Received 3 July 2024, accepted 19 July 2024, date of publication 29 July 2024, date of current version 8 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3435333

RESEARCH ARTICLE

Enhancing Software Fault Prediction Through Feature Selection With Spider Wasp Optimization Algorithm

HIMANSU DAS[1], SWARNAVA DAS[1], MAHENDRA KUMAR GOURISARIA[1], (Member, IEEE), SURBHI BHATIA KHAN[2,3], (Senior Member, IEEE), AHLAM ALMUSHARRAF[4], ABDULLAH I. ALHARBI[5], AND T. R. MAHESH[6], (Senior Member, IEEE)

[1]School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, Odisha 751024, India
[2]Department of Data Science, School of Science, Engineering and Environment, University of Salford, M5 4WT Manchester, U.K.
[3]Department of Electrical and Computer Engineering, Lebanese American University, Byblos 13-5053, Lebanon
[4]Department of Management, College of Business Administration, Princess Nourah Bint Abdulrahman University, P. O. Box 84428, Riyadh 11671, Saudi Arabia
[5]Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[6]Department of Computer Science and Engineering, Faculty of Engineering and Technology, JAIN (Deemed-to-be University), Bengaluru 562112, India

Corresponding author: Himansu Das (das.himansu2007@gmail.com)

This research is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R432), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

ABSTRACT Software fault prediction (SFP) is a critical focus in software engineering, aiming to enhance productivity and minimize costs by detecting faults early. Feature selection (FS) is pivotal in SFP, enabling the identification of pertinent features for fault prognosis. Existing Feature Selection methods face challenges such as high computational complexity and poor generalization. This paper introduces Feature Selection using Spider Wasp Optimization (FSSWO), a novel FS approach employing the Spider Wasp Optimization (SWO) algorithm, specifically designed for SFP. FSSWO selects optimal feature subsets inspired by spider wasps' behavior. The proposed FSSWO approach is compared with several existing feature selection algorithms, namely FS using Genetic Algorithm (FSGA), FS using Particle Swarm Optimization (FSPSO), FS using Differential Evolution (FSDE), and FS using Ant Colony Optimization (FSACO). Using eleven benchmark datasets, the performance of the proposed FSSWO technique has been assessed and contrasted with its equivalent. The results of the proposed FSSWO approach provide comparable and even superior results to the existing algorithms. The significance of the results has been statistically validated using Friedman and Holm tests. The statistical result of the proposed FSSWO approach reveals that the performance of proposed FSSWO models is improved which leads to better quality software at reduced costs.

INDEX TERMS Spider Wasp optimization algorithm, feature selection, wrapper method, software fault prediction.

I. INTRODUCTION

SFP [1] is an essential area of research in software engineering. Software systems are ubiquitous and play a crucial role in modern society. They are used in various domains, such as healthcare [2], finance [3], transportation [4], and communication [5]. However, software systems are prone to faults, which can lead to significant consequences, such as system crashes, data loss, security breaches, and financial losses. The early identification of software system flaws can lower overall costs and boost software quality. The method of determining and forecasting the likelihood of software system flaws is known as SFP. Various SFP techniques [6], including statistical, hybrid and machine learning, can be used in current times, with each technique having its own strengths and limitations.

Static analysis and dynamic analysis are the two divisions that may be made for SFP. Dynamic analysis is the process of examining the program while it is being executed, whereas static analysis is the process of studying the

The associate editor coordinating the review of this manuscript and approving it for publication was Donato Impedovo.

software code without actually executing it. Static analysis techniques include code inspection, code review, and code analysis, while dynamic analysis techniques include testing, debugging, and profiling. SFP is a difficult task given the complexity and heterogeneity of software systems. Software systems are often composed of multiple components and layers, and the interactions between these components and layers can be intricate. Furthermore, software systems can exhibit different behavior under different circumstances, making it difficult to capture their overall behavior accurately. Presently, there are a lot of different SFP techniques, including statistical methods [7], machine learning algorithms [8], and hybrid approaches [9]. The advantages and disadvantages of each technique dictate when and where these techniques can be used for a multitude of test cases [10].

FS [11] serves as an essential step in SFP as it helps to determine the important characteristics that influence fault prediction. FS techniques can help reduce the dimensionality of the data, which can then improve the accuracy and effectiveness of the prediction models. Several FS techniques [12] have been proposed in the literature, such as filter-based [13], wrapper-based [14], and embedded-based [15] methods. However, these methods [11], [12] suffer from various limitations, such as high computational complexity, overfitting, and poor generalization ability. FS techniques in machine learning and their applications in various domains are also a major aspect of today's field of knowledge. The comparative evaluation [16] of these techniques on various datasets provides important information on when to use which technique and on what type of test case for increased accuracy and effectiveness in the performance of the model.

Recently, several optimization algorithms have been proposed for FS in SFP. The optimal or nearly optimal solution to a particular problem can be found using optimization algorithms, which are heuristic search approaches. Global and local search are two categories into which optimization algorithms can be divided. Global search algorithms like simulated annealing [17], particle swarm optimization (PSO) [18], genetic algorithms (GA) [19], and ant colony optimization (ACO) [20], find the global optimal by searching the entire solution space. Local search algorithms, such as hill climbing [21] and the N-Queen algorithm [22], search the local neighborhood of the current solution and can find a local optimum. All the aforesaid optimization algorithms require several hyperparameters to be tuned and sometimes trapped in local optima. To address this issue, the SWO algorithm is considered to select the ideal number of the most important features to identify the fault in the software.

The SWO [23] algorithm is a recently developed optimization technique that was motivated by spider wasp activity. The SWO algorithm has been successful at resolving several optimization issues, including FS. The SWO algorithm is a global search algorithm that simulates the foraging behavior of spider wasps. Spider wasps are known for their ability to search for and capture spiders, which are their primary food source. A comprehensive survey [24] of the SWO algorithm, which reviews the current status and future directions of this metaheuristic algorithm, is used as a reference to influence the proposed methodology. The article covers the origins, principles, and variations of SWO, as well as its applications and performance compared to other metaheuristics.

The orthodox operation used to perform FS boils down to being an NP-hard [25] task. This method searches the solution space in its entirety and thus performs an exhaustive search. This method for searching leads to a significantly high computation time for FS, which increases exponentially with an increase in the feature set. This motivates to reduction of the dataset dimensions to achieve a boost in accuracy due to FS, which consecutively carries forward a reduction in the quantity of time needed to compute the factors. Many of the previously stated FS techniques using WOA, GWO, and BOA heavily rely on adjusting hyperparameters, which can be a time-consuming and expensive computational activity, to achieve high performance. Furthermore, improper tuning of these hyper-parameters could cause the algorithm to perform poorly or possibly fall into a local optimum trap. However, the FS using SWO can perform better than the existing ones and is less reliant on hyperparameters. It is designed to efficiently search the feature space without the need for intensive hyperparameter adjustment.

The purpose of the suggested FSSWO approach is to acquire an optimal subset of features, which would elevate the classification model's accuracy by determining the aforementioned subset of optimal features. In the proposed FSSWO approach, all columns except the target column are converted into their correlated representation in binary (0s and 1s), which is used to determine the features that should be taken into consideration (1) and the features that should not be taken into consideration (0) while performing classification. The proposed FSSWO approach is then carried out as explained in the paper to compute the optimal number of features that should be selected to obtain the maximum possible accuracy and effectiveness. The results are then to be analyzed through graphs and statistical analysis. The comparison of the proposed FSSWO approach to other approaches to draw derivatives on the metrics of accuracy and efficiency. It would lead to describing the comparative function r wasps use a combination of visual and olfactory cues to locate spiders. It can adjust their search strategy of the proposed FSSWO approach.

The key contributions of the research article are given below

- In this article, a novel FS approach called FSSWO has been proposed for SFP using the SWO algorithm.
- The proposed FSSWO approach is compared with other existing FS algorithms, namely FSGA, FSDE, FSACO, and FSPSO, and evaluated on eleven benchmark SFP datasets.
- The features of the datasets along with the experimental results are analyzed to draw derivatives from the acquired results.

- Statistical analysis (Friedman Test and Holm Test) has also been performed to verify the significance of the difference in the results between the aforementioned approaches and the proposed FSSWO approach.

The remainder of this paper follows the following scheme of organization. A survey of related literature is included in Section II. The problem statement is presented in Section III. Section IV proposed the methodology for FS. Section V discusses the results and their analysis. Section VI contains statistical analysis including the Friedman test and Holm's procedure. Section VII provides a conclusion and the paper's future focus.

II. RELATED WORKS

SWO is an optimization process influenced by nature that has derived a basis from the hunting behavior of spider wasps. The SWO algorithm imitates the hunting characteristics of spider wasps, which search for their prey by sensing the vibration of the prey and moving towards the source of the vibration. The algorithm consists of three phases: searching, chasing, and attacking. In the searching phase, the algorithm explores the search space to find potential solutions. In the chasing phase, the algorithm moves towards the best solution found in the searching phase. Finally, in the attacking phase, the algorithm intensifies the search around the best solution to refine the solution.

SFP [1] is a critical and significant activity during the aboriginal stage of software development life cycle for increase software quality and reducing the maintenance cost. The early detection of defect can lead to quicker problem resolve and the transferral of rectifiable software. SFP intensify the software quality by evaluating faults using previous data. A comprehensive evaluation of SFP methods based on machine learning which highlighted the effectiveness of machine learning algorithms and identified the key factors affecting the prediction performance has also been taken into consideration as observed previously [26]. It also provides a roadmap for upcoming avenues for this field of research. The basics of FS and minute improvements through many years improve the current performance of models through a generative effect [1], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38]. Software testing also goes hand in hand with the proposed methodology [39].

There have been extensively researched articles written previously which present an insight into optimal FS [40], [41], [42], [43], [44]. For example, a comprehensive review of FS techniques and their applications in machine learning has been carried out before which discussed various types of FS methods, their advantages, disadvantages, and provided guidelines for selecting appropriate techniques for different types of datasets [45]. A survey of FS techniques and their applications in cancer research has been performed previously [46]. There are a multitude of different methods of FS based on similar concepts to the above-mentioned papers [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57].

Researchers have suggested and investigated numerous wrapper-based FS techniques that rely on optimisation algorithms over the past ten years. A. Fatima et al. [96] developed an FS technique based on GA. The detection of android malware utilised their FS model. To detect faults in power distribution networks, Cho et al. [97] presented an FS model depending on PSO for feature selection and SVM parameter optimisation. Dixit et al. [98] suggested another FS model employing DE for the classification of text and image data. To enhance the performance of the classifiers, they combined DE with NB and SVM classifiers in their model. ACO was used in an FS strategy.

