Received: 20 December 2020 | Revised: 17 June 2021 | Accepted: 25 June 2021

DOI: 10.1111/exsy.12768

ORIGINAL ARTICLE

Expert Systems WILEY

Meta-heuristic optimization algorithm for predicting software defects

Mahmoud A. Elsabagh[1] | Marwa S. Farhan[2,3] | Mona G. Gafar[1,4]

[1]Department of Machine Learning and Information Retrieval, Faculty of Artificial Intelligence, Kafrelsheikh University, Kafrelsheikh, Egypt

[2]Department of Information Systems, Faculty of Computers and Artificial Intelligence, Helwan University, Cairo, Egypt

[3]Department of Information Systems, Faculty of Informatics and Computer Science, British University in Egypt, Cairo, Egypt

[4]Department of Computer Science, College of Science and Humanities in Al-Sulail, Prince Sattam bin Abdulaziz University, Kharj, Saudi Arabia

Correspondence
Mahmoud A. Elsabagh, Department of Machine Learning and Information Retrieval, Artificial Intelligence, Kafrelsheikh University, 33511 Kafrelsheikh, Egypt.
Email: mahmoud_mohsen@fci.kfs.edu.eg

Abstract

Software engineering companies strive to improve software quality by predicting software defects-prone modules. Although various data mining methods have been developed, unstable accuracy rates are still critical issues owing to the imbalanced nature and high dimensionality of software defect datasets. To deal with this issue, we propose a spotted hyena, a novel meta-heuristic optimization algorithm for predicting software defects. Support and confidence in classification rules are the basis of a multi-objective fitness function that assists the spotted hyena algorithm in serving as a classifier by finding the fittest classification or standard rules among individuals. Experiments were conducted on four NASA software datasets, JM1, KC2, KC1, and PC3. The spotted hyena classifier provides an accuracy of 85.2, 84, 89.6, and 81.8%, respectively, for these datasets. These accuracy rates are better than those achieved using other popular data mining techniques. We also discuss other classification measures in connection with the experimental results, such as precision, recall, and confusion matrices, in connection with the experimental results. Moreover, the Gaussian mixture model is used to study the uncertainty quantification of the proposed classifier. The study proved the feasible performance of the spotted hyena classifier in four different case studies.

KEYWORDS
confidence, software defect prediction, software metric, spotted hyena optimizer algorithm, support

1 | INTRODUCTION

Software engineering demands high quality and reliability to fulfil user requirements at a limited time. Software defect prediction (SDP) models can assist quality assurance teams in investigating and testing software products using an effective allocation of limited resources (Raukas, n.d.; Nam, 2014).

Initially, software companies used manual testing to detect defects. These required 27% of a project's effort and could not address all software defects. Software companies often do not have the necessary stuff and time to remove all faults before product release. This has a negative impact on company reputations and overall product quality. SDP models can assist these companies in solving critical problems and allocating resources to the most defect-prone code (Raukas, n.d.).

Many techniques have been used in predicting software defects, such as Naïve Bayes (NB; Okutan & Yıldız, 2014), bagging (Kuncheva et al., 2002), boosting (Aljamaan & Elish, 2009), support vector machines (SVMs; Shan et al., 2014), and C4.5 (Koru & Liu, 2005).

Correction added on 25 August 2021, after first online publication: Affiliations 2 and 3 have been corrected in this version.

Moreover, recent studies have recommended data mining methodologies that are using machine learning (ML) as important models. Detection accuracy is the most challenging issue for SDP models (Jayanthi & Florence, 2019). Previous experience and studies can assist in predicting defects in software products.

Our research uses the spotted hyena optimizer (SHO; Dhiman & Kumar, 2017; Kumar & Kaur, 2020), a novel meta-heuristic algorithm, as a classifier for foreseeing faults in within-project (training and testing phases in the same project; Elsabagh et al., 2020). The principal idea of the SHO is the behaviour of spotted hyenas. Support and confidence are used as a Multi-Objective Fitness Function (*MOF*) to locate the fittest classification rules during the experiments. This improves classification accuracy in predicting software defects. Ultimately, this can assist companies in enhancing quality, effort, time estimation, reliability checking, and risk reduction during development.

The research contribution is devoted to improving SDP in within-projects that suffer from various challenges such as detection accuracy and precision (Kaur & Sharma, 2018). The paper proposes the SHO algorithm as a classifier that iterates to find the optimal individuals learning from training data (60% percentage of the NASA software dataset). The classification model is a set of if-then rules used to detect defects on unseen instances during the test process on the rest of the dataset (40%). Besides, accuracy and precision scores are computed and compared with literature data mining techniques using the WEKA tool (Witten et al., 2016). Finally, uncertainty quantification (UQ; Abdar et al., 2021; Kompa et al., 2021) study using Gaussian mixture model (GMM; Shafiullah et al., 2020) is implemented to measure the amount of uncertainty in the SHO classifier output. Using the SHO algorithm is a novelty particularly in within- projects software defect detection according to our search before beginning the experiments.

The rest of the paper is organized as follows. Section 2 surveys recent defect prediction models. Section 3 describes the mathematical equations used in the SHO. The proposed classifier is presented in Section 4. Section 5 discusses our experimental studies. Section 6 concludes and discusses future work.

2 | RELATED WORK

Meta-heuristic algorithms (Gandomi et al., 2013) are optimization techniques that tend to merge randomization, local search, and global exploration to find a feasible and efficient solution to complicated problems quickly. These algorithms got great interest from researchers in different fields like feature selection, classification, and optimization problems. Table 1 lists abstracted comparison between meta-heuristic algorithms mentioned in this section.

Pourpanah et al. (2016) utilized meta-heuristic algorithms for data classification. They use a Fuzzy ARTMAP classifier with Q-learning during the training process and a genetic algorithm (GA) for rule extraction. The hybrid model provides the prediction for the target class in addition to a fuzzy if-then rule to explain the prediction.

Das and Saha (2021) used six algorithms (Whale Optimization Algorithm, Eagle Perching Optimization, Dragonfly Algorithm, Flower Pollination Algorithm, Bird Swarm Algorithm, and Firefly Algorithm) for real-life structural health monitoring They utilized real-life quarter-scaled ASCE-Benchmark structure using stiffness-based fitness. The performance of all the algorithms is relatively good except for BSA and FA because of local optima.

Pławiak Pawełand Abdar et al. (2020) proposed Deep Genetic Hierarchical Network of Learners (DGHNL) for credit scoring. They utilized several learners including Support Vector Machines, k-Nearest Neighbours, Probabilistic Neural Networks, and fuzzy systems. DGHNL model with a 29-layer achieved the prediction accuracy of 94.60% for the Statlog German credit approval data.

Eid et al. (2021) addresses multi-objective water cycle algorithm (MOWCA) for truss optimization problems employing the real-world water cycle. The results were compared with various literature algorithms. The proposed technique has an excellent performance in optimizing multi-objective and truss problems.

Abdar et al. (2019) diagnosed coronary artery disease (CAD) by a nested ensemble nu-Support Vector Classification (NE-nu-SVC) that combines several conventional ML strategies and outfits learning strategies for the compelling conclusion of CAD. They approved the model utilizing two well-known CAD datasets (Z-Alizadeh Sani and Cleveland). The NE-nu-SVC model gives the most noteworthy exactness of 94.66% and 98.60% within the Z-Alizadeh Sani and Cleveland CAD datasets.

Predicting software defects is an interesting research topic in software engineering. It plays a vital role in assisting software engineers and developers in accelerating marketing time and producing more valuable software. Many researchers have recently taken an interest in predicting software defects. This section discusses common related work on that topic.

