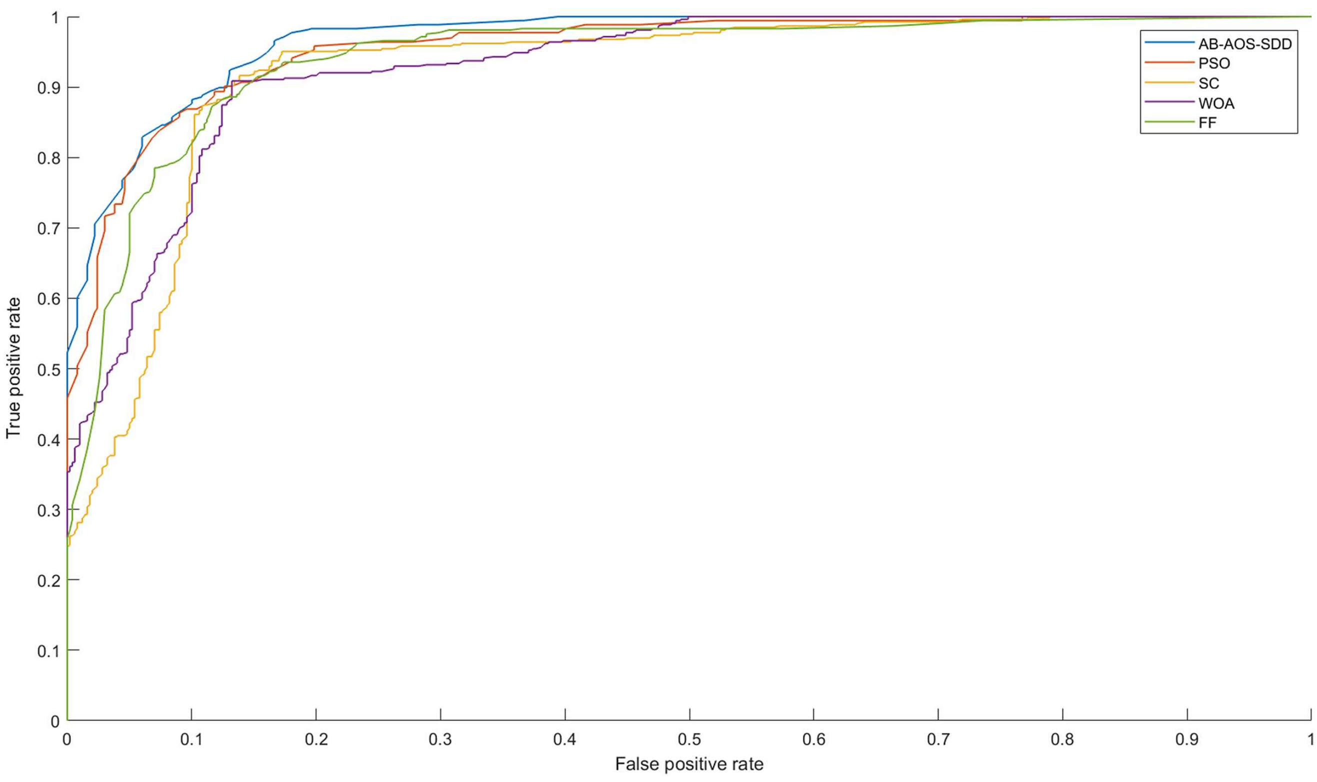


Figure 5
ROC of AB-AOS-SDD and competing models



SC, the convergence is not stable with increasing iterations. Convergence of AB-AOS-SSD is very stable, and with very few iterations, it reaches the convergence point. Only WOA reaches convergence earlier than AB-AOS-SDD, but it is not optimal. Among all the models, only AB-AOS-SDD reaches the most optimal solution with a lesser number of iterations.

From the convergence plot, it can be inferred that the proposed model is the best performer among all the competing models. Next, the ROCs are plotted for all five models in Figure 5.

The ROC plot reflects the prediction performance of the model by plotting the TPR against false positive rate (FPR). The ideal curve passes through (0,0), (1,0), and (1,1), covering a unit square area.

From Figure 5, the curve for AB-AOS-SDD is the closest one to the upper left corner of the plot (1,0). That represents a good TPR with low FPR. The comparative analysis shows better performance of AB-AOS-SDD over other competing models. Next, the AUC recorded in Table 3 and plotted in Figure 6. WOA shows 89.0% for CM1, 78.5% for JM1, 86.0% for KC1, 89.4% for KC2, 91.2% for PC1, 83.6% for Camel, 80.0% for JEdit, 79.1% for Lucene, 89.0% for Synapse, and 85.6% for Lucene.