A previously written article which proposes a hybrid model combining neuro-fuzzy systems and feature reduction techniques for classification tasks is also relevant to the current methodology. The proposed model used fuzzy rules to capture the non-linear relationships between input features and output classes, while feature reduction methods are employed to enhance the efficiency of the classification process [58]. The model was tested on various datasets, and the results demonstrated its effectiveness in improving classification accuracy and reducing computational costs. Research has been done previously on a neuro-fuzzy model for biomedical data analysis that combines the strengths of neural networks and fuzzy logic. The proposed model employs methods for reducing features to enhance the performance and efficiency of the analysis. The article uses trials on two biomedical datasets to show that the method is successful [59]. There has also been research on a linguistic neuro-fuzzy model for disease classification. The proposed model uses fuzzy logic and linguistic variables to address the ambiguity and accuracy issues with medical data. Real-world medical datasets are used to assess the model's performance, demonstrating its effectiveness in accurately classifying diseases [60]. Articles that have been previously written also provide light to the use of different methods for Classification problems related to the methods mentioned above [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74]. The aforesaid FS algorithms could be applied in different diversified area [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85] of research to solve real life problems.

III. PROBLEM STATEMENT

This section represents the formulation of the problem that has been used in this paper. FS refers to a technique for picking the best subset of columns (used as features) from the features that are present in the dataset in order to increase the performance of classification models. The process is used to select the elements that have a significance in the process of making the decision for accurate results. The most important group of characteristics is selected by removing superfluous and pointless features, in order to lower the cost of computation for the problem. This is an NP-hard problem, which indicates that it will not be able to be solved in polynomial time. Obtaining the most effective feature subset in order to boost the process' overall performance of classification is the end goal for FS. The process of FS is composed of the four phases : (i) Generating a subset of features; (ii) evaluating and comparing fitness levels by using the selected subsets of characteristics; (iii) Verifying that the requirements for termination have been satisfied and doing the processes (i) and (ii) if they have not been met; validating the results by using the optimal characteristic subset. 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).

$$\begin{aligned} & f(x) = min\_err(d) \ and \ d \subset D \\ & \text{Minimize f(x), Subject to Condition,} \\ & \text{x} = |\text{D}| \text{ and x} \geq 0 \end{aligned} \tag{1}$$

IV. BASIC CONCEPTS OF SWO ALGORITHM

The SWO Algorithm (SWOA) is a metaheuristic optimization algorithm inspired by the hunting behavior of spider wasps. Spider wasps are known for their unique hunting strategy, where they search for spiders, paralyze them with a venomous sting, and then carry them back to their nests for their larvae to feed on. SWO explores the space of possible feature subsets by representing each solution (a potential subset of features) as a spider wasp's position in the search space. The algorithm iteratively explores different combinations of features, evaluating their performance based on a fitness function. The SWOA algorithm mimics this strategy by using two types of agents: spiders and wasps. Spiders in the SWOA algorithm represent possible answers to the optimization issue. Each spider is represented by a binary string of length N, where N is the number of decision variables in the problem. The value of each bit in the binary string represents whether the corresponding decision variable is selected or not. For example, if the problem has three decision variables, the binary string "101" represents a solution where the first and third variables are selected, and the second variable is not selected. Wasps in the SWOA algorithm represent exploratory agents that help spiders find better solutions. Each wasp is also represented by a binary string of length N, but its bits have a different meaning than those of spiders. A wasp's bits represent the degree of similarity between the wasp and each spider, based on the Hamming distance between their binary strings. The Hamming distance between two binary strings is the number of bits that differ between them. For example, if a spider has the binary string "101" and a wasp has the binary string "111", the Hamming distance between them is 1. The SWOA algorithm starts by initializing a population of spiders and wasps randomly. The fitness of each spider is evaluated by the implementation of a fitness function that determines how effectively the spider's binary string embodies a suitable response to the optimization problem. The fitness of each wasp is calculated based on its similarity to the spiders, using the Hamming distance. The spiders and wasps are then sorted by their fitness, and the top half of each group is selected for further breeding. New spiders are created by combining the first half of the top spiders with the second half of the top wasps, using a crossover operation. The crossover operation selects a random point in the binary string and swaps the bits on either side of the point between the spider and the wasp. For example, if the spider has the binary string "101" and the wasp has the binary string "111", and the crossover point is at position 2, the new spider will have the binary string "111". This operation helps the spiders explore new areas of the search space by incorporating the best features of the wasps. New wasps are created by mutating the second half of the top wasps, using a bit-flip operation. The bit-flip operation selects a random bit in the binary string and flips its value. This operation helps the wasps explore new areas of the search space by creating small perturbations to their binary strings. The spiders and wasps are then combined into a new population, and each agent's fitness is assessed once more. The process of selection, breeding, and mutation is recurred for a predetermined number of times. or until a satisfactory solution is found. The SWOA algorithm has demonstrated to be successful in resolving a variety of optimization issues, such as feature selection in machine learning. Mathematically, the SWOA algorithm fundamental parameters such as (i) spider representation; (ii) wasp representation; (iii) hamming distance is shown in Equation (2), (3), and (4) respectively.

$$S_i = \left(s_{i,1}, s_{i,2}, \ldots, s_{i,N}\right) \tag{2}$$

$$W_j = \left(w_{j,1}, w_{j,2}, \ldots, w_{j,N}\right) \tag{3}$$

The Hamming distance between two binary strings S_i and W_j can be defined in Equation (4).

$$H_{i,j} = \sum_{k=1}^{N} \delta\left(s_{i,k}, w_{j,k}\right) \tag{4}$$

Here, $\delta(a, b)$ is the Kronecker delta function, which is 1 if $a = b$ and 0 otherwise.

The fitness function used in the SWOA algorithm for FS can vary depending on the specific problem being solved. In this case, Gaussian Naive Bayes (NB) algorithm, Decision Tree Classifier (DTC), K-Nearest Neighbors Classifier (KNN) and Linear Discriminant Analysis (LDA) are used as fitness functions. These are probabilistic algorithms of machine learning which are commonly used for classification tasks. The fitness function for the SWOA algorithm using these classifiers (CLF) can be defined in Equation (5).

$$Fitness\left(S_i\right) = accuracy\left(CLF\left(X_{S_i}, y\right)\right) \tag{5}$$

Here, X_{S_i} is the feature subset represented by spider $S_{i,}y$ is the target variable, and $accuracy\left(CLF\left(X_{S_i}, y\right)\right)$ is the accuracy of the CLF algorithm trained on X_{S_i} and Y after performing train test split operation with testing size = 0.2 (20%).

The crossover operation used in the SWOA algorithm can be represented by Equation (6).

$$Crossover\left(S_i, W_j\right) = \left(s_{i,1}, s_{i,2}, \ldots, s_{i,k-1}, w_{j,k}, w_{j,k+1}, \ldots, w_{j,N}\right) \quad (6)$$

Here, k is a random crossover point selected uniformly at random from 1 to $N-1$. The bit-flip mutation operation used in the SWOA algorithm can be represented in Equation (7).

$$Mutation\left(W_j\right) = \left(w_{j,1}, w_{j,2}, \ldots, w_{j,k-1}, 1-w_{j,k}, w_{j,k+1}, \ldots, w_{j,N}\right) \quad (7)$$

Here, k is a random bit position selected uniformly at random from 1 to N.

In summary, the SWOA algorithm is a metaheuristic optimization algorithm inspired by the hunting behavior of spider wasps. It uses two types of agents, spiders and wasps, to explore the search space efficiently and maintain diversity in the population. The algorithm can be applied to various optimization problems, including FS in machine learning. The fitness function used in the SWOA algorithm varies based on the particular issue being treated, and the crossover and mutation operations help the spiders and wasps explore new areas of the search space.

V. PROPOSED FEATURE SELECTION APPROACH USING SWO (FSSWO)

In this section, the proposed FSSWO method is broken down into several steps such as (i) initialization, (ii) fitness evaluation, (iii) Spider Movement (Crossover), and (iv) Wasp Movement (Mutation).

A. INITIALIZATION OF PARAMETERS

In this step, the initial population of spiders and wasps are randomly generated within the search space. The size of the population is defined as (Number of spiders/Wasps(i/j)) * (Number of features(N)). The spider population and wasp population is denoted in Equation (8), and (9) respectively.

$$S = \begin{bmatrix} S_{1,1} & \ldots & S_{1,N/2} & \ldots & S_{1,N} \\ \vdots & \ddots & \vdots & / & \vdots \\ S_{i/2,1} & \ldots & S_{i/2,N/2} & \ldots & S_{i/2,N} \\ \vdots & / & \vdots & \ddots & \vdots \\ S_{i,1} & \ldots & S_{i,N/2} & \ldots & S_{i,N} \end{bmatrix}_{i\chi N} \quad (8)$$

$$W = \begin{bmatrix} W_{1,1} & \ldots & W_{1,N/2} & \ldots & W_{1,N} \\ \vdots & \ddots & \vdots & / & \vdots \\ W_{j/2,1} & \ldots & W_{j/2,N/2} & \ldots & W_{j/2,N} \\ \vdots & / & \vdots & \ddots & \vdots \\ W_{j,1} & \ldots & W_{j,N/2} & \ldots & W_{j,N} \end{bmatrix}_{j\chi N} \quad (9)$$

B. FITNESS EVALUATION

The fitness of each subset of features (each row of spiders and wasps) is calculated by passing each subset into the fitness function, and the results are stored in two new arrays (spider_fitness and wasp_fitness). Two new populations are then created (spider_sorted and wasp_sorted) which store the subsets of features in ascending order of error (fitness value) for the spider and wasp populations respectively. The top half of these populations are stored in newly defined spider_top and wasp_top populations respectively for the current iteration. The fitness functions used in this method implement NB, DTC, KNN and LDA as four different fitness functions that are passed into the main SWO algorithm function for the results. The calculation of fitness (F_i) for each selected subset of features(i) is performed by calculating the summation of differences between the original and predicted results $\left(Error_i^g\right)$ and dividing by the total number of instances (N_i) as shown in Equations (10) and (11).

$$Error_i^g = \left[y_i'^g \neq y_i^g\right] \quad (10)$$

$$F_i = \frac{\sum_{i=1}^{N_i} Error_i^g}{N_i} \quad (11)$$

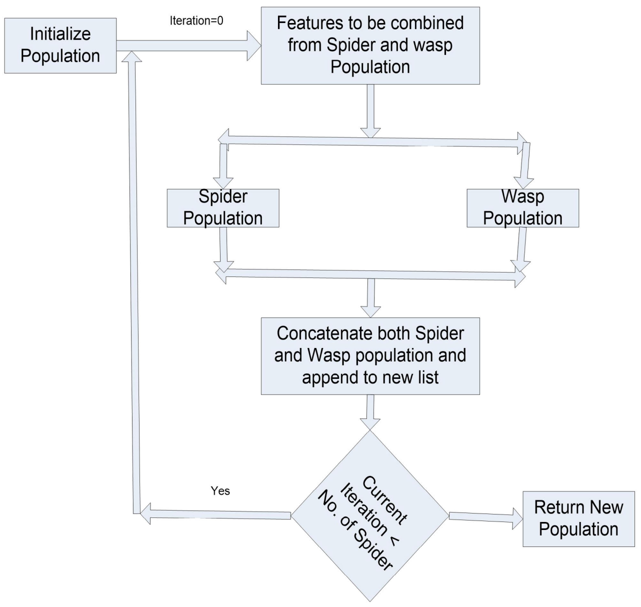
Initialize Population
Iteration=0
Features to be combined from Spider and wasp Population
Spider Population
Wasp Population
Concatenate both Spider and Wasp population and append to new list
Current Iteration < No. of Spider
Yes
Return New Population

FIGURE 1. Process of crossover for spiders.