Dam et al. (2018) presented a model for predicting software defects by automatically learning features that represent a module and using them for defect prediction. Their prediction system is built on a tree-structured long short-term memory network matched with an abstract syntax tree source code representation for powerful deep learning.

Hammouri et al. (2018) presented a model for predicting software defects using ML algorithms. Basing their method on historical data for predicting future software defects, they used three supervised ML algorithms, artificial neural networks (ANNs), Decision Trees (DTs), and NB.

TABLE 1 Comparison between some meta-heuristic algorithms and their applications in literature

#	Reference and year	Application	Meta-heuristic optimization algorithm	Task results
1	(Pourpanah et al., 2016)	Data Classification	ARTMAP classifier with Q-learning and Genetic Algorithm (GA)	The experimental result emphatically illustrates the potential effect of QFAM-GA within the real-life environment
2	(Das & Saha, 2021)	Health Monitoring	Whale Optimization Algorithm, Eagle Perching Optimization, Dragonfly Algorithm, Flower Pollination Algorithm, Bird Swarm Algorithm and Firefly Algorithm	The performance of all the algorithms is relatively good except for BSA and FA because of local optima.
3	(Pławiak Paweł and Abdar et al., 2020)	Credit Scoring	Deep Genetic Hierarchical Network of Learners (DGHNL)	Prediction accuracy of 94.60% for the Statlog German credit approval data.
4	(Eid et al., 2021)	Optimization	Multi-Objective Water Cycle Algorithm (MOWCA)	The proposed technique has an excellent performance.
5	(Abdar et al., 2019)	Diagnosis of Coronary artery disease (CAD)	nu-Support Vector Classification (NE-nu-SVC)	The exactness of 94.66% and 98.60% within the Z-Alizadeh Sani and Cleveland CAD datasets, respectively.
6	(Dam et al., 2018)	Predicting software defects	Deep Learning.	A tree-structured long short-term memory network matched with tree source code
7	(Hammouri et al., 2018)	Predicting software defects	Artificial Neural Networks (ANNs), Decision Trees (DTs) and NB.	Presented a model for predicting software defects using Machine Learning (ML) algorithms.
8	(Jayanthi & Florence, 2019)	Predicting software defects	Principal component analysis (PCA) & ANN	Reduced errors to predict future defects.
9	(Singh & Chug, 2017)	Predicting software defects	NB, Particle Swarm Optimization (PSO), Linear Classifiers (LCs), ANNs, and DTs.	Used KEEL and validated by k-fold validation.
10	(Wei et al., 2019)	Predicting software defects	SVM algorithm as a classifier.	Parameters of the model are optimized by 10-fold cross-validation and grid search.
11	(Alaia et al., 2018)	Optimization	GA and PSO	GA is superior to PSO, which gives little advancement for clustered populations.
12	(Kumar et al., 2014)	Predicting software defects	Asymmetric kernel PCA and partial least squares.	Applying using a kernel function with SOFTLAB on NASA datasets.

Jayanthi and Florence (2019) proposed an approach based on feature reduction. They applied principal component analysis (PCA) enhanced by the estimation of maximum likelihood to reduce errors in PCA and an ANN as a classifier to predict future defects.

Singh and Chug (2017) analysed the widely used and most well-known ML algorithms, such as NB, Particle Swarm Optimization (PSO), linear classifiers (LCs), ANNs, and DTs, using KEEL and validated them using the technique of k-fold validation.

Wei et al. (2019) approached the problem of predicting software defects by using a model based on the SVM algorithm as the basic classifier. The parameters of the model are optimized by using a 10-fold cross-validation and grid search method.

Xu et al. (2016) noticed that conventional techniques used feature reduction schemes to reduce irrelevant metrics (features), but that some metrics were still important and influenced performance in predicting software defects. To face this challenge, they used the correlation technique of maximal information and computed from the selected features followed by clustering at a later stage.

Kumar et al. (2014) proposed two classifiers for predicting defects using asymmetric kernel PCA and partial least squares, applying them using a kernel function with SOFTLAB on NASA datasets.

Elsabagh et al. (2020) optimized an ML classifier to predict defects in cross-projects. The classifier is trained on projects available in the NASA dataset, and then the classification model is used on different or new projects. That research assisted in detecting software defects when insufficient historical data was available.

Khan (2020) used eight datasets to compare notable classifiers of hybrid ensembles and supervised learning. The results showed that bagging and an AdaBoost SVM achieved high rates in accuracy, recall, an area under the curve, and F-measure.

Meta-heuristic techniques like genetic algorithms (GA; Hosseinabadi et al., 2019) and particle swarm optimization (PSO; Bansal, 2019) are also widely used for optimization and data mining in literature. Both strategies start with an arbitrary population and use a fitness function to assess. PSO begins from an initial swarm and utilizes cooperation to find the best solution where GA utilizes genetic operations. (Alaia et al., 2018) compared GA and PSO implementation in an optimization problem. They concluded that GA is the favourable approach although that it endures a few issues. It is troublesome to decide the foremost compelling crossover methodology in advance. Moreover, through its genetic processes, weak solutions may proceed to be the portion of the cosmetics of future candidate arrangements. In their experiment, GA is superior to PSO, which gives little advancement for clustered populations. Compared with GA, the points of interest of PSO are that it is simple to actualize and there are few parameters to alter.

The previous comparison between GA and PSO pushed us to use new meta-heuristic techniques to avoid their difficulties. SHO is an efficient meta-heuristic algorithm that requires fewer operators and hence simple implementation. During several comparisons in our experiment, SHO obtained the most optimal solutions at a limited time, high efficiency, and low complexity.

The literature work presents the need for defect prediction in the beginning periods of the SDLC. The related work still struggles from precision and accuracy according to (Kaur & Sharma, 2018). The proposed SHO classifier is devoted to improving SDP in projects that suffer from low detection accuracy and precision (Kaur & Sharma, 2018). This paper compares the proposed model results with the most popular data mining techniques found in literature work. The comparisons are illustrated using different NASA projects by the WEKA mining tool (Witten et al., 2016) in the experimental results subsection. In addition, the paper introduces a novel artificial intelligent algorithm. The proposed model advantages the capabilities and intelligence of the spotted hyena in modelling a new data mining classifier. The next section discusses the SHO algorithm.

3 | SHO ALGORITHM

Meta-heuristic techniques are divided into three categories, swarm, evolutionary, and physical-based (Ren et al., 2014). SHO (Dhiman & Kumar, 2017) is a novel meta-heuristic and is implemented in three steps, searching for, encircling, and attacking prey. This algorithm is compared here with 29 well-known testing functions and eight recently developed algorithms (Dhiman & Kumar, 2017). The results show that SHO performs better than the other meta-heuristic techniques. SHO can reach the optimal solution in a limited time with high efficiency and low complexity(Dhiman & Kumar, 2017; Kumar & Kaur, 2020). The following subsections explain the mathematical equations used in the spotted hyena.

3.1 | Mathematical model of encircling

Dhiman and Kumar (2017) modelled the social chain of spotted hyenas mathematically. Since spotted hyenas are familiar with the location of their prey, the prey can be encircled. They considered the current most feasible competitor solution as the objective prey or target. This present best is assumed to be close to the perfect as the search area was not known previously. The other individuals are expected to change their locations according to the most feasible search competitor solution. The following equations represent the encircling of prey mathematically:

$$\vec{D}_h = \left| \vec{B} \times \vec{P}_p(x) - \vec{P}(x) \right| \tag{1}$$

$$\vec{P}(x+1) = \vec{P}_p(x) - \vec{E} \times \vec{D}_h \tag{2}$$

where $\vec{D}_h$ characterizes the separation between the prey and hyena, $\vec{B}$ and $\vec{E}$ are coefficients, $\times$ is the multiplication operation, x shows the current iteration, $\vec{P}_p$ shows the prey position, and $\vec{P}$ is the spotted hyena position. || represents the absolute value.