The proposed AB-AOS-SDD has an AUC of 93.1% for CM1, 81.2% for JM1, 90.0% for KC1, 93.0% for KC2, 94.0% for PC1, 90.1% for Camel, 92.3% for JEdit, 93.4% for Lucene, 91.2% for Synapse, and 91.3% for Lucene.

In terms of AUC measure, the AB-AOS-SDD has 6.8% improvement over WOA, 2.9% improvement over PSO, 5.0% improvement over FF, and 3.6% improvement over the SC model.

Another metric is chosen for comparison, that is, accuracy. The accuracy of the predictions made by all five models is recorded in Table 4 and plotted as Figure 7.

Table 3
Performance of AB-AOS-SDD versus state-of-the-art models (in terms of AUC)

AUC	WOA	PSO	FF	SC	AB-AOS-SDD
CM1	0.89	0.91	0.88	0.88	0.93
JM1	0.79	0.80	0.77	0.75	0.81
KC1	0.86	0.87	0.89	0.85	0.90
KC2	0.89	0.89	0.87	0.89	0.93
PC1	0.91	0.92	0.90	0.92	0.94
Camel	0.84	0.89	0.87	0.85	0.90
Edit	0.80	0.89	0.86	0.89	0.92
Lucene	0.79	0.90	0.89	0.92	0.93
Synapse	0.89	0.88	0.90	0.90	0.91
Xalan	0.86	0.88	0.84	0.92	0.91
Average	0.85	0.88	0.87	0.88	0.91

Figure 6
AUC of AB-AOS-SDD and competing models

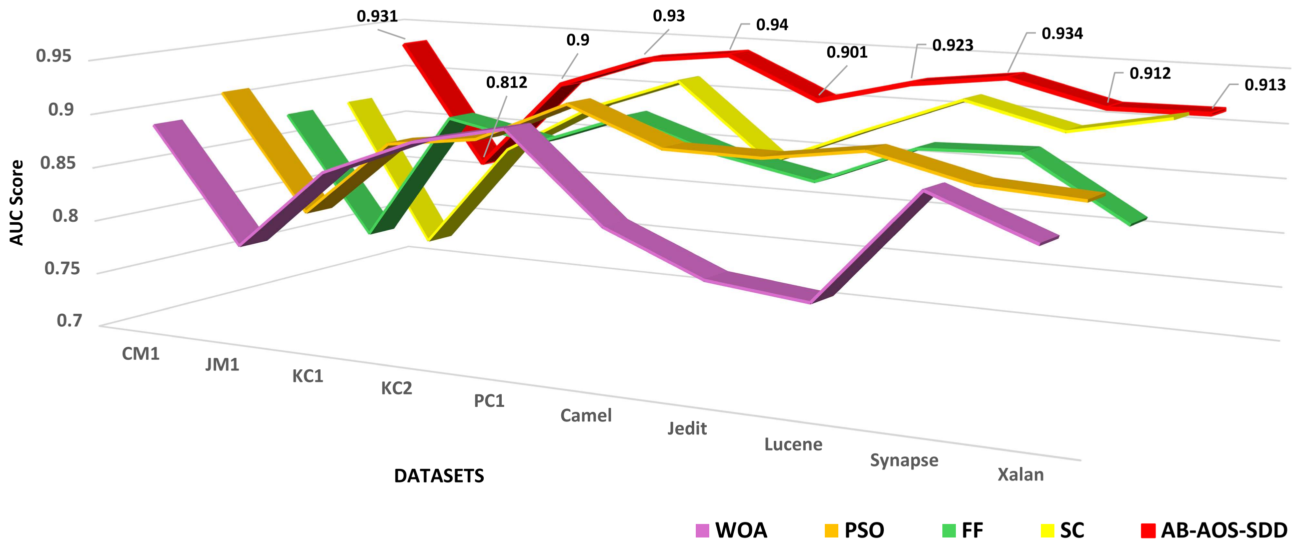


Table 4
Performance of AB-AOS-SDD versus state-of-the-art models (in terms of accuracy)