C. SPIDER MOVEMENT (CROSSOVER)

A new blank population is created (spider_new) and feature subsets are assigned by performing crossover as shown in Fig. 1.

The Crossover operation is done by concatenating certain features from the spider_top(S_t) and wasp_top(W_t) population which are selected as depicted in Equation(12) and Equation(13).

$$Featj_j^S = S_t\left[Rem\left(j, \frac{N_s}{2}\right)\right]\left[:\frac{N_{feat}}{2}\right] \quad (12)$$

$$Featj_j^W = W_t\left[Rem\left(j, \frac{N_w}{2}\right)\right]\left[\frac{N_{feat}}{2}:\right] \quad (13)$$

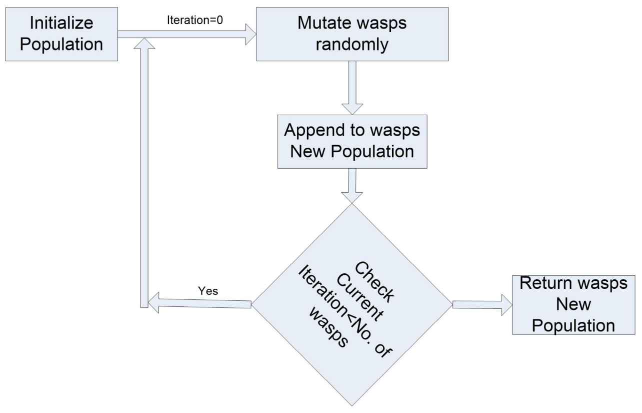
Initialize Population
Iteration=0
Mutate wasps randomly
Append to wasps New Population
Check Current Iteration<No. of wasps
Yes
Return wasps New Population

FIGURE 2. Process of mutation of wasps.

Here, $Feat_j^S$ and $Feat_j^W$ denote the features selected from S_t and W_t for iteration number j respectively. $Rem(a, b)$ calculates the remainder when dividing a by b.N_s and N_w represent number of spiders and number of wasps respectively and N_{feat} represents the total number of features available. The features from the spiders and wasps are then concatenated to retrieve the original number of features and the newly formed subset of feature for each iteration is appended to a new population (S_{new}).

D. WASP MOVEMENT (MUTATION)

A new blank population is created (wasps_new) and feature subsets are assigned by performing mutation as shown in Fig. 2.

The Mutation operation is performed by randomly changing the selection of certain features in the wasp_top population which gives rise to change in the features being selected for fitness calculation and thus changing the feature subset while keeping high performing subsets as a base. The newly formed feature subsets are then stored in a new population (W_{new}) as depicted in Equation(14).

$$Mut_{ji}^W = W_t\left[Rem\left(j, \frac{N_w}{2}\right)\right][i] \times [Rand(0, 1)] \tag{14}$$

Here, Mut_{ji}^W is the mutated value for the feature with index j for iteration number $\dot{i}$ and $Rand(a, \dot{b})$ returns an integer within the bounds of a and b.

Then the initial spider and wasp populations are concatenated with the newly formed populations (spiders_new and wasps_new). Then the fitness of each of the feature subsets are calculated and compared to the global_best fitness for all iterations. Whenever a local fitness value exceeds a global fitness value, the global best fitness and feature set are updated to the current better values and are then appended to the error curve before moving on to the next iteration. Up until the maximum number of iterations is achieved, this entire procedure continues, and the final variables that are returned are global_best_features, global_best_fitness and curve (which contains the global_best_fitness for successive

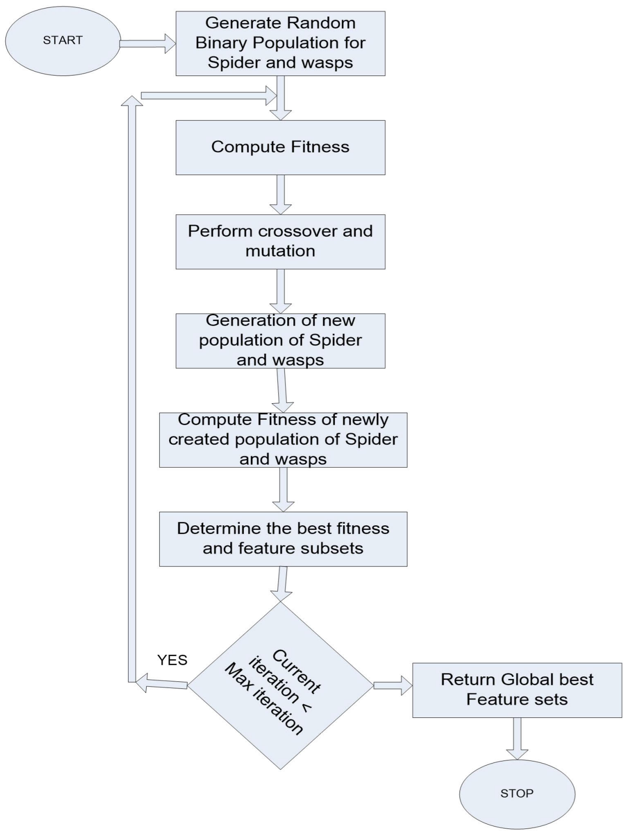
START
Generate Random Binary Population for Spider and wasps
Compute Fitness
Perform crossover and mutation
Generation of new population of Spider and wasps
Compute Fitness of newly created population of Spider and wasps
Determine the best fitness and feature subsets
Current iteration < Max iteration
YES
Return Global best Feature sets
STOP

FIGURE 3. Representation FSSWO for feature selection.

iterations until current iteration(i) reaches maximum number of iterations). The complete process is shown through flowcharts as depicted in Fig 3 (Spider Wasp Optimization Algorithm use) and Fig 4 (Complete process of FS used). The algorithm for the implementation of FSSWO is provided in Algorithm 1.

VI. 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

The research has been performed using eleven datasets of software defects that were open-source and obtained from the PROMISE repository [86], which is run by NASA. This repository provides standardized datasets that are commonly used by software engineering researchers to benchmark and compare different techniques for identifying and preventing software faults. By analyzing code attributes in these datasets, researchers develop models that can predict future faults by identifying patterns and characteristics commonly associated with software faults. This approach to software fault prediction has led to improved techniques for identifying and preventing software faults, which ultimately enhances the general effectiveness and dependability of software systems.

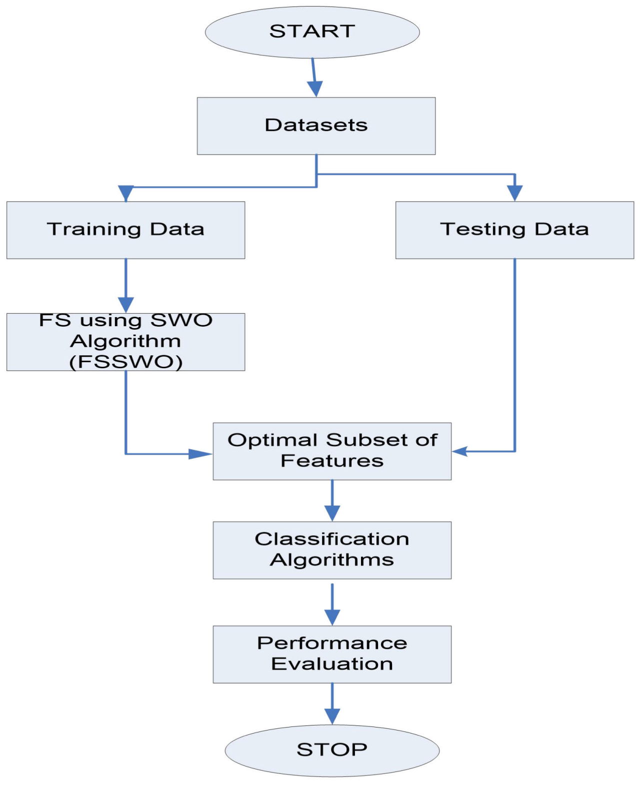
START
Datasets
Training Data
Testing Data
FS using SWO Algorithm (FSSWO)
Optimal Subset of Features
Classification Algorithms
Performance Evaluation
STOP

FIGURE 4. Complete feature selection process.

TABLE 1. Datasets and number of instances and features.

Sl. No.	Datasets	No. of instances	No. of features
1	CM1	327	37
2	KC1	1183	22
3	KC3	194	40
4	MC1	1988	39
5	MC2	125	40
6	MW1	253	38
7	PC1	705	38
8	PC2	745	38
9	PC3	1077	38
10	PC4	1287	38
11	PC5	1711	39

The datasets used in the experiments have specific attributes that are described below in Table 1.

The datasets that have been used contain static code attributes of Java software systems which provide important information to dictate if the software defects present in a system can be attributed to singular features or a combination of features based on a binary feature that represents if the current software contains a defect or not, which in turn, provides a basis for analysis of the corresponding features that may or may not have caused the defect to occur. For example, in the datasets, some columns present are as described in table 1 above. These columns are : (i) loc: How many lines of code there are in the source file. This column represents the total number of lines that contain code, comments, and whitespace, (ii) v(g): The McCabe complexity of the code, i.e., the number of decision points plus one. Based on the number of decision points in the control flow graph, this column represents the complexity of the code. (iii) ev(g): The McCabe complexity of the code, weighted by the control flow graph's edge count. This column is similar to v(g) but considers the number of edges in the control flow graph. (iv) iv(g): The McCabe complexity of the code, weighted by the control flow graph's number of independent pathways. This column is similar to v(g) and ev(g) but considers the number of independent paths in the control flow graph. (v) n: a This column represents the total number of statements that are executable in the source code, including statements within loops and conditions. (vi) v: The Halstead volume of the code, i.e., the total number of operations and operands. This column is a measure of the size of the code based on the quantity of distinct operators and operands. (vii) l: The Halstead program

Algorithm 1 Feature Selection Using SWO

```
1.  Initialize n_features, n_iterations, n_spiders and n_wasps
2.  Computation of training and testing data
3.  Generate binary populations for spiders and wasps randomly
4.  for (i =1 to n_iterations) do
5.  Compute the fitness for each subset of features present in the
      spider population(S_i) and wasp population(W_i)
6.  Sort spiders and wasps by fitness and store in spider_sorted and
      wasp_sorted respectively
7.  Select top half of sorted populations and store in spider_top and
      wasp_top respectively
8.    for (j =1 to n_spiders) do
9.        Perform crossover for spiders by extracting features from
          spider_top and wasp_top populations
          using () and () for n_spiders
10.       Append each subset to a new population (spider_new)
11    end
12    for (j =1 to n_wasps) do
13.       Perform mutation of wasps by assigning random binary
          values to the features of wasp_top population
          using () for n_wasps
14.       Append each feature subset to new population
          (wasp_new)
15    end
16.   Concatenate spider_new to spiders population and
        wasp_new to wasps population
17.   Sort both spiders and wasps population in increasing order
        of fitness and select the top n_spiders and n_wasps
        feature subsets to initialize the spider and wasp
        populations for the next iteration
18.   Store feature set with best fitness value for output as local
best
19.   Check if local best is better than global best and update
        the values as necessary
20.    i = i+1
21  end
22.  Use the global best selected subset of features
       for classification
```

length of the code, i.e., the total number of operations and operands. This column is similar to v but includes duplicates. (viii) d: The Halstead program difficulty of the code, i.e., the ratio of the number of unique operators and operands to the total number. Based on the proportion of unique operators and operands to the overall number, this column indicates how challenging the code is to comprehend. (ix) i: The Halstead intelligence of the code, i.e., the proportion of distinctive operators to distinctive operands. This column is a measure of the intelligence or expressiveness of the code based on the ratio of unique operators to unique operands. (x) e: The Halstead effort of the code, i.e., the product of program length and program difficulty. This column is a measure of the effort required to develop and maintain the code based on program length and program difficulty.