$\vec{B}$ and $\vec{E}$ are calculated as follows:

$$\vec{B} = 2 \times r\vec{d}_1 \tag{3}$$

$$\vec{E} = 2\vec{h} \times r\vec{d}_2 - \vec{h} \tag{4}$$

$$\vec{h} = 5 - (\text{iteration} \times (5 \div \text{Max}_{\text{iteration}})) \tag{5}$$

where iteration $= 1, 2, 3 \ldots, \text{Max}_{\text{iteration}}$.

$\vec{h}$ decreases from five: 0 through $\text{Max}_{\text{iteration}}$, and $\vec{rd}_1$,$\vec{rd}_2$ are random vectors between zero and one.

3.2 | Mathematical model of chasing

Hyenas typically chase in groups, relying on a system of companions and the capacity to perceive the area in which prey are located (Dhiman & Kumar, 2017). The following equations represent the hunting of prey mathematically:

$$\vec{D}_h = |\vec{B} \times \vec{P}_h - \vec{P}_k| \tag{6}$$

$$\vec{P}_k = \vec{P}_h - \vec{E} \times \vec{D}_h \tag{7}$$

$$\vec{C}_h = \vec{P}_k + \vec{P}_{k+1} + \ldots + \vec{P}_{k+N} \tag{8}$$

where $\vec{P}_h$ characterizes the position of the first feasible hyena, and $\vec{P}_k$ represents other hyenas. N is the number of hyenas, calculated as follows:

$$N = \text{count}_{\text{nos}}\left(\vec{P}_h, \vec{P}_{h+1}, \vec{P}_{h+2}, \ldots, \left(\vec{P}_h + \overrightarrow{M}\right)\right) \tag{9}$$

where $\vec{M}$ a random number between 0.5 and one, nos characterizes the number of solutions, and $\vec{C}_h$ is a set of N optimal solutions.

3.3 | Mathematical model of assaulting

Modelling assaulting mathematically requires reducing the $\vec{h}$ value. The assortment in $\vec{E}$ is also decreased to change the value of $\vec{h}$, which can decrease from five to zero through cycles (Dhiman & Kumar, 2017). Equation (10) describes this phase mathematically:

$$\vec{P}(x+1) = \vec{C}_h \div N \tag{10}$$

where $\vec{P}(x+1)$ saves the most feasible hyena and refreshes the positions of the others.

4 | OPTIMIZING CLASSIFICATION RULES BY SHO

Defect prediction is widely considered to be one of the most essential software engineering tasks. There are many real-life problems in software engineering design, such as increasing complexity and efficiency, requiring meta-heuristic techniques for obtaining optimal solutions (Ren et al., 2014). Section 2 explains briefly how SDP can be based on ML, but overall performance and accuracy classification is still challenging (Kaur & Sharma, 2018).

To deal with this issue, our work optimizes if-then classification rules by a novel meta-heuristic optimization algorithm, SHO, based on the conduct of spotted hyenas (Elsabagh et al., 2020) for within-project defect prediction.

Suppose that we have a pool of if-then rules where the conditional part of these rules is software engineering metrics, and the decision part is whether defective or non-defective. Under appropriate circumstances, the search algorithm would find the optimal set of rules that forms the classification model. But what are the appropriate circumstances? Here, we define them as the appropriate fitness function that helps the algorithm to rationally select the optimal if-then rules from the perspective of the dataset (i.e., only select the rule that matches the largest number of instances from the training data). First, we use a random pool of rules, and then we implement two fitness functions (MOF) to find the best rules. The MOF is composed of the degree of support and confidence of the rules (explained in Section 4.1). These two functions measure how well the rule represents the dataset. Hence, we could optimize the classification model using the SHO algorithm, which is selected upon its novelty and efficiency.

SHO algorithm is an efficient optimizer for constrained and unconstrained engineering problems engineering (Dhiman & Kumar, 2017). The problem of optimizing a classification model could be formatted as a constrained problem (i.e., find the optimal if-then rules which optimally match the dataset understudy). Hence, we preferred to utilize the SHO as a meta-heuristic search algorithm.

Figure 1 shows the data flow and major processes in the proposed classifier. The instances or objects are created from archives of software such as version control. Each method, class, and file are represented by an instance, which is labelled as defective or not.

The datasets are discretized using RapidMiner 5.3 (Mierswa & Klinkenberg, 2018) because it is difficult to perform exact matching between individuals (rules) of a population and the instances of datasets to calculate the *MOF*. A combination of the support and confidence degrees of the generated rules serves as an *MOF* to evaluate suitable classification rules. The SHO algorithm iterates to locate the optimal individuals (rules) based on a subset of the dataset with a percentage of 60% (training dataset). The remainder of the software dataset (40%) is saved for testing, in which new instances are classified as defective or not. Based on the test results, a detailed output of algorithm accuracy, precision, sensitivity, recall, specificity, and F-measure is reported. These measures are used for comparisons with other popular techniques found in the literature, including C4.5, ANNs (Wang et al., 2017; Zhang et al., 2014), random forest, bagging, K-NN, SVMs, and NB. The following subsections summarize the *MOF* and SHO classifier phases.

4.1 | MOF

An objective function (Deb, 2014) is also called an evaluation function. Such a function assesses how close a given solution is to the ideal solution of the desired problem. It determines how to fit a solution is. This subsection describes support and confidence (Qodmanan et al., 2011) as the objective functions during our experiments to locate the fittest standard rules. First, we calculate the support of the rule by the number of rows that satisfy the standard rule. This is calculated as follows:

$$\text{Support} = (\text{COUNT\_S}/\text{R}) \times 100 \tag{11}$$

Second, the confidence of the rule is computed. This is the ratio of the total number of modules that fulfil the entire standard rule to the total number of modules that satisfy the antecedent, calculated as follows:

$$\text{Confidence} = (\text{COUNT\_S}/\text{COUNT\_C}) \times 100 \tag{12}$$

where COUNT_C is the total number of modules that satisfy the antecedent of the standard rule, COUNT_S is the total number of modules that fulfil the standard rule, and *R* is the number of rows. Finally, the *MOF* for each rule is computed as follows:

$$MOF = w_1 \times \text{Support} + w_2 \times \text{Confidence} \tag{13}$$

where w_1 and w_2 are weights attached to the support and confidence based on their relative importance.

4.2 | SHO phases and flowchart as a classifier

This section summarizes the stages of the proposed classifier based on SHO and the *MOF* (support and confidence).

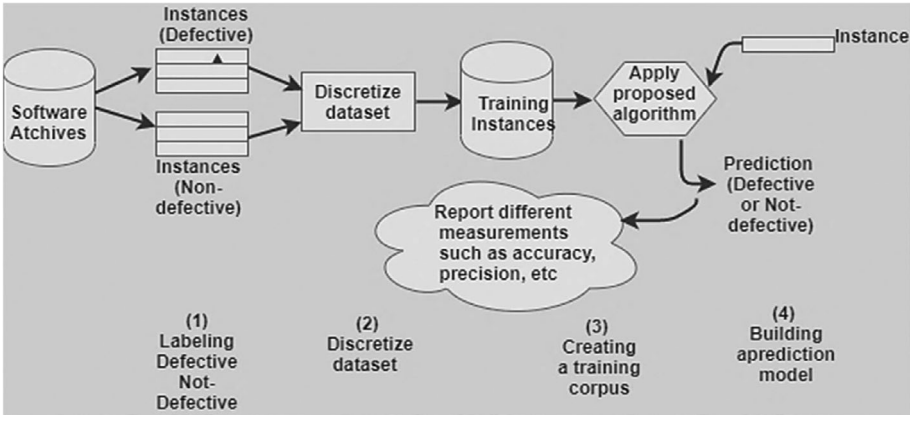

FIGURE 1 General model of software defect prediction (within projects) using spotted hyena optimizer as a classifier

The hyena population is created, and the initial parameters are chosen as shown in Figure 2. The *MOF* is applied for each instance of the spotted hyena by using a combination of the support and confidence degrees of the generated population to evaluate the suitability of the rules of the classification. Next, the location of each hyena is updated by learning from the hyena with the maximum *MOF*. Then, the SHO algorithm iterates to find the optimal individuals based on a percentage of 60% (training data). The output of the training process that returns the classification rules is used as input for the testing process on the remainder of the software dataset (40%) to predict and classify the new instances as defective or not.