Accuracy	WOA	PSO	FF	SC	AB-AOS-SDD
CM1	0.88	0.91	0.90	0.92	0.94
JM1	0.81	0.85	0.79	0.82	0.92
KC1	0.89	0.92	0.92	0.90	0.97
KC2	0.88	0.90	0.89	0.93	0.98
PC1	0.91	0.92	0.89	0.91	0.93
Camel	0.84	0.89	0.87	0.85	0.90
Edit	0.86	0.89	0.88	0.92	0.97
Lucene	0.89	0.90	0.91	0.92	0.96
Synapse	0.91	0.92	0.91	0.93	0.97
Xalan	0.87	0.89	0.91	0.92	0.95
Average	0.87	0.90	0.89	0.91	0.95

Figure 7
Accuracy of AB-AOS-SDD and competing models

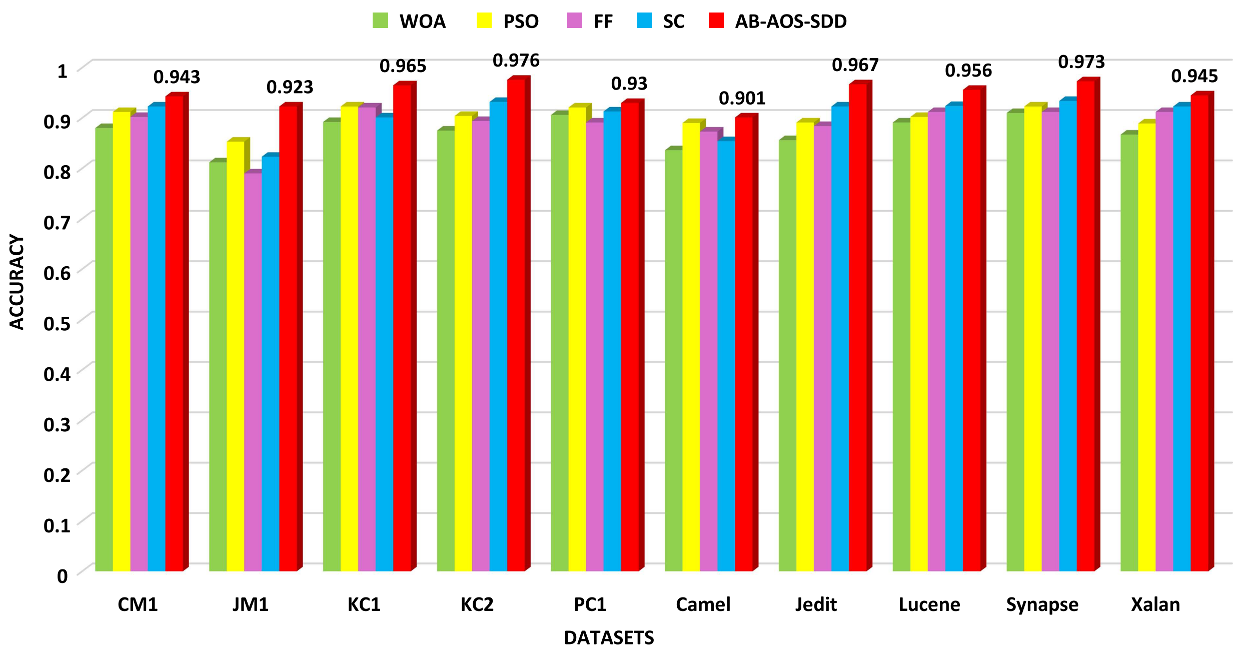
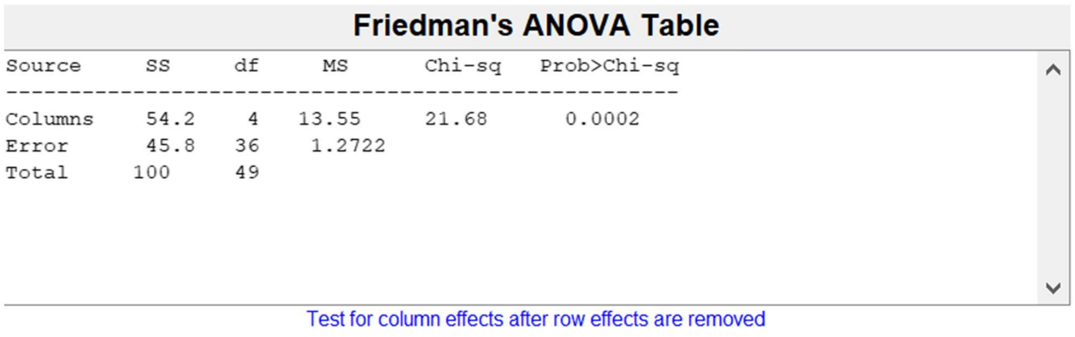