Here, the target column is taken as Defective as it provides us with information if the software under analysis contains a defect or not. The other columns are then treated as features on which the FS model is run, which provides us with information on the amount of contribution that each feature has in influencing the software having or not having a defect present, thus providing a basis on which features can be excluded from the analysis which provide little to no significant effect on the performance of the model and increase the accuracy of the model leading to better predictions for classification.

B. EXPERIMENTAL SETUP

In this experiment, the simulation environment used is Jupyter Notebook with Python version (3.9.6). along with details of the hardware in the system as follows; a processor with an Intel i5-10300H Central Processing Unit, with a pulse generation of frequency 2.50GHz from the clock and 16GB 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 20 and 200, respectively. The termination criterion for every experiment is set to 200.

C. EXPERIMENTAL ANALYSIS

In the ongoing section, we compare the error curves of each of the algorithms with respect to each of the four classifiers taken in order to calculate fitness. The error curves for each dataset have been taken into another jupyter notebook for comparison and the ‘‘mathplotlib.pyplot’’ library has been used to compare the error curves for all of the algorithms for each classifier and successive datasets respectively. This gives a general idea about how each of the algorithm’s function at different points of dimensions and iterations and provides a deeper understanding of how each algorithm reacts to different situations. The time complexity of feature selection using spider wasp optimization (FSSWO) depends on a number of factors, including the number of features, the number of iterations, and the parameters of the FSSWO algorithm. However, in general, the time complexity of FSSWO is O(n*t), where n is the number of features and t is the number of iterations. The graphs depicted for convergence curves of five distinct FS algorithms for various classifiers in Fig. 5-6 and the table 2 depict the performance of each algorithm.

As seen from the graphs presented above which display the performance of the different models for the different datasets used in the experiment, the proposed methodology provides comparable or better results from the other FS methods used for classification with minimal effect on performance due to the dimensionality of the dataset, be it the number of features or the number of instances, whereas, the other models may

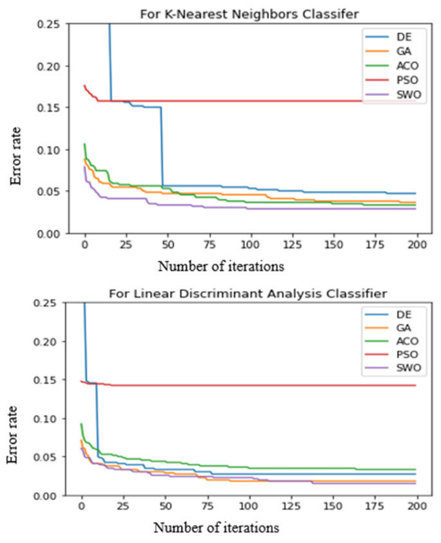

FIGURE 5. **Fitness error curve for dataset (CM1) for the five distinct FS approaches.**

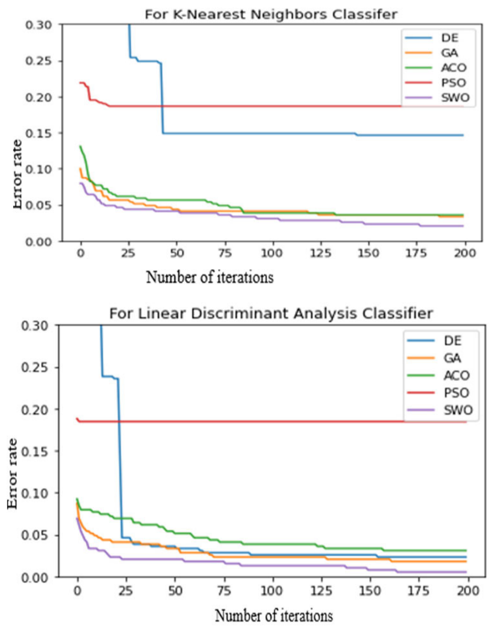

FIGURE 6. **Fitness error curve for dataset (KC3) for five distinct FS approaches.**

TABLE 2. Comparison of accuracy of the FSSWO algorithm with other algorithms and their average selected features.

S.No.	Datasets	Classifiers	Accuracy (%)					
			Without FS	FSGA	FSDE	FSACO	FSPSO	FSSWO
1	CM1	NB	78.7878	84.8484	81.8181	87.8787	80.303	**87.8787**
		DTC	74.2424	86.3636	84.8484	83.3333	74.2424	**86.3636**
		KNN	80.0133	87.8787	80.303	95.4545	83.3333	**90.909**
		LDA	83.3333	90.909	81.8181	84.8484	87.8787	**92.4242**
		Avg. Selected Features	37	17.8	22.02	19	15.22	**19.325**
2	KC1	NB	70.886	75.5274	70.886	73.8396	74.1835	**74.2616**
		DTC	63.2911	68.3544	68.7763	69.6202	64.5569	**69.6202**
		KNN	67.5105	67.9324	76.7932	73.4177	69.6202	**70.0421**
		LDA	69.6202	76.7932	79.7468	75.1054	70.886	**80.5907**
		Avg. Selected Features	22	9.57	13.37	10.42	17.85	**11.6**
3	KC3	NB	74.3589	87.1794	79.4871	82.0512	76.923	**87.1794**
		DTC	69.2307	79.4871	71.7948	74.3589	74.3589	**82.0512**
		KNN	71.7948	82.0512	82.0512	89.7435	79.4871	**84.6153**
		LDA	74.3589	92.3076	87.1794	84.6153	82.0512	**89.7435**
		Avg. Selected Features	40	20.02	22.4	7.85	20.12	**19.97**
4	MC1	NB	73.1155	95.9798	95.4773	89.6984	88.4422	**96.9849**
		DTC	80.6532	97.7386	95.7286	97.7386	95.4773	**98.9949**
		KNN	95.9798	96.7336	96.2311	97.4874	96.7336	**97.4874**
		LDA	96.7336	97.2361	96.7336	95.2261	98.2412	**98.4924**
		Avg. Selected Features	39	17.85	23.6	5.22	17.97	**18.3**
5	MC2	NB	64	72	68	80	64	**80**
		DTC	56	74	68	68	60	**72**
		KNN	60	68	64	72	64	**72**
		LDA	68	80	76	84	72	**88**
		Avg. Selected Features	40	19.4	23.07	19.95	15.35	**21.05**
6	MW1	NB	76.4705	84.3137	90.196	82.3529	88.2352	**86.2745**
		DTC	78.4313	86.2745	84.3137	80.3921	86.2745	**88.2352**
		KNN	80.3921	86.2745	90.196	86.2745	90.196	**90.196**
		LDA	82.3529	88.2352	92.1568	84.3137	92.1568	**96.0784**
		Avg. Selected Features	38	19.12	22.77	0.95	17.52	**19.25**
7	PC1	NB	82.2695	91.4893	87.234	82.9787	92.9078	**91.4893**
		DTC	82.9787	90.0709	86.5248	87.234	86.5248	**87.234**
		KNN	90.0709	92.1985	92.9078	91.4893	90.7801	**93.617**
		LDA	90.7801	93.617	92.1985	94.3262	93.617	**94.3262**
		Avg. Selected Features	39	18.27	22.35	18.52	14.4	**18.05**
8	PC2	NB	74.4966	93.2885	94.6308	93.9597	91.9463	**95.302**
		DTC	91.9463	97.3154	96.6442	96.6442	96.6442	**97.3154**
		KNN	93.2885	97.9865	98.6577	95.302	95.302	**97.9865**
		LDA	93.9597	95.302	95.302	97.9865	96.6442	**99.3288**
		Avg. Selected Features	38	18.02	21.15	4.5	14.7	**19.4**

TABLE 2. *(Continued.)* **Comparison of accuracy of the FSSWO algorithm with other algorithms and their average selected features.**

9	PC3	NB	78.2407	83.7962	75.9259	79.1666	64.3518	**81.9444**
		DTC	79.1666	81.4814	76.8518	78.2407	81.4814	**81.4814**
		KNN	85.1851	81.9444	89.8148	85.1851	84.2592	**86.1111**
		LDA	83.7962	85.6481	84.2592	83.3333	83.7962	**86.1111**
		Avg. Selected Features	38	17.37	22.22	19	17.52	**17.67**
10	PC4	NB	72.4806	89.1472	84.4961	81.0077	75.9689	**81.7829**
		DTC	81.7829	87.9844	86.0465	85.2713	84.4961	**84.1085**
		KNN	84.1085	85.2713	84.4961	84.4961	86.0465	**84.8837**
		LDA	80.2325	84.1085	83.7209	88.7596	87.5968	**91.8604**
		Avg. Selected Features	38	18.37	23.85	7.6	20.32	**19.27**
11	PC5	NB	65.5976	74.0524	75.2186	70.5539	74.0524	**75.8017**
		DTC	67.0553	67.0553	69.6793	70.5539	67.0553	**65.3061**
		KNN	69.9708	67.3469	73.4693	71.4285	69.9708	**69.9708**
		LDA	72.5947	72.8862	70.2623	73.7609	72.5947	**74.6355**
		Avg. Selected Features	39	19.05	24.8	18.45	20.85	**20.1**

differ in a large range for performance depending on the dimensionality of the provided dataset. The results that can be seen in Table 2 indicate that the proposed methodology of FS using the Spider Wasp Optimization (SWO) Algorithm generally produces consistently good results that rivals the performance of the other algorithms used for comparison. It can also be seen from the graphs Fig. 5-6 and table 2 provided above that the proposed methodology of FS works better with LDA classifier and KNN classifier than the NB classifier and DTC by a slight margin consistently through most of the datasets.