Our method also uses the steps shown in Algorithm 1 to implement the phases of the proposed SHO classifier. In addition, we answer the question here of how to implement the SHO algorithm to classify and predict whether another test instance is defective using support and confidence as the *MOF*.

The SHO algorithm iterates initially on a random population of rules to select the optimal set. Each rule is composed of conditional metrics and the class label. The main goal of SHO is to find the best set of rules that represent the dataset. The MOF (explained in Section 4.1) is

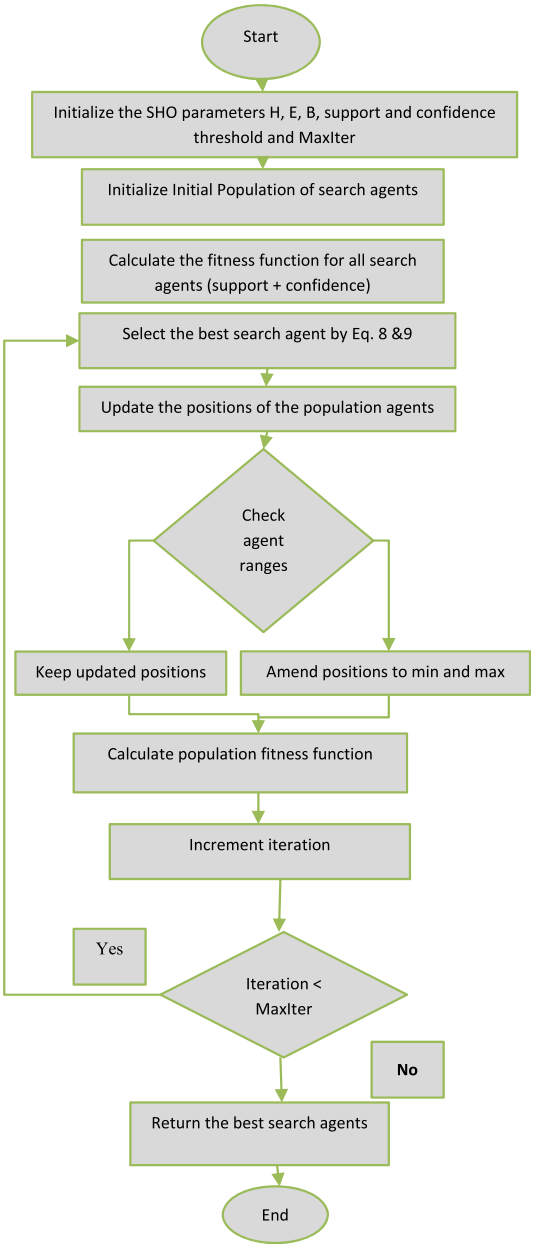

FIGURE 2 Spotted hyena optimizer classifier flowchart

Algorithm 1

Steps of the SHO classifier through the within-projects SDP

Input: nPop = 500, PD = 22 or 38, MaxIt = 50, Ds, Iteration = 0,

Output: Return the solution of the best classification.

1: **Start** proposed classifier
2: H = 5- (Iteration * [5/MaxIt])
3: E = 2*H*rand(PD,1)-H
4: B = 2*rand(PD,1)
5: pop = init (PD, nPop)
6: MOF = fitness_Psc(pop)
7: $\vec{P}_h$= The best individual (classification rule) is investigated in the inquiry area.
8: $\vec{C}_h$= Characterize the cluster of ideal solutions using Equations (8, 9).
9: **For** Iteration: Max-Iteration
10: current_fit = max(ft)
11: current_sol = H_best
12: **For** i = 1:n % for each hyena
13: D = |(B^T.*H_best) – x|
14: xnew(i,:) = round |(H_best-(E^T.*D) |
15: **end For**
16: Refresh E, B, and h.
17: **for** i = 1:n
18: **for** j = 1:PD-1
19: **if** $xnew_{i,j} < l_j$ || $xnew_{i,j} > u_j$
20: $xn_{i,j} = xn_{i,j}$;
21: **else**
22: $xn_{i,j} = xnew_{i,j}$;
23: **end**
24: **end**
25: MOF = fitness_Psc
26: Refresh the vector $\vec{P}_h$ to check whether there is a more feasible individual than the previous one
27: Refresh the cluster of spotted hyenas $\vec{C}_h$
28: Iteration = Iteration+1
29: **end For**
30: Return $\vec{P}_h$
31: **end** proposed classifier

composed of support and confidence measures. These two measures calculate the matching between the rules and dataset instances. At the end of iterations, the final population is processed to remove repetitions and extract the unique rule, which will represent the classification model.

The test process decides of defective or not for new instances. The conditional metrics of these instances are input into the classification model to find the rules that best match them. The rule label (class attribute) is compared to the target class which is the actual class of the instance. The true positives, false positives, true negatives, and false negatives are calculated to measure the performance functions of the classification model. These functions are the accuracy, precision, sensitivity, recall, specificity, and F-score which will judge the SHO algorithm in finding the appropriate if-then rules (classification model).

First, the initial parameters are computed as shown in Equations 3, 4, and 5. The *MOF* for each spotted hyena of the population is calculated. The *MOF* used during the process of optimization is the support and confidence, as shown in Equations 11, 12, and 13. This *MOF* supports SHO by finding the fittest standard rules among initial individuals (randomly created rules). The SHO algorithm iterates to locate optimal individuals and then characterizes the cluster of ideal solutions using Equations (8) and (9). The separation between the hyena and prey is characterized, and the spotted hyena is updated. Then, the position and memory are updated by checking whether any hyena moves past the limit in the search area

and are changed if necessary. Then, the *MOF* and the new values of the coefficient vectors of the updated spotted hyena are calculated. The next step is to refresh the cluster of spotted hyenas $\vec{C}_h$ and the vector $\vec{P}_h$ to determine whether a better solution than the previous one is produced. Finally, the output of the proposed classifier returns the best individuals (classification rules).

5 | EXPERIMENTAL EVALUATION

This section explains the study of the SHO in SDP and how to assess the performance of this algorithm as a classifier based on support and confidence.

5.1 | Experiment setup

These experiments are conducted using MATLAB R2016a (MATLAB, 2016), RapidMiner 5 (Mierswa & Klinkenberg, 2018), the open-source datasets PROMISE (Software Defect Dataset: Promise repository, n.d.) and OPENML (OpenML, n.d.), and WEKA 3.6 (Witten et al., 2016) on a PC with Intel(R) Core (TM) i5 CPU and 4 GB RAM.