Table 5
Performance of AB-AOS-SDD versus state-of-the-art models (in terms of computation time)

Computation time (in seconds)	AB-AOS-SDD	FF	PSO	SC	WOA
CM1	0.59	8.80	1.38	1.32	1.28
JM1	6.96	102.03	21.92	7.12	12.96
KC1	4.88	20.09	2.02	2.06	2.18
KC2	1.02	9.05	1.25	1.54	1.49
PC1	1.22	10.04	1.56	2.19	1.40
Camel	1.75	8.24	2.41	2.21	2.31
Edit	0.53	8.33	1.30	1.02	1.15
Lucene	0.50	9.12	1.20	1.18	1.19
Synapse	0.33	8.01	1.47	1.01	1.06
Xalan	1.70	8.01	1.47	1.01	1.06
Average	1.95	19.17	3.60	2.07	2.61

Figure 8
***P*-statistic for Friedman test**



Friedman's ANOVA Table

```
Source      SS     df     MS       Chi-sq   Prob>Chi-sq
---------------------------------------------------------
Columns     54.2    4   13.55      21.68      0.0002
Error       45.8   36    1.2722
Total      100     49
```

Test for column effects after row effects are removed

On average, the performance of WOA, PSO, FF, SC, and AB-AOS-SDD in terms of accuracy is 87.0%, 90.0%, 88.9%, 90.5%, and 94.5 %, respectively.

It is evident that the AB-AOS-SDD has a 7.7% improvement over WOA, 4.7% improvement over PSO, 5.8% improvement over FF, and 4.2% improvement over the SC model.

The comparison of the proposed method with the state-of-the-art models is made over the computational time taken by them to make the predictions. The results are recorded in Table 5. That clearly shows that the proposed model is computationally efficient in comparison to competing models. The measurements of time are taken in unit of second.

From Table 5, it is observed that the proposed model AB-AOS-SDD is better than competing models in terms of computational complexity, as it takes less time than the competing models. Further, it scales gracefully with large-sized data as it does well with JM1 (i.e., one of the largest datasets). It performs equally well with the real-life project datasets (Camel, Edit, Lucene, Synapse, and Xalan).

From experimental results, it is inferred that ***the proposed AB-AOS-SDD model performs better than the state-of-the-art models***. Next, we seek the statistical evidence for the results we brought here.

4.3. Statistical evidence of results

Addressing RQ#3: Does statistical evidence exist in support of the effective performance of the proposed model?

It is necessary to validate the experimental results statistically. The authors run the statistical tests over the recorded performance of WOA, PSO, FF, SC, and AB-AOS-SDD in terms of AUC. Our data are matched groups (multiple) and nonparametric. Hence, the Friedman test is considered for statistical evidence [24–26] as it is a nonparametric method that allows for the analysis of repeated measures or matched data without the stringent assumptions of parametric tests. Hence, the Friedman test is suitable for statistical validation of results here. Figure 8 shows the result of the test at a 95% confidence level.

The value for *p*-statistic obtained is 0.0002, which shows the statistical evidence for the better performance of AB-AOS-SDD over the WOA, FF, SC, and PSO models. It is inferred that ***statistical evidence in support of the effective performance of the proposed model does exist***.

5. Conclusions

Enhancing the accuracy of the defect prediction models is a vital strategy for promoting SSD. By leveraging data-driven approaches, ML, and optimization techniques, software teams can not only improve software quality but also contribute to more sustainable practices in the tech industry. This approach helps in minimizing waste, optimizing resource use, and reducing the environmental footprint of software development activities. Contributions to the proposed work are as follows:

1) Early identification of defects using the proposed AB-AOS-SDD model leads to significant cost savings, as fixing defects early in the development process is typically less expensive than addressing them after deployment.
2) By focusing on high-risk areas, development teams can work more efficiently, reducing the time spent on less critical issues.