Since the datasets are very varied in their number of instances and features, this provides a solid understanding that the proposed methodology works in a number of situations and provides consistent and comparable results even with classifiers with lower results. Table 3 outlines the various hyperparameters used for feature selection in five different algorithms: GA, DE, ACO, PSO and SWO. The hyperparameters include population size, epoch, mutation rate, crossover rate, weight, C1, C2, scaling factor, number of spiders, number of wasps, α, ρ, and β. Population size specifies the number of solutions generated in each iteration of the algorithm, which in this case is 20 for all five algorithms. Epoch defines the number of iterations that the algorithm runs, which is 200 for all five algorithms. Mutation rate controls the probability of a gene in an individual being mutated, with GA using a rate of 0.8 while PSO, DE and ACO do not use mutation. Crossover rate is specific to GA and DE and determines the probability of two individuals exchanging genetic information to produce new offspring, with a rate of 0.5 for GA and 0.7 for DE respectively. Weight is specific to PSO and is used to calculate the individual's velocity during optimization. C1 and C2 are unique parameters used in PSO

TABLE 3. **Hyperparameters used in the experiment.**

Parameters	GA	PSO	ACO	DE	SWOA
Population Size	20	20	20	20	20
Epoch	200	200	200	200	200
Mutation Rate	0.1	-	-	-	-
Crossover Rate	0.5	-	-	0.7	-
Weight	-	0.5	-	-	-
C1	-	1	-	-	-
C2	-	1	-	-	-
Scaling Factor	-	-	-	0.5	-
No. of spiders					20
No. of wasps	-	-	-	-	20
α	-	-	1	-	-
β	-	-	0.1	-	-
ρ	-	-	0.5	-	-

to maintain equilibrium between a particle's present location and its established optimum position, with PSO using a value of 1 for both C1 and C2. Scaling factor is specific to DE and controls the amount by which the difference between two individuals is scaled to generate a new individual, with DE using a factor of 0.5. Number of spiders and number of ants is specific to SWO. Finally, α, ρ, and β are parameters used in ACO to control the probability of selecting a particular path, with values of 1, 0.1, and 0.5, respectively.

VII. STATISTICAL ANALYSIS

The Statistical Analysis section of this article on SFP using SWO for FS is a critical component of the research [87], [88]. The main design of this section is to judge and interpolate

TABLE 4. Friedman ranking of algorithms.

Sl. Nos.	Datasets	Classifiers	Accuracy in percentage					
			Without FS	FSGA	FSDE	FSACO	FSPSO	FSSWO
1	CM1	NB	78.7878(5)	84.8484(2)	81.8181(3)	87.8787(1)	80.3030(4)	**87.8787(1)**
		DTC	74.2424(4)	86.3636(1)	84.8484(2)	83.3333(3)	74.2424(4)	**86.3636(1)**
		KNN	80.0133(5)	87.8787(3)	80.3030(5)	95.4545(1)	83.3333(4)	**90.9090(2)**
		LDA	81.8181(6)	90.9090(2)	83.3333(5)	84.8484(4)	87.8787(3)	**92.4242(1)**
		Avg. Rank	5	2	3.75	2.25	3.75	**1.25**
2	KC1	NB	70.8860(5)	75.5274(1)	70.8860(5)	73.8396(4)	74.1835(3)	**74.2616(2)**
		DTC	63.2911(5)	68.3544(3)	68.7763(2)	69.6202(1)	64.5569(4)	**69.6202(1)**
		KNN	67.5105(6)	67.9324(5)	76.7932(1)	73.4177(2)	69.6202(4)	**70.0421(3)**
		LDA	69.6202(6)	76.7932(3)	79.7468(2)	75.1054(4)	70.8860(5)	**80.5907(1)**
		Avg. Rank	5.5	3	2.5	2.75	4	**1.75**
3	KC3	NB	74.3589(5)	87.1794(1)	79.4871(3)	82.0512(2)	76.9230(4)	**87.1794(1)**
		DTC	69.2307(5)	79.4871(2)	71.7948(4)	74.3589(3)	74.3589(3)	**82.0512(1)**
		KNN	71.7948(5)	82.0512(3)	82.0512(3)	89.7435(1)	79.4871(4)	**84.6153(2)**
		LDA	74.3589(6)	92.3076(1)	87.1794(3)	84.6153(4)	82.0512(5)	**89.7435(2)**
		Avg. Rank	5.25	1.75	3.25	2.5	4	**1.5**
4	MC1	NB	73.1155(6)	95.9798(2)	95.4773(3)	89.6984(4)	88.4422(5)	**96.9849(1)**
		DTC	80.6532(5)	97.7386(2)	95.7286(3)	97.7386(2)	95.4773(4)	**98.9949(1)**
		KNN	95.9798(4)	96.7336(2)	96.2311(3)	97.4874(1)	96.7336(2)	**97.4874(1)**
		LDA	96.7336(4)	97.2361(3)	96.7336(4)	95.2261(5)	98.2412(2)	**98.4924(1)**
		Avg. Rank	4.75	2.25	3.25	3	3.25	**1**
5	MC2	NB	64.0000(5)	72.0000(2)	68.0000(3)	80.0000(1)	64.0000(4)	**80.0000(1)**
		DTC	56.0000(5)	74.0000(1)	68.0000(3)	68.0000(3)	60.0000(4)	**72.0000(2)**
		KNN	60.0000(4)	68.0000(2)	64.0000(3)	72.0000(1)	64.0000(3)	**72.0000(1)**
		LDA	68.0000(6)	80.0000(3)	76.0000(4)	84.0000(2)	72.0000(5)	**88.0000(1)**
		Avg. Rank	5	2	3.25	1.75	4	**1.25**
6	MW1	NB	76.4705(6)	84.3137(4)	90.1960(1)	82.3529(5)	88.2352(2)	**86.2745(3)**
		DTC	78.4313(5)	86.2745(2)	84.3137(3)	80.3921(4)	86.2745(2)	**88.2352(1)**
		KNN	80.3921(3)	86.2745(2)	90.1960(1)	86.2745(2)	90.1960(1)	**90.1960(1)**
		LDA	82.3529(5)	88.2352(3)	92.1568(2)	84.3137(4)	92.1568(2)	**96.0784(1)**
		Avg. Rank	4.75	2.75	1.75	3.75	1.75	**1.5**
7	PC1	NB	82.2695(5)	91.4893(2)	87.2340(3)	82.9787(4)	92.9078(1)	**91.4893(2)**
		DTC	82.9787(4)	90.0709(1)	86.5248(3)	87.2340(2)	86.5248(3)	**87.2340(2)**
		KNN	90.0709(6)	92.1985(3)	92.9078(2)	91.4893(4)	90.7801(5)	**93.6170(1)**
		LDA	90.7801(4)	93.6170(2)	92.1985(3)	94.3262(1)	93.6170(2)	**94.3262(1)**
		Avg. Rank	4.75	2	2.75	2.75	2.75	**1.5**
8	PC2	NB	74.4966(6)	93.2885(4)	94.6308(2)	93.9597(3)	91.9463(5)	**95.3020(1)**
		DTC	91.9463(3)	97.3154(1)	96.6442(2)	96.6442(2)	96.6442(2)	**97.3154(1)**
		KNN	93.2885(4)	97.9865(2)	98.6577(1)	95.3020(3)	95.3020(3)	**97.9865(2)**
		LDA	93.9597(5)	95.3020(4)	95.3020(4)	97.9865(2)	96.6442(3)	**99.3288(1)**
		Avg. Rank	4.5	2.75	2.25	2.5	3.25	**1.25**
9	PC3	NB	64.3518(6)	83.7962(1)	75.9259(5)	79.1666(3)	78.2407(4)	**81.9444(2)**

TABLE 4. *(Continued.)* **Friedman ranking of algorithms.**

		DTC	79.1666(2)	81.4814(1)	76.8518(3)	78.2407(4)	81.4814(1)	**81.4814(1)**
		KNN	85.1851(3)	81.9444(5)	89.8148(1)	85.1851(3)	84.2592(4)	**86.1111(2)**
		LDA	83.7962(4)	85.6481(2)	84.2592(3)	83.3333(5)	83.7962(4)	**86.1111(1)**
		Avg. Rank	3.75	2.25	3	3.75	3.25	**1.5**
10	PC4	NB	72.4806(6)	89.1472(1)	84.4961(2)	81.0077(4)	75.9689(5)	**81.7829(3)**
		DTC	81.7829(6)	87.9844(1)	86.0465(2)	85.2713(3)	84.4961(4)	**84.1085(5)**
		KNN	84.1085(5)	85.2713(2)	84.4961(4)	84.4961(4)	86.0465(1)	**84.8837(3)**
		LDA	80.2325(6)	84.1085(4)	83.7209(5)	88.7596(2)	87.5968(3)	**91.8604(1)**
		Avg. Rank	5.75	1.5	3.25	3.25	3.25	**3**
11	PC5	NB	65.5976(5)	74.0524(3)	75.2186(2)	70.5539(4)	74.0524(3)	**75.8017(1)**
		DTC	67.0553(3)	65.3061(4)	69.6793(2)	70.5539(1)	67.0553(3)	**67.0553(3)**
		KNN	69.9708(3)	67.3469(4)	73.4693(1)	71.4285(2)	69.9708(3)	**69.9708(3)**
		LDA	72.5947(4)	72.8862(3)	70.2623(5)	73.7609(2)	72.5947(4)	**74.6355(1)**
		Avg. Rank	3.75	3.5	2.5	2.25	3.25	**2**

information from the results obtained from the experimental study conducted in the previous section. There are also certain factors that must be kept in mind in order to procure accurate results from statistical analysis [89].

This section discusses the application of the Friedman Test [90], a nonparametric rank-based test used to compare multiple paired groups of data (more than or equal to three) that have been tested under different situations. Specifically, the test is useful for analyzing the performance of various models for ranking by comparing them to the proposed model and its results. The test involves ranking all of the groups that were matched, from lower to higher, continuing for every instance, and then calculating the average rank of each column by using the summation for all rows. If all models are equivalent, which is indicated if there is no noteworthy difference among them, the null hypothesis is accepted [91]. In contrast, The null hypothesis is disproved if the models are not equal and subsequently, the acceptance of the alternative hypothesis. If the p-value is less than the threshold at which a difference might be considered significant, It denotes that the nullifying hypothesis is ignored and that at least two models are considerably distinct from one another. In the end, every single model is attributed a rank based on its performance for the task at hand (classification) during the course of the experimentation [92]. The mean rank of all of the models that have been evaluated (FSSWO, FSDE, FSGA, FSACO, FSPSO and Without FS), including the different models used in classification (NB, DTC, KNN and LDA) is calculated as described in Equation (15) and the outcomes are depicted in Tables 4. Then, averaging over all related models' rankings is calculated., by using the division of the number of models that have been used for classification as denominator and the total added value of the ranks of the used models as numerator. The norm of all ranks of all of the used models FSSWO, FSDE, FSGA, FSACO, FSPSO and Without FS) for all of the used datasets is calculated as depicted in Equation (16) and the result is differentiated in Table 4.