In this experiment, we used NASA software datasets from the PROMISE and OPENML archives. NASA data is composed of various software datasets. During experiments, we focused on JM1, KC2, KC1, and PC3. Distinctive features are available in the respective datasets. Table 2 discusses different parameters associated with the datasets including the number of attributes (software metrics), modules, defectives, and percentage of depicted defects. There are some software metrics coded in parameters like v(g), ev(g), iv(g), and so on. These metrics are used in JM1, KC2, and KC1 datasets. We preferred to describe them separately in Table 3 (Elsabagh et al., 2020). There are some common metrics between the four used datasets. Table 4 (Khan et al., 2021) lists the metrics category, name and the dataset. Using the datasets, we trained the SHO to serve as a classifier for defect prediction. Accuracy classification is compared with that of popular techniques of data mining in WEKA 3.6.

A confusion matrix (Sammut & Webb, 2011) is a kind of table that is commonly used to depict classification model performance. Such a matrix contains values of a predicted class and the associated actual class. Classifier results are determined using these values. One common confusion matrix is shown in Table 5.

A confusion matrix facilitates computing different metrics of measurements for data mining algorithms, like accuracy, precision, sensitivity, recall, specificity, and F-score. For example, accuracy (ACC) considers both true negative and positive overall instances. In other words, it is the ratio of all instances that are characterized effectively. It is calculated as follows:

$$\mathrm{ACC} = (\mathrm{TP}+\mathrm{TN})/(\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}) \tag{14}$$

Another parameter is specificity (SP; true negative rate). This measures the extent of actual negatives that are identified correctly. It is calculated as follows:

$$\mathrm{SP} = \mathrm{TN}/(\mathrm{TN}+\mathrm{FP}) \tag{15}$$

The next parameter is sensitivity (*S*). This is also called the probability of detection and recall. It measures the extent of actual positives that are identified effectively. It is calculated as follows:

$$S = \mathrm{TP}/(\mathrm{TP}+\mathrm{FN}) \tag{16}$$

Then, the precision (*P*) of the proposed classifier is calculated as follows:

$$P = \mathrm{TP}/(\mathrm{TP}+\mathrm{FP}) \tag{17}$$

TABLE 2 Dataset details of software defect prediction (Elsabagh et al., 2020)

Dataset	# attributes	Module	Defect	Defect (%)
KC1	22	2109	326	15.5
KC2	22	522	107	20.5
PC3	38	1563	160	10.23
JM1	22	10,885	2106	19.34

TABLE 3 Attribute details of software defect prediction for KC1, KC2, and JM1 datasets

Attributes	Description
Loc	McCabe's total number of lines
v(g)	'Cyclomatic measurements of complexity'
ev(g)	'Essential complexity'
iv(g)	McCabe 'complexity of design'
N	Halstead total operands + operators
V	'volume' Halstead
L	Halstead's 'program length'
D	'difficulty' Halstead
I	Halstead's 'intelligence'
E	'Measurement of effort' (Halstead)
B	Halstead's 'effort estimation'
t	'Time estimator'
lOCode	'Line count'
lOComment	number of comments lines Halstead
lOBlank	number of blank lines
lOCodeAndComment	number of comments and line of code
uniq_Op	quantity of one kind of operators
uniq_Opnd	quantity of one kind of operands
total_Op	quantity of operators
total_Opnd	quantity of operands
branchCount	quantity of the stream graph

F_measure (F_M) is computed as follows:

$$F\_M = (2 \times P \times S)/(P+S) \tag{18}$$

Finally, the average (W_A) of every estimation is calculated and weighted using the following equation:

$$W\_A\ of\ F\_M = ((F\_M_{c1} \times N_{c1}) + (F\_M_{c2} \times N_{c2}))/N_{c1+c2} \tag{19}$$

where F_M_{c1} and F_M_{c2} are the F_measures for Classes 1 and 2, respectively. In addition, N_{c1} and N_{c2} are the numbers of rows in Classes 1 and 2, respectively. N_{c1+c2} is the total number of rows. The weighted averages of the precision, sensitivity, recall, and specificity metrics can be calculated similarly to Equation (19).

The convergence rate (Cartis & Scheinberg, 2018) is another important performance measure of the proposed classifier. In evaluating iterative optimization algorithms, convergence has defined as the sequence of solutions obtained through the iterations until it converges to an optimum or satisfactory point. Less time is required to reach a good approximation if the convergence rate is high. This answers the question of how fast the algorithms reach their best solution and how steady this solution remains.

UQ (Abdar et al., 2021; Kompa et al., 2021) right now supports numerous basic predictions, and expectations made without UQ are ordinarily not dependable or precise. Uncertainty emerges for many reasons. Noise, data distribution, class overlapping, proposed model parameters and old data are reasons for uncertainty. These reasons divide into two types. The first presents from the dataset characteristics. We deal with ML repositories that contain standard datasets. These data may be incomplete, contain anomalous, not up-to-date, or even contradictory. These characteristics will certainly affect the model's final decision. The uncertainty inherited from these characteristics is called aleatoric uncertainty. We could enhance the missing values and remove anomalous and contradictory during preprocessing. Nevertheless, the data will still not up to date. In addition, the methods used during preprocessing are statistical (i.e., we will give the data metrics estimated values not the real ones). Therefore, this type of uncertainty is unavoidable.

The second type is epistemic uncertainty (known as model instability). In the prediction framework, we must find a suitable solution for preprocessing, training, and testing phases. In situations of large and inadequate datasets, we resort to AI-based methodologies to solve data problems in all phases. These AI methods are different from each other in parameters and processing. Which model is the best for the dataset under study? What are the appropriate values of the methodology parameters? Therefore, epistemic uncertainty arises (in other words the uncertainty of the classifier used in experiments; Abdar et al., 2021).

TABLE 4 Categorizing software engineering metrics for NASA datasets

Metrics		Datasets			
Category	**Name**	**JM1**	**KC1**	**KC2**	**PC3**
Halstead attributes	Halstead_content	*	—	—	*
	Halstead_difficulty	*	*	*	*
	Halstead_effort	*	*	*	*
	Halstead_error_estimator	*	—	—	*
	Halstead_length	*	*	*	*
	Halstead_level	*	*	*	*
	Halstead_program_time	*	*	*	*
	Halstead_volume	*	*	*	*
	Number_of_perands	*	*	*	*
	Number_of_operators	*	*	*	*
	Number_of_unique operands	*	*	*	*
	Number_of_unique operators	*	*	*	*
McCabe attributes	Essential_complexity	*	*	*	*
	Cyclomatic_complexity	*	*	*	*
	Design_complexity	*	*	*	*
	Cyclomatic_density	—	—	—	*
Size attributes	Number_of_lines	—	*	*	*
	LOC_total	*	*	*	*
	LOC_executable	*	—	—	*
	LOC_comments	*	*	*	*
	LOC_code_and_comments	*	*	*	*
	LOC_blank	*	*	*	*
Other attributes	Branch_count	*	*	*	*
	Condition_count	—	—	—	*
	EDGE_count	—	—	—	*
	Parameter_count	—	—	—	*
	Modified_condition_count	—	—	—	*
	Multiple_condition_count	—	—	—	*
	Node_count	—	—	—	*
	Design_density	—	—	—	*
	Essential_density	—	—	—	*
	Decision_count	—	—	—	*
	Decision_density	—	—	—	—
	Call_pairs	—	—	—	*
	Maintenance_severity	—	*	*	*
	Normalized_cyclomatic complexity	—	—	—	*
	Percent_comments	—	*	*	*
Class attribute	Defective	*	*	*	*

TABLE 5 General model of confusion

	Predicted	
Actual	**Non-defective**	**Defective**
Non-defective	True negative (TN)	False positive(FP)
Defective	False negative (FN)	True positive (TP)

During experiments, we are interested in epistemic uncertainty. We need to measure the proposed classifier uncertainty as well as accuracy.

There are three uncertainty quantification models namely Monte Carlo (MC; Kimaev et al., 2020), the Bootstrap model (Lai, 2020) and the GMM (Shafiullah et al., 2020).