3) High-quality software that requires fewer resources for maintenance supports long-term sustainability goals by reducing the overall environmental impact of software development.
4) The proposed model shows an AUC of 90.9% and improves the performance of the AdaBoost-based SFP, having an AUC of 83.56% by 8.85% on average.
5) The proposed model is empirically compared with four state-of-the-art models, namely, WOA [18], FF [13], PSO [15], and SC [20], and determined that the proposed is the best performer among all the competing models.
6) The comparison is made over 10 datasets from NASA and Promise Repository, which ensures the generalizability of the model.
7) The proposed SFP model brings about an AUC improvement of 6.8%, 2.6%, 5.02%, and 3.6% and an accuracy raise of 7.7%, 4.7%, 5.8%, and 4.2% for WOA, PSO, FF, and SC, respectively.
8) The results are statistically validated by conducting the Friedman Test. The *p*-statistic is less than 0.5, which validates the experimental results at the confidence level of 95%.

The authors propose to extend the work for real-time industry data for determining the defect density by counting the number of defects in the currently developing software at various industries. Furthermore, the utilization of evolving metaheuristics algorithms will be considered for extending the study, along with a focus on the prediction of energy bugs dedicatedly in direct support of Green Software Engineering.

Funding Support

This work is sponsored by the Manipal University Jaipur, Jaipur-303007, Rajasthan, India.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in GitHub at https://github.com/feiwww/PROMISE-backup/tree/master/bug-data. The data that support the findings of this study are openly available in the GitHub at https://github.com/ApoorvaKrisna/NASA-promise-dataset-repository.

Author Contribution Statement

Somya Rakesh Goyal: Conceptualization, Methodology, Validation, Formal analysis, Writing – original draft, Writing – review & editing, Visualization, Funding acquisition, Project administration. **Sunil Kumar:** Conceptualization, Writing – review & editing, Visualization.

References

[1] Li, Z., Niu, J., & Jing, X. Y. (2024). Software defect prediction: Future directions and challenges. *Automated Software Engineering*, *31*(1), 19. https://doi.org/10.1007/s10515-024-00424-1

[2] Zhao, Y., Damevski, K., & Chen, H. (2023). A systematic survey of just-in-time software defect prediction. *ACM Computing Surveys*, *55*(10), 201. https://doi.org/10.1145/3567550

[3] Leong, J., May Yee, K., Baitsegi, O., Palanisamy, L., & Ramasamy, R. K. (2023). Hybrid project management between traditional software development lifecycle and agile based product development for future sustainability. *Sustainability*, *15*(2), 1121. https://doi.org/10.3390/su15021121

[4] Çınar, Z. M., Abdussalam Nuhu, A., Zeeshan, Q., Korhan, O., Asmael, M., & Safaei, B. (2020). Machine learning in predictive maintenance towards sustainable smart manufacturing in Industry 4.0. *Sustainability*, *12*(19), 8211. https://doi.org/10.3390/su12198211

[5] Khalid, A., Badshah, G., Ayub, N., Shiraz, M., & Ghouse, M. (2023). Software defect prediction analysis using machine learning techniques. *Sustainability*, *15*(6), 5517. https://doi.org/10.3390/su15065517

[6] Chen, L. Q., Wang, C., & Song, S. L. (2022). Software defect prediction based on nested-stacking and heterogeneous feature selection. *Complex & Intelligent Systems*, *8*(4), 3333–3348. https://doi.org/10.1007/s40747-022-00676-y

[7] Yuan, Z., Chen, X., Cui, Z., & Mu, Y. (2020). ALTRA: Cross-project software defect prediction via active learning and tradaboost. *IEEE Access*, *8*, 30037–30049. https://doi.org/10.1109/ACCESS.2020.2972644

[8] Rathor, K., Kaur, J., Nayak, U. A., Kaliappan, S., Maranan, R., & Kalpana, V. (2023). Technological evaluation and software bug training using genetic algorithm and time convolution neural network (GA-TCN). In *2023 Second International Conference on Augmented Intelligence and Sustainable Systems*, 7–12. https://doi.org/10.1109/ICAISS58487.2023.10250760

[9] Kumar, K., Pande, S. V., Kumar, T. C. A., Saini, P., Chaturvedi, A., Reddy, P. C. S., & Shah, K. B. (2023). Intelligent controller design and fault prediction using machine learning model. *International Transactions on Electrical Energy Systems*, *2023*(1), 1056387. https://doi.org/10.1155/2023/1056387

[10] Saeed, M. S. (2023). Role of feature selection in cross project software defect prediction-A review. *International Journal of Computations, Information and Manufacturing*, *3*(2), 37–56. https://doi.org/10.54489/ijcim.v3i2.277