TABLE 5. **Calculation of average rank.**

Dataset	Without FS	FSGA	FSDE	FSACO	FSPSO	FSSWO
CM1	5	2	3.75	2.25	3.75	1.25
KC1	5.5	3	2.5	2.75	4	1.75
KC3	5.25	1.75	3.25	2.5	4	1.5
MC1	4.75	2.25	3.25	3	3.25	1
MC2	5	2	3.25	1.75	4	1.25
MW1	4.75	2.75	1.75	3.75	1.75	1.5
PC1	4.75	2	2.75	2.75	2.75	1.5
PC2	4.5	2.75	2.25	2.5	3.25	1.25
PC3	3.75	2.25	3	3.75	3.25	1.5
PC4	5.75	1.5	3.25	3.25	3.25	3
PC5	3.75	3.5	2.5	2.25	3.25	2
Avg Rank	4.795 (6)	1.909 (2)	2.864 (4)	2.772 (3)	3.318 (5)	1.590 (1)

$$AvgRank_{Models} = \frac{\sum rank\ of\ all\ classificationmodels}{Number\ of\ classificationmodels} \tag{15}$$

$$AvgRank(Datasets) = \frac{\sum AvgRank(Models)}{Datasetsused} \tag{16}$$

After Calculation of Average ranks for all the algorithms for each dataset in Table 4, we then calculate the total average rank of all the algorithms as shown in Table 5.

The Holm procedure [93], [94], [95] is commonly employed for Post Hoc testing after the null hypothesis

TABLE 6. Holm's method analysis.

Sl. Nos.	Models used for FS	Value of z	Value of p	$\propto/(M-i)$
1	FSSWO : Without FS	4.018	0.000029	0.01
2	FSSWO : FSGA	0.4	0.344578.	0.0125
3	FSSWO : FSDE	1.597	0.055133	0.0167
4	FSSWO : FSACO	1.482	0.06917	0.025
5	FSSWO : FSPSO	2.166	0.015156	0.05

TABLE 7. ANOVA test result for six models with 11 number of datasets.

	Without FS	FSGA	FSDE	FSACO	FSPSO	FSSWO	Total
N	n_1=11	n_2=11	n_3=11	n_4=11	n_5=11	n_6=11	N=66
$\sum x_i$	T_1=852.35	T_2=929.25	T_3=915.16	T_4=920.81	T_5=897.35	T_6=945.19	$\sum x$ =5460.11
$\sum x_i^2$	$\sum x_i^2$ =6679 2.15	$\sum x_i^2$ =79325. 93	$\sum x_i^2$ =7 6975.34	$\sum x_i^2$ =7 7734.01	$\sum x_i^2$ =7 4257.80	$\sum x_i^2$ =8 1978.41	$\sum x_i^2$ =457072.6 6
Mean $\bar{x}_i$	$\bar{x}_1$=77.48	$\bar{x}_2$=84.47	$\bar{x}_3$=83.19	$\bar{x}_4$=83.71	$\bar{x}_5$=81.57	$\bar{x}_6$=85.92	$\bar{x}$=82.72
Std Dev S_i	S_1=8.64	S_2=9.08	S_3=9.15	S_4=8.08	S_5=10.26	S_6=8.77	

has been taken out of the possibilities and subsequently, the alternative hypothesis has been regarded as the current hypothesis. By utilizing the value of p and the value of z, the Holm procedure assesses the performance of each model relative to others. This is achieved through a comparison with other models. The calculation of z-value is done from Equation (17). The determined z-value and the normal distribution table are used to determine the value of p.

$$z = \frac{\left(AvgRank_i - AvgRank_j\right)}{\sqrt{\frac{V(V+1)}{6U}}} \tag{17}$$

From the values that are calculated and represented in Table 6, an indication is identified that in the majority of the values, the p-values are higher than the value of alpha/(M − i), except in FSSWO and FSPSO models. This shows that, in comparison to all other models save the FSPSO model, the FSSWO model is statistically significant and produces superior outcomes. Analysis of variance (ANOVA) is a statistical analysis technique to determine the experimental results are significance or not by accepting or rejecting the null or alternative hypothesis. It is used to compare means among models using F-distribution to determine the all the models are same or different.In experimental conditions, the FSSWO model is seen to be better than FSPSO model, although when taking statistics into account, the performance variances amongst the models presented are not that substantial. Table 7. shows the result of ANOVA Test for six models with 11 number of datasets. Here, the features of Table 7 represents six models(Without FS, FSGA, FSDE, FSACO, FSPSO and FSSWO) respectively.

VIII. DISCUSSION

This section addresses the brief discussion on the strengths and weaknesses of each compared algorithms used in this experiment. The FS models are generally NP-hard problems. This method searches the solution space entirely and by performing an exhaustive search. This method for searching leads to a significantly high computation time for FS, which increases exponentially with an increase in the feature set. In this experiment, wrapper-based approach of FS approach has been used for analysis. In this approach, the features are selected by training the classifiers using an initial subset of features and founded on the inferences drawn from the previous model, a decision is taken whether to add or remove features. In this experiment, four FS models such as FSGA, FSDE, and FSPSO has been considered for analysis. All the aforesaid FS based optimization algorithms require several hyperparameters to be tuned and sometimes trapped in local optima. To address this issue, the FSSWO algorithm is considered to select the ideal number of the most important features to identify the fault in the software.

IX. CONCLUSION

The proposed FSSWO method efficiently identifies the optimal subset of features using the SWO metaheuristic optimization algorithm. The data considered for predicting the software faults are high dimensional in nature and to select the applicable information from data by using several FS approaches. For the experiments, 12 open-source software defect datasets from the PROMISE repository of NASA were used. FSSWO demonstrates significant improvements in both precision and computational efficiency compared to alternative models like FSGA, FSPSO, FSDE, and FSACO, across various classifiers including Gaussian NB, DTC, KNN, and LDA. The selected feature subset size achieved by FSSWO is competitive with some classifiers and superior with others. Moreover, it pick out the most befitting and optimal number of features while ignoring noise and irrelevant features. Statistical analysis employing the Friedman and Holm procedures confirms the superiority of the proposed FSSWO approach in terms of accuracy. However, it is worth noting that FSSWO requires more time to train the model compared to other conventional FS methods, although it is still faster than the other comparative methods discussed in this study. Future research avenues could explore hybrid approaches integrating different metaheuristic algorithms with wrapper-based FS methods to optimize feature selection for diverse classification models. This FS algorithm could be further improved with different classifiers can be embedded in the future to provide even better accuracy. Additionally, the FSSWO model holds promise for applications such as advertisement click-through rate prediction, weather forecasting, hazardous asteroid classification, disease prediction, space debris classification, and language analysis.

REFERENCES

[1] C. Catal, "Software fault prediction: A literature review and current trends," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4626–4636, Apr. 2011.

[2] Z. Lian, Q. Zeng, W. Wang, T. R. Gadekallu, and C. Su, "Blockchain-based two-stage federated learning with non-IID data in IoMT system," *IEEE Trans. Computat. Social Syst.*, vol. 1, no. 2, pp. 1–10, Jul. 2022.

[3] B. S. Kumar and V. Ravi, "A survey of the applications of text mining in financial domain," *Knowl.-Based Syst.*, vol. 114, pp. 128–147, Dec. 2016.

[4] A. M. Judith, S. B. Priya, R. K. Mahendran, T. R. Gadekallu, and L. S. Ambati, "Two-phase classification: ANN and A-SVM classifiers on motor imagery BCI," *Asian J. Control*, vol. 25, no. 5, pp. 3318–3329, Sep. 2023.

[5] S. Hakak, T. R. Gadekallu, P. K. R. Maddikunta, S. P. Ramu, M. Parimala, C. D. Alwis, and M. Liyanage, "Autonomous vehicles in 5G and beyond: A survey," *Veh. Commun.*, vol. 39, Feb. 2023, Art. no. 100551.

[6] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, May 2009.

[7] G. Xie, G. Hou, Q. Pei, and H. Huang, "Lightweight privacy protection via adversarial sample," *Electronics*, vol. 13, no. 7, p. 1230, Mar. 2024.

[8] X. Zhang, J. Wang, J. Xu, and C. Gu, "Detection of Android malware based on deep forest and feature enhancement," *IEEE Access*, vol. 11, pp. 29344–29359, 2023, doi: 10.1109/ACCESS.2023.3260977.

[9] C. Zheng, Y. An, Z. Wang, H. Wu, X. Qin, B. Eynard, and Y. Zhang, "Hybrid offline programming method for robotic welding systems," *Robot. Comput.-Integr. Manuf.*, vol. 73, Feb. 2022, Art. no. 102238, doi: 10.1016/j.rcim.2021.102238.

[10] S. Xu, H. Dai, L. Feng, H. Chen, Y. Chai, and W. X. Zheng, "Fault estimation for switched interconnected nonlinear systems with external disturbances via variable weighted iterative learning," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 6, pp. 2011–2015, Sep. 2023, doi: 10.1109/TCSII.2023.3234609.

[11] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, Jul. 2018.

[12] B. Amarnath, S. Balamurugan, and A. Alias, "Review on feature selection techniques and its impact for effective data classification using UCI machine learning repository dataset," *J. Eng. Sci. Technol.*, vol. 11, no. 11, pp. 1639–1646, 2016.

[13] G. P. Wang and J. X. Yang, "SKICA: A feature extraction algorithm based on supervised ICA with kernel for anomaly detection," *J. Intell. Fuzzy Syst.*, vol. 36, no. 1, pp. 761–773, Feb. 2019, doi: 10.3233/jifs-17749.

[14] Z. Xuemin, D. Haitao, X. Zenggang, R. Ying, L. Yanchao, L. Yuan, and H. Delin, "Self-organizing key security management algorithm in socially aware networking," *J. Signal Process. Syst.*, vol. 96, nos. 6–7, pp. 369–383, Jul. 2024, doi: 10.1007/s11265-024-01918-7.

[15] S. Maldonado and J. López, "Dealing with high-dimensional class-imbalanced datasets: Embedded feature selection for SVM classification," *Appl. Soft Comput.*, vol. 67, pp. 94–105, Jun. 2018.

[16] O. Villacampa, "Feature selection and classification methods for decision making: A comparative analysis," Ph.D. dissertation, Nova Southeastern Univ., Davie, FL, USA, 2015.

[17] E. Wang, Y. Yang, J. Wu, W. Liu, and X. Wang, "An efficient prediction-based user recruitment for mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 16–28, Jan. 2018, doi: 10.1109/TMC.2017.2702613.

[18] G. Sun, Z. Xu, H. Yu, X. Chen, V. Chang, and A. V. Vasilakos, "Low-latency and resource-efficient service function chaining orchestration in network function virtualization," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5760–5772, Jul. 2020, doi: 10.1109/JIOT.2019.2937110.

[19] M. Mitchell, "Genetic algorithms: An overview," *Complex*, vol. 1, no. 1, pp. 31–39, 1995.

[20] M. Dorigo and T. Stutzle, *Ant Colony Optimization: Overview and Recent Advances*. Cham, Switzerland: Springer, 2019, pp. 311–351.

[21] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," *Mach. Learn.*, vol. 65, no. 1, pp. 31–78, Oct. 2006.

[22] H. S. Stone and J. M. Stone, "Efficient search techniques—An empirical study of the N-Queens problem," *IBM J. Res. Develop.*, vol. 31, no. 4, pp. 464–474, Jul. 1987.