Srivastav et al. (2013) used the GMM approach that fathoms the issue of assessing the joint probability density of the data under consideration in the maximum-likelihood. They integrated the uncertainty estimation task and prediction model as one modelling task. Inspiring by the same mechanism, we will take the output file predicted by the proposed SHO and fed it as input to the GMM workspace. During the experiment, the OriginPro (Seifert, 2014) is used to plot the GMM by maximum-likelihood 0.001 and maximum iteration 1000. The output file contains the values of software metrics used in prediction and the actual predicted class (defective and non-defective). Origin plots the GMM contours of the model output and the scatter plot of the data. If the scatter plot is well captured by the GMM contours, this will give an evidence of the proposed SHO classifier feasible performance in uncertainty quantification.

5.2 | Experimental results

During experiments, we implemented the SHO algorithm on random individuals of rules to select the optimal set. Each rule will contain conditional metrics and the class attribute. The MOF (explained in Section 4.1 Equation 13) will judge the optimal rules from the completely random population during the training. At the end of the training process, these individuals will represent the rule set forming the classification if-then rules.

To complete the performance measuring process, we used new test instances with unknown class labels (in other words only conditional metrics are input to the test process). The classification model (if-then classification rules or optimal rule set obtained from the training process) matches the conditional metrics of new instances with the rules. When the classification model finds the matching rule, it compares the actual class label of the rule with the target one. The numbers of true positives, true negatives, false positives, and false negatives for each dataset calculate the performance measures (accuracy, weighted average of precision, recall, specificity, and F-score) listed in Section 5.1 Equations 14–19. This section discusses the classification performance of the SHO is on NASA datasets, KC1, KC2, JM1, and PC3.

5.2.1 | Predict case 1: Dataset KC1

KC1 contains 22 attributes, 2109 instances, and 326 defects. Here, the proposed classifier is used for predicting software defects using the *MOF*. Most importantly, the confusion matrix describing the performance of the classifier on a random set of test data is calculated, as shown in Table 6. Then, a statistical analysis of performance, such as a weighted average of accuracy, recall, precision, sensitivity, specificity, and F-measure of the proposed algorithm, is conducted. Table 7 shows comparisons of the proposed classifier with other popular techniques, including ANNs, NB, SVMs, bagging, K-NN, random forest, PCA for reduction followed by ANN, and C4.5 using WEKA 3.6.

In Table 7, the values of the proposed classifier (SHO), such as the accuracy, weighted average of precision, recall, specificity, and *F*-score, are extracted from Table 6 using Equations (14–19). Accuracy measures the percentage of correctly classified test instances, while the weighted average of precision is estimated as the positive prediction overall testing instances. The recall is the proportion between true positive predictions to the total ones with regarding each real class separately. Specificity measures the extent of negatives that are accurately recognized. F-score depends on precision and recall measurement, and it considers the weighted average of them. F-score is valuable more than accuracy rate in a state of dissimilar dispersion of classification. Figures 3 and 4 show the results of the comparison for accuracy and precision. The comparisons indicate that the proposed classifier using SHO is the best in terms of accuracy and precision (85.2 and 85.3, respectively) for the KC1 dataset. The convergence of the proposed classifier using SHO determines the relationship between the number of iterations and the best scores (*MOF* values) of the KC1 dataset. As noted, the classifier converges after about 30% of the total number of iterations.

Figure 5 represents the confidence regions of the GMM contours and the scatter plot of the instances in the output KC1 test file from the proposed SHO. The output file contains the selected metrics by the model namely LOC, v(g), ev(g), and iv(g) along with the predicted class (defective or

TABLE 6 Confusion matrix using the proposed classifier

	Predicted	
Actual	**Non-defective**	**Defective**
Non-defective	706	2
Defective	123	12

TABLE 7 Comparative investigation of the proposed classifier with algorithms for KC1

Algorithm/weighted average	*P* (%)	*SP* (%)	*FPR* (%)	*S* (%)	*F_M* (%)	*ACC* (%)
ANN	81.5	43.7	56.3	83.1	82.1	83.1
ANN + PCA	80.8	36.4	63.6	83.6	81.6	83.6
NB	80.4	35.1	57.2	81.5	80.9	81.5
Bagging	81.5	32.4	67.6	84.7	81.7	84.7
K-NN	82.0	42.6	57.4	84.0	82.7	84.0
C4.5	81.3	32.3	67.7	84.6	81.6	84.6
Random forest	80.8	31.1	68.9	84.4	81.2	84.4
SVM	81.9	19.5	80.5	84.6	78.5	84.6
Proposed classifier	85.3	23.4	76.6	85.2	79.7	85.2

Abbreviation: SVM, support vector machine.

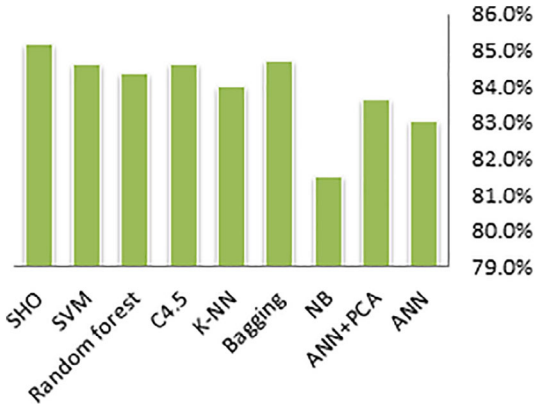

FIGURE 3 Comparative analysis in terms of accuracy for KC1

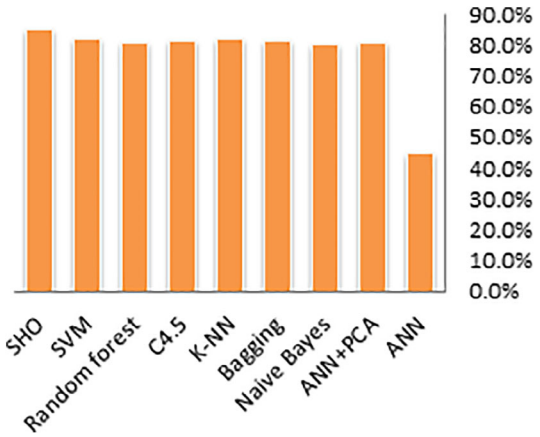

FIGURE 4 Comparative analysis in terms of weighted average precision for KC1

non-defective). There are 3D contour plots for 4-unique triples of the selected metrics in the KC2 data. The scatter plot of the KC2 test data is feasibly captured by the GMM contours except for few points. This is evidence for the adequate uncertainty performance of the proposed SHO.

5.2.2 | Predict case 2: Dataset JM1

JM1 contains 22 attributes, 10,885 instances, and 2106 defects. Table 8 shows the confusion matrix resulting from the testing of the proposed classifier.