[11] Kaur, K., & Kumar, A. (2023). MCDM-EFS: A novel ensemble feature selection method for software defect prediction using multi-criteria decision making. *Intelligent Decision Technologies*, *17*(4), 1283–1296. https://doi.org/10.3233/IDT-230251

[12] Goyal, S. (2022). Genetic evolution-based feature selection for software defect prediction using SVMs. *Journal of Circuits, Systems and Computers*, *31*(11), 2250161. https://doi.org/10.1142/S0218126622501614

[13] Goyal, S. (2022). FOFS: Firefly optimization for feature selection to predict fault-prone software modules. In *Data Engineering for Smart Systems: Proceedings of SSIC 2021*, 479–487. https://doi.org/10.1007/978-981-16-2641-8_46

[14] Goyal, S., & Bhatia, P. K. (2021). Software fault prediction using lion optimization algorithm. *International Journal of Information Technology*, *13*(6), 2185–2190. https://doi.org/10.1007/s41870-021-00804-w

[15] Rhmann, W. (2020). Cross project defect prediction using hybrid search based algorithms. *International Journal of Information Technology*, *12*(2), 531–538. https://doi.org/10.1007/s41870-018-0244-7

[16] Valmohammadi, C., & Hejri, F. M. (2023). Designing a conceptual green process model in software development: A mixed method approach. *International Journal of Information*

Management Data Insights, *3*(2), 100204. https://doi.org/10.1016/j.jjimei.2023.100204

[17] Azizi, M. (2021). Atomic orbital search: A novel metaheuristic algorithm. *Applied Mathematical Modelling*, *93*, 657–683. https://doi.org/10.1016/j.apm.2020.12.021

[18] Mafarja, M., Thaher, T., Al-Betar, M. A., Too, J., Awadallah, M. A., Abu Doush, I., & Turabieh, H. (2023). Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning. *Applied Intelligence*, *53*(15), 18715–18757. https://doi.org/10.1007/s10489-022-04427-x

[19] Das, H., Prajapati, S., Gourisaria, M. K., Pattanayak, R. M., Alameen, A., & Kolhar, M. (2023). Feature selection using golden jackal optimization for software fault prediction. *Mathematics*, *11*(11), 2438. https://doi.org/10.3390/math11112438

[20] Nasser, A. B., Ghanem, W. A. H. M., Saad, A.-M. H. Y., Abdul-Qawy, A. S. H., Ghaleb, S. A. A., Alduais, N. A. M., ..., & Ghetas, M. (2024). Depth linear discrimination-oriented feature selection method based on adaptive sine cosine algorithm for software defect prediction. *Expert Systems with Applications*, *253*, 124266. https://doi.org/10.1016/j.eswa.2024.124266

[21] Harikiran, J., Chandana, B. S., Srinivasarao, B., Raviteja, B., & Reddy, T. S. (2023). Software defect prediction based ensemble approach. *Computer Systems Science & Engineering*, *45*(3), 2313–2331. https://doi.org/10.32604/csse.2023.029689

[22] Zivkovic, T., Nikolic, B., Simic, V., Pamucar, D., & Bacanin, N. (2023). Software defects prediction by metaheuristics tuned extreme gradient boosting and analysis based on shapley additive explanations. *Applied Soft Computing*, *146*, 110659. https://doi.org/10.1016/j.asoc.2023.110659

[23] Potharlanka, J. L. (2024). Feature importance feedback with Deep Q process in ensemble-based metaheuristic feature selection algorithms. *Scientific Reports*, *14*(1), 2923. https://doi.org/10.1038/s41598-024-53141-w

[24] Lehmann, E. L., & Romano, J. P. (2005). *Testing statistical hypothesis* (3rd ed.). USA: Springer. https://doi.org/10.1007/0-387-27605-X

[25] Ross, S. M. (2017). *Introductory statistics* (4th ed.). UK: Academic Press.

[26] Pham, H. (2023). *Springer handbook of engineering statistics* (2nd ed.). UK: Springer. https://doi.org/10.1007/978-1-4471-7503-2

How to Cite: Rakesh Goyal, S., & Kumar, S. (2026). Optimizing Software Fault Prediction Using Atomic Orbital Search for Sustainable Software Development. *Journal of Computational and Cognitive Engineering*, *5*(1), 91–102. https://doi.org/10.47852/bonviewJCCE52024884