[23] H. E. Evans, "Comparative ethology and the systematics of spider wasps," *Systematic Zoology*, vol. 2, no. 4, p. 155, Dec. 1953.

[24] M. Abdel-Basset, R. Mohamed, M. Jameel, and M. Abouhawwash, "Spider wasp optimizer: A novel meta-heuristic optimization algorithm," *Artif. Intell. Rev.*, vol. 56, no. 10, pp. 11675–11738, Oct. 2023.

[25] T. Naghibi, S. Hoffmann, and B. Pfister, "Convex approximation of the NP-hard search problem in feature subset selection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3273–3277.

[26] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Appl. Soft Comput.*, vol. 27, pp. 504–518, Feb. 2015.

[27] R. S. Wahono, "A systematic literature review of software defect prediction," *J. Software Engineering*, vol. 1, no. 1, pp. 1–16, 2015.

[28] V. U. B. Challagulla, F. B. Bastani, I.-L. Yen, and R. A. Paul, "Empirical assessment of machine learning based software defect prediction techniques," *Int. J. Artif. Intell. Tools*, vol. 17, no. 2, pp. 389–400, Apr. 2008.

[29] M. Cetiner and O. K. Sahingoz, "A comparative analysis for machine learning based software defect prediction systems," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2020, pp. 1–7.

[30] R. Rana, M. Staron, J. Hansson, and M. Nilsson, "Defect prediction over software life cycle in automotive domain state of the art and road map for future," in *Proc. 9th Int. Conf. Softw. Eng. Appl.*, Aug. 2014, pp. 377–382.

[31] D. Sharma and P. Chandra, "Software fault prediction using machine-learning techniques," in *Proc. 1st Int. Conf. SCI*, 2018, pp. 541–549.

[32] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Inf. Softw. Technol.*, vol. 58, pp. 388–402, Feb. 2015.

[33] L. Kumar, S. Misra, and S. K. Rath, "An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes," *Comput. Standards Inter.*, vol. 53, pp. 1–32, Aug. 2017.

[34] J. Walden, J. Stuckman, and R. Scandariato, "Predicting vulnerable components: Software metrics vs text mining," in *Proc. IEEE 25th Int. Symp. Softw. Rel. Eng.*, Nov. 2014, pp. 23–33.

[35] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "A comparative study of ensemble feature selection techniques for software defect prediction," in *Proc. 9th Int. Conf. Mach. Learn. Appl.*, Dec. 2010, pp. 135–140.

[36] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, "Machine learning based methods for software fault prediction: A survey," *Expert Syst. Appl.*, vol. 172, Jun. 2021, Art. no. 114595.

[37] S. Saab, Y. Fu, A. Ray, and M. Hauser, ‘‘A dynamically stabilized recurrent neural network,’’ *Neural Process. Lett.*, vol. 54, no. 2, pp. 1195–1209, Apr. 2022.

[38] A. Kundu, P. Dutta, K. Ranjit, S. Bidyadhar, M. K. Gourisaria, and H. Das, ‘‘Software fault prediction using machine learning models,’’ in *Proc. Int. Conf. Inf. Technol.*, 2022, pp. 170–175.

[39] S. Rani, H. Babbar, G. Srivastava, T. Reddy Gadekallu, and G. Dhiman, ‘‘Security framework for Internet-of-Things-based software-defined networks using blockchain,’’ *IEEE Internet Things J.*, vol. 10, no. 7, pp. 6074–6081, Sep. 2022.

[40] H. Das, B. Naik, and H. S. Behera, ‘‘Optimal selection of features using artificial electric field algorithm for classification,’’ *Arabian J. Sci. Eng.*, vol. 46, no. 9, pp. 8355–8369, Sep. 2021.

[41] H. Das, B. Naik, and H. S. Behera, ‘‘A Jaya algorithm based wrapper method for optimal feature selection in supervised classification,’’ *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 6, pp. 3851–3863, Jun. 2022.

[42] H. Das, S. Chakraborty, B. Acharya, and A. K. Sahoo, ‘‘Optimal selection of features using teaching-learning-based optimization algorithm for classification,’’ *Appl. Intell. Decis. Making Mach. Learn.*, vol. 213, pp. 1–12, Jul. 2020.

[43] B. K. Padhi, S. Chakravarty, B. Naik, R. M. Pattanayak, and H. Das, ‘‘RHSOFS: Feature selection using the rock hyrax swarm optimization algorithm for credit card fraud detection system,’’ *Sensors*, vol. 22, no. 23, p. 9321, Nov. 2022.

[44] H. Dutta, M. K. Gourisaria, and H. Das, ‘‘Wrapper based feature selection approach using black widow optimization algorithm for data classification,’’ in *Computational Intelligence in Pattern Recognition*. Cham, Switzerland: Springer, 2022, pp. 487–496.

[45] I. Guyon and A. Elisseeff, ‘‘An introduction to variable and feature selection,’’ *J. Mach. Learn. Res.*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[46] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, ‘‘Machine learning applications in cancer prognosis and prediction,’’ *Comput. Struct. Biotechnol. J.*, vol. 13, pp. 8–17, Aug. 2015.

[47] J. Li, K. Cheng, S. Wang, F. Morstatter, P. R. Trevino, J. Tang, and H. Liu, ‘‘Feature selection: A data perspective,’’ *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–45, 2017.

[48] L. Yu and H. Liu, ‘‘Feature selection for high-dimensional data: A fast correlation-based filter solution,’’ in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 856–863.

[49] M. B. Kursa and W. R. Rudnicki, ‘‘Feature selection with the Boruta package,’’ *J. Stat. Softw.*, vol. 36, pp. 1–13, Nov. 2010.

[50] F. Alimoglu and E. Alpaydin, ‘‘Combining multiple representations for pen-based handwritten digit recognition,’’ *Turkish J. Elect. Eng. Comput. Sci.*, vol. 9, no. 1, pp. 1–12, 2001.

[51] H. Liu and L. Yu, ‘‘Toward integrating feature selection algorithms for classification and clustering,’’ *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, Apr. 2005.

[52] I. Rish, G. Grabarnik, G. Cecchi, F. Pereira, and G. J. Gordon, ‘‘Closed-form supervised dimensionality reduction with generalized linear models,’’ in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 832–839.

[53] Z. Zhao, F. Morstatter, S. Sharma, A. Alelyani, S. Anand, and H. Liu, ‘‘Advancing feature selection research,’’ *ASU Feature Selection Repository*, vol. 1, pp. 1–28, Nov. 2010.

[54] M. Dash and H. Liu, ‘‘Feature selection for classification. Intelligent data analysis,’’ Tech. Rep., pp. 131–156.

[55] H. Li, J. Zhu, J. Zhang, C. Zong, and X. He, ‘‘Keywords-guided abstractive sentence summarization,’’ in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 8196–8203.

[56] C. L. Huang and C. J. Wang, ‘‘A GA-based feature selection and parameters optimization for support vector machines,’’ *Expert Syst. Appl.*, vol. 31, no. 2, pp. 231–240, 2006.

[57] B. Waske, S. van der Linden, J. A. Benediktsson, A. Rabe, and P. Hostert, ‘‘Sensitivity of support vector machines to random feature selection in classification of hyperspectral data,’’ *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 7, pp. 2880–2889, Jul. 2010.

[58] H. Das, B. Naik, and H. S. Behera, ‘‘A hybrid neuro-fuzzy and feature reduction model for classification,’’ *Adv. Fuzzy Syst.*, vol. 2020, pp. 1–15, Mar. 2020.

[59] H. Das, B. Naik, H. S. Behera, S. Jaiswal, P. Mahato, and M. Rout, ‘‘Biomedical data analysis using neuro-fuzzy model with post-feature reduction,’’ *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2540–2550, Jun. 2022.

[60] H. Das, B. Naik, and H. S. Behera, ‘‘Disease classification using linguistic neuro-fuzzy model,’’ in *Progress in Computing, Analytics and Networking*. Cham, Switzerland: Springer, 2020, pp. 45–53.

[61] H. Das, B. Naik, and H. S. Behera, ‘‘Medical disease analysis using neuro-fuzzy with feature extraction model for classification,’’ *Informat. Med. Unlocked*, vol. 18, Jul. 2020, Art. no. 100288.

[62] H. Das, B. Naik, and H. S. Behera, ‘‘An experimental analysis of machine learning classification algorithms on biomedical data,’’ in *Proc. 2nd Int. Conf. Commun., Devices Comput.*, 2020, pp. 525–539.

[63] H. Das, B. Naik, and H. S. Behera, ‘‘Classification of diabetes mellitus disease (DMD): A data mining (DM) approach,’’ in *Advances in Intelligent Systems and Computing*. Cham, Switzerland: Springer, 2018, pp. 539–549.

[64] H. Das, M. K. Gourisaria, B. K. Sah, S. Bilgaiyan, J. C. Badajena, and R. M. Pattanayak, ‘‘E-healthcare system for disease detection based on medical image classification using CNN,’’ in *Empirical Research for Futuristic e-commerce Systems*. Hershey, PA, USA: IGI Global, 2022, pp. 213–230.

[65] A. A. Freitas, ‘‘Comprehensible classification models: A position paper,’’ *ACM SIGKDD Explorations Newslett.*, vol. 15, no. 1, pp. 1–10, Mar. 2014.

[66] S. Dreiseitl and L. Ohno-Machado, ‘‘Logistic regression and artificial neural network classification models: A methodology review,’’ *J. Biomed. Informat.*, vol. 35, nos. 5–6, pp. 352–359, Oct. 2002.

[67] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, ‘‘Machine learning: A review of classification and combining techniques,’’ *Artif. Intell. Rev.*, vol. 26, no. 3, pp. 159–190, Nov. 2006.

[68] J. Brownlee, ‘‘Imbalanced classification with Python: Better metrics, balance skewed classes, cost-sensitive learning,’’ *Mach. Learn. Mastery*, vol. 1, no. 1, pp. 1–23, 2020.

[69] J.-B. Lamy, ‘‘Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies,’’ *Artif. Intell. Med.*, vol. 80, pp. 11–28, Jul. 2017.

[70] J. D. Novakovic, A. Veljovic, S. S. Ilic, Z. Papic, and M. Tomovic, ‘‘Evaluation of classification models in machine learning,’’ *Theory Appl. Math. Comput. Sci.*, vol. 7, no. 1, p. 39, 2017.

[71] D. L. Verbyla and J. A. Litvaitis, ‘‘Resampling methods for evaluating classification accuracy of wildlife habitat models,’’ *Environ. Manage.*, vol. 13, no. 6, pp. 783–787, Nov. 1989.

[72] Ž. Đ. Vujovic, ‘‘Classification model evaluation metrics,’’ *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 6, pp. 599–606, 2021.

[73] C. Beleites, U. Neugebauer, T. Bocklitz, C. Krafft, and J. Popp, ‘‘Sample size planning for classification models,’’ *Analytica Chim. Acta*, vol. 760, pp. 25–33, Jan. 2013.