Similarly, the confusion matrix for the JM1 dataset is shown in Table 8, which leads to Table 9. Figures 6 and 7 and Table 9 show the results of the statistical analysis described above in the subsection on the experimental setup. These results show that the proposed classifier using the

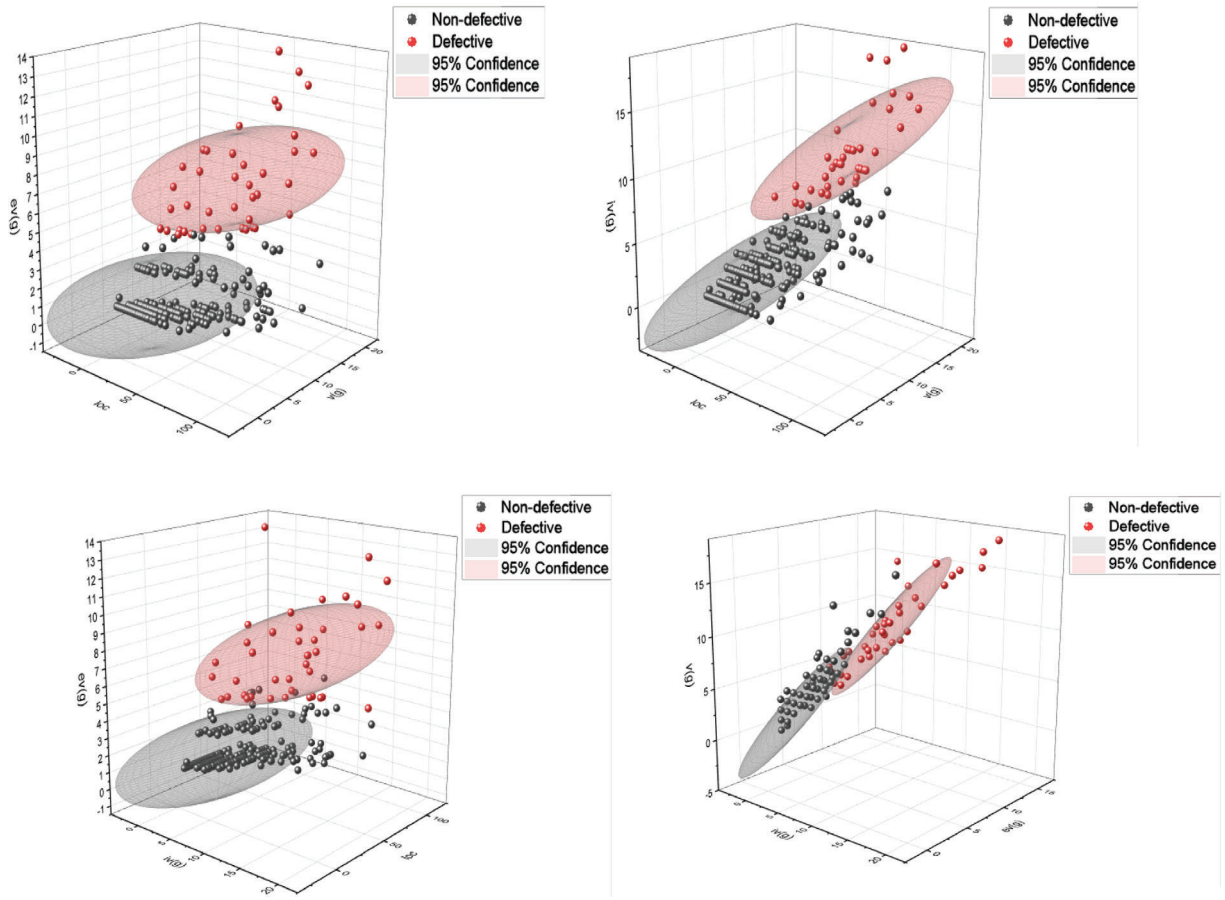

FIGURE 5 Contours of marginal distributions of spotted hyena optimizer metrics taken three at a time and scatter plots of KC1 data

TABLE 8 JM1 confusion matrix using the proposed classifier

	Predicted	
Actual	**Non-defective**	**Defective**
Non-defective	3554	0
Defective	793	8

TABLE 9 Comparative investigation of the proposed classifier with algorithms for JM1

Algorithm/weighted average	*P* (%)	*SP* (%)	*FPR* (%)	*S* (%)	*F_M* (%)	*ACC* (%)
ANN	76.1	23.0	77.0	81.2	74.4	81.2
ANN + PCA	77.3	31.1	68.9	81.3	77.1	81.3
NB	76.4	32.8	67.2	80.6	76.8	80.6
Bagging	77.6	32.5	67.5	81.3	77.2	81.3
K-NN	75.6	44.3	55.7	76.4	76.0	76.4
C4.5	75.5	36.3	63.7	79.2	76.6	79.2
Random forest	76.6	34.8	65.2	80.5	77.2	80.5
SVM	84.5	19.5	80.5	80.8	72.4	80.8
Proposed classifier	85.1	19.2	80.8	81.8	73.8	81.8

Abbreviation: SVM, support vector machine.

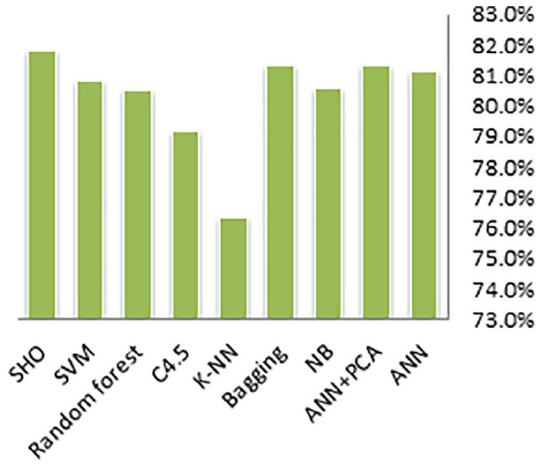

FIGURE 6 Comparative analysis in terms of accuracy for JM1

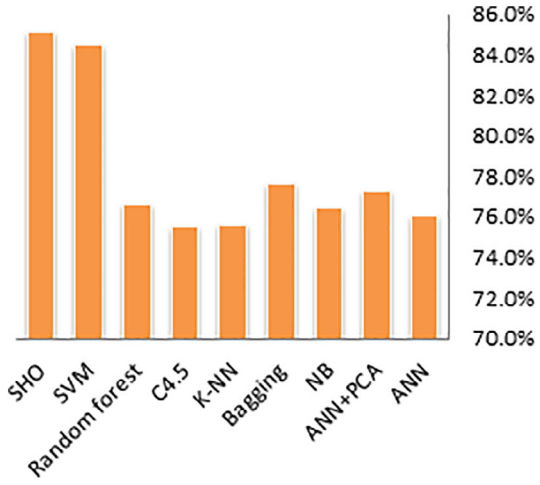

FIGURE 7 Comparative analysis in terms of weighted average precision for JM1

SHO algorithm performs the best in terms of accuracy for the JM1 dataset with scores of 81.8 and 85.1 for precision. The algorithms converge after approximately 4% of the total number of iterations and especially high rate.

Figure 8 represents the confidence regions of the GMM contours and the scatter plot of the instances in the output JM1 test file from the proposed SHO. The output file contains the selected metrics by the model namely LOC, v(g), ev(g), and iv(g) along with the predicted class (defective or non-defective). There are 3D contour plots for 4-unique triples of the selected metrics in the JM1 data. The scatter plot of the KC2 test data is estimated by the GMM contours except for few points. This shows that the output class predicted by the proposed model fits the PDF of the GMM model and is considered as a superior performance in uncertainty quantification.

5.2.3 | Predict case 3: Dataset PC3

PC3 contains 38 attributes, 1563 instances, and 160 defects, with 10.23% defective. The confusion matrix for this PC3 is shown in Table 10. The results in Table 11, Figures 9 and 10 shows that the SVM algorithm yields scores of 88.5 and 78.3 for accuracy and precision, respectively, better than those for other techniques, but the proposed classifier using the SHO algorithm performs best in terms of accuracy and precision for the PC3 dataset with scores of 89.6 and 94.8, respectively. Again, the algorithm converges after approximately 4% of the total number of iterations.