[74] M. Carney, B. Webster, I. Alvarado, K. Phillips, N. Howell, J. Griffith, J. Jongejan, A. Pitaru, and A. Chen, ‘‘Teachable machine: Approachable Web-based tool for exploring machine learning classification,’’ in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, Apr. 2020, pp. 1–8.

[75] M. T. Sadiq, H. Akbari, S. Siuly, Y. Li, and P. Wen, ‘‘Alcoholic EEG signals recognition based on phase space dynamic and geometrical features,’’ *Chaos, Solitons Fractals*, vol. 158, May 2022, Art. no. 112036.

[76] M. T. Sadiq, H. Akbari, A. U. Rehman, Z. Nishtar, B. Masood, M. Ghazvini, J. Too, N. Hamedi, and M. K. A. Kaabar, ‘‘Exploiting feature selection and neural network techniques for identification of focal and nonfocal EEG signals in TQWT domain,’’ *J. Healthcare Eng.*, vol. 2021, pp. 1–24, Aug. 2021.

[77] A. M. Anter, A. W. Mohamed, M. Zhang, and Z. Zhang, ‘‘A robust intelligence regression model for monitoring Parkinson's disease based on speech signals,’’ *Future Gener. Comput. Syst.*, vol. 147, pp. 316–327, Oct. 2023.

[78] A. M. Anter and Z. Zhang, ‘‘RLWOA-SOFL: A new learning model-based reinforcement swarm intelligence and self-organizing deep fuzzy rules for fMRI pain decoding,’’ *IEEE Trans. Affect. Comput.*, vol. 1, no. 1, pp. 1–12, Aug. 2023.

[79] A. M. Anter and L. Abualigah, "Deep federated machine learning-based optimization methods for liver tumor diagnosis: A review," *Arch. Comput. Methods Eng.*, vol. 30, no. 5, pp. 3359–3378, Jun. 2023.

[80] A. Thakare, A. M. Anter, and A. Abraham, "Seizure disorders recognition model from EEG signals using new probabilistic particle swarm optimizer and sequential differential evolution," *Multidimensional Syst. Signal Process.*, vol. 34, no. 2, pp. 397–421, Jun. 2023.

[81] A. M. Anter, H. S. Elnashar, and Z. Zhang, "QMVO-SCDL: A new regression model for fMRI pain decoding using quantum-behaved sparse dictionary learning," *Knowl.-Based Syst.*, vol. 252, Sep. 2022, Art. no. 109323.

[82] A. M. Anter, M. Abd Elaziz, and Z. Zhang, "Real-time epileptic seizure recognition using Bayesian genetic whale optimizer and adaptive machine learning," *Future Gener. Comput. Syst.*, vol. 127, pp. 426–434, Feb. 2022.

[83] A. M. Anter, Y. S. Moemen, A. Darwish, and A. E. Hassanien, "Multi-target QSAR modelling of chemo-genomic data analysis based on extreme learning machine," *Knowl.-Based Syst.*, vol. 188, Jan. 2020, Art. no. 104977.

[84] A. M. Anter, G. Huang, L. Li, L. Zhang, Z. Liang, and Z. Zhang, "A new type of fuzzy-rule-based system with chaotic swarm intelligence for multiclassification of pain perception from fMRI," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 6, pp. 1096–1109, Jun. 2020.

[85] A. M. Anter, D. Gupta, and O. Castillo, "A novel parameter estimation in dynamic model via fuzzy swarm intelligence and chaos theory for faults in wastewater treatment plant," *Soft Comput.*, vol. 24, no. 1, pp. 111–129, Jan. 2020.

[86] G. Boetticher. (2007). *The PROMISE Repository of Empirical Software Engineering Data*. [Online]. Available: http://promisedata.org/repository

[87] Y. Wang, F. Li, H. Zheng, L. Jiang, M. F. Mahani, and Z. Liao, "Human trust in robots: A survey on trust models and their controls/robotics applications," *IEEE Open J. Control Syst.*, vol. 1, no. 1, pp. 58–86, Dec. 20, 2023.

[88] C. Carlberg, *Statistical Analysis: Microsoft Excel*. Seattle, WA, USA: Que Publishing, 2014.

[89] J. Hao, P. Chen, J. Chen, and X. Li, "Multi-task federated learning-based system anomaly detection and multi-classification for microservices architecture," *Future Gener. Comput. Syst.*, vol. 159, pp. 77–90, Oct. 2024, doi: 10.1016/j.future.2024.05.006.

[90] X. Fu, P. Pace, G. Aloi, A. Guerrieri, W. Li, and G. Fortino, "Tolerance analysis of cyber-manufacturing systems to cascading failures," *ACM Trans. Internet Technol.*, vol. 23, no. 4, pp. 1–23, Nov. 2023, doi: 10.1145/3579847.

[91] J. Zhang, D. Yang, W. Li, H. Zhang, G. Li, and P. Gu, "Resilient output control of multiagent systems with DoS attacks and actuator faults: Fully distributed event-triggered approach," *IEEE Trans. Cybern.*, vol. 1, no. 1, pp. 1–10, Oct. 2024, doi: 10.1109/tcyb.2024.3404010.

[92] M. R. Sheldon, M. J. Fillyaw, and W. D. Thompson, "The use and interpretation of the Friedman test in the analysis of ordinal-scale data in repeated measures designs," *Physiotherapy Res. Int.*, vol. 1, no. 4, pp. 221–228, Nov. 1996.

[93] A. Y. Gordon and P. Salzman, "Optimality of the Holm procedure among general step-down multiple testing procedures," *Statist. Probab. Lett.*, vol. 78, no. 13, pp. 1878–1884, Sep. 2008.

[94] M. Aickin and H. Gensler, "Adjusting for multiple testing when reporting research results: The Bonferroni vs Holm methods," *Amer. J. Public Health*, vol. 86, no. 5, pp. 726–728, May 1996.

[95] H. Abdi, "Holm's sequential Bonferroni procedure," *Encyclopedia Res. Des.*, vol. 1, no. 8, pp. 1–8, 2010.

[96] A. Fatima, R. Maurya, M. K. Dutta, R. Burget, and J. Masek, "Android malware detection using genetic algorithm based optimized feature selection and machine learning," in *Proc. 42nd Int. Conf. Telecommun. Signal Process. (TSP)*, Jul. 2019, pp. 220–223.

[97] M.-Y. Cho and T. T. Hoang, "Feature selection and parameters optimization of SVM using particle swarm optimization for fault classification in power distribution systems," *Comput. Intell. Neurosci.*, vol. 2017, pp. 1–9, Jun. 2017.

[98] A. Dixit, A. Mani, and R. Bansal, "Feature selection for text and image data using differential evolution with SVM and Naïve Bayes classifiers," *Eng. J.*, vol. 24, no. 5, pp. 161–172, Sep. 2020.

HIMANSU DAS received the B.Tech. degree from the Institute of Technical Education and Research, Odisha, India, the M.Tech. degree in computer science and engineering from the National Institute of Science and Technology, Odisha, and the Ph.D. degree in engineering from the Veer Surendra Sai University of Technology (VSSUT), Odisha. He is currently an Associate Professor with the School of Computer Engineering, Kalinga Institute of Industrial Technology (KIIT), Deemed to be University, Bhubaneswar, India. He has published several research papers in various international journals and has presented at conferences. He has also edited several books published by IGI Global, Springer, CRC Press, and Elsevier. He has also served on many journals and conferences as an editorial or reviewer board member. He is proficient in the field of computer science engineering and served as the organizing chair, the publicity chair, and acted as a member of the technical program committees for many national and international conferences. He is also associated with various educational and research societies, such as IET, IACSIT, ISTE, UACEE, CSI, IAENG, and ISCA. His research interests include the field of data mining, soft computing, and machine learning. He has also more than 12 years of teaching and research experience in various engineering colleges and universities.

SWARNAVA DAS received the B.Tech. degree from the Kalinga Institute of Industrial Technology (KIIT), Deemed to be University, Bhubaneswar, India. He is currently with Dish Network Ltd., as a Software Engineer. He was an Intern for the companies Expand AI and Highradius for a tenure of two months and ten months, respectively. His research interests include machine learning computer vision and artificial intelligence.

MAHENDRA KUMAR GOURISARIA (Member, IEEE) received the master's degree in computer application from Indira Gandhi National Open University, New Delhi, the M.Tech. degree in computer science and engineering from the Biju Patnaik University of Technology, Rourkela, and the Ph.D. degree from KIIT Deemed to be University, Bhubaneswar, Odisha. He is currently an Assistant Professor with the School of Computer Engineering, KIIT Deemed to be University. He has an experience of more than 20 years in academia and nine years in research. He has guided more than 60 B.Tech. students in their project work and seven M.Tech. thesis. He has published more than 100 research papers in different book chapters, international journals, and conferences of repute. His google scholar citation is more than 1000 with an H-index of 17 and i10- index of 35. He has served as an organizing committees members for various conferences and workshops. He chaired session in many international conferences and acted as a reviewer in many reputed journals of Springer and Hindawi and many reputed conferences. His research interests include cloud computing, machine learning, deep learning, data mining, soft computing, and internet and web technology. He is a member of IAENG and UACEE; and a Life Member of ISTE, CSI, and ISCA.

SURBHI BHATIA KHAN (Senior Member, IEEE) is currently pursuing the Doctorate degree in computer science and engineering in the area of machine learning and social media analytics. She received the project management professional certification from the reputed Project Management Institute, USA. She is currently an Assistant Professor with the Department of Data Science, School of Science, Engineering and environment, University of Salford, Manchester, U.K. She has more than ten years of academic and teaching experience in different universities. She has published many papers in reputed journals and conferences in high indexing outlets. She has published ten international patents from India, Australia, and USA; and also authored and edited around 12 books. Her research interests include information systems, artificial intelligence, and data science.

AHLAM ALMUSHARRAF received the Ph.D. degree in business administration with concentration in information systems. She is currently an Assistant Professor with the College of Business and Administration, Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia. Her research interests include AI, IS applications, social media, e-commerce, and ICT.

ABDULLAH I. ALHARBI received the Ph.D. degree in computer science from the University of Birmingham, U.K., in 2022. He is currently an Assistant Professor with the Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University. He specializes in artificial intelligence (AI), natural language processing (NLP), machine learning, and deep learning, focusing on analyzing social media to explore social phenomena and gather public opinion. He has extensive experience in detecting sentiment, emotions, events, abusive content from online textual data, and applying advanced AI-driven methodologies to address real-world challenges.

T. R. MAHESH (Senior Member, IEEE) is currently an Associate Professor and the Program Head of the Department of Computer Science and Engineering, Faculty of Engineering and Technology, JAIN (Deemed-to-be University), Bengaluru, India. He has to his credit more than 100 research articles in Scopus/WoS and SCIE indexed journals of high repute. He has been an editor for books on emerging and new age technologies with publishers like Springer, IGI Global, and Wiley. He has served as a reviewer and a technical committee member for multiple conferences and journals of high reputation. His research interests include image processing, machine learning, deep learning, artificial intelligence, the IoT, and data science.