Figure 11 represents the confidence regions of the GMM contours and the scatter plot of the instances in the output PC3 test file from the proposed SHO. The output file contains the selected metrics by the model namely loc_comments, halstead_effort, halstead_prog_time,

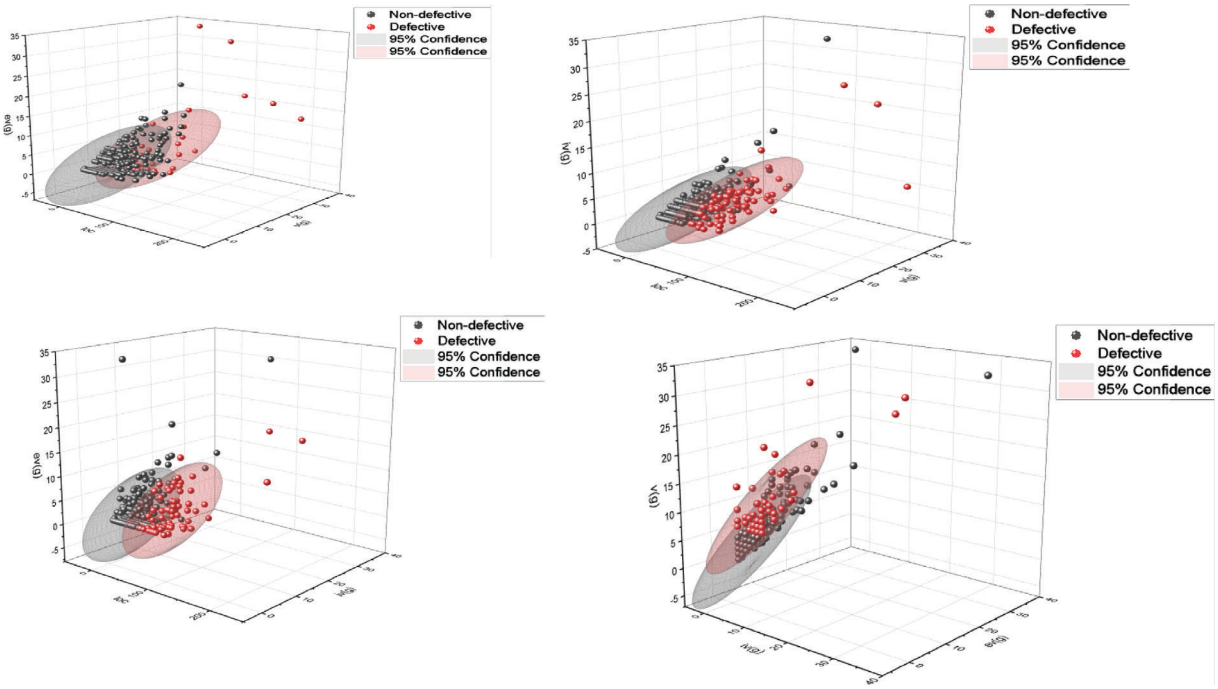

FIGURE 8 Contours of marginal distributions of spotted hyena optimizer metrics taken three at a time and scatter plots of JM1 data

TABLE 10 PC3 confusion matrix using the proposed classifier

	Predicted	
Actual	**Non-defective**	**Defective**
Non-defective	508	45
Defective	14	1

TABLE 11 Comparative investigation of the proposed classifier with algorithms for PC3

Algorithm/weighted average	*P* (%)	*SP* (%)	*FPR* (%)	*S* (%)	*F_M* (%)	*ACC* (%)
ANN	84.4	31.8	68.2	86.7	85.3	86.7
ANN + PCA	84.6	24.8	75.3	88.0	85.3	88.0
NB	87.3	84.1	15.9	42.6	49.8	42.6
Bagging	84.6	23.6	76.4	88.2	85.3	88.2
K-NN	84.7	36.6	63.4	85.9	85.2	85.9
C4.5	84.1	27.1	72.9	87.2	85.1	87.2
Random forest	82.5	18.6	81.4	87.4	84.0	87.4
SVM	78.3	11.5	88.5	88.5	83.1	88.5
Proposed classifier	94.8	8.9	91.1	89.6	92.1	89.6

Abbreviation: SVM, support vector machine.

and num_operators along with the predicted class (defective or non-defective). There are 3D contour plots for 4-unique triples of the selected metrics in the PC3 data. The scatter plot of the PC3 test data is fitted well to the GMM contours except for few points. This shows that the output class predicted by the proposed model fit the PDF of the GMM model and is considered as a reliable performance in uncertainty quantification.

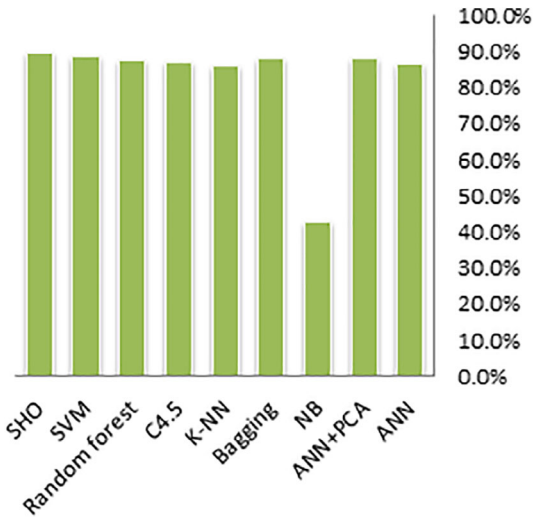

FIGURE 9 Comparative analysis in terms of accuracy for PC3

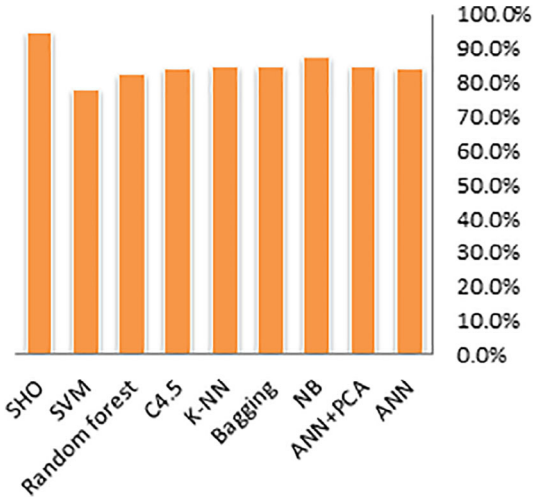

FIGURE 10 Comparative analysis in terms of weighted average precision for PC3

5.2.4 | Predict case 4: KC2 Dataset

KC2 contains 22 attributes, 522 instances, and 107 defects with 20.5% defective. The confusion matrix for KC2 is shown in Table 12.

Table 13, Figures 12 and 13 show the statistical analysis and results with the proposed algorithm yielding accuracy and precision scores of 84.0 and 84.5, respectively, better than those of ANN, which scores 83.6 and 82.1, respectively. The convergence of SHO resulting from Case 4 for the KC2 dataset occurs after approximately 4% of the total number of iterations.

Figure 14 represents the confidence regions of the GMM contours and the scatter plot of the instances in the output KC2 test file from the proposed SHO. The output file contains the selected metrics by the model namely LOC, v(g), ev(g), and iv(g) along with the predicted class (defective or non-defective). There are 3D contour plots for 4-unique triples of the selected metrics in the KC2 data. The scatter plot of the KC2 test data is fitted well to the GMM contours except for few points. This shows that the output class predicted by the proposed model fits the PDF of the GMM model and is considered as a good performance in uncertainty quantification.

The standard deviation is a good measure for algorithm stability. The low standard deviation shows that the values tend to be near the average, whereas the high standard deviation shows that the values spread with different values. Therefore, the low standard deviation of the accuracy scores proves that algorithms reach approximately the same accuracy in every run during the experiments. In evaluating the algorithms, if there is an equality of accuracy values, the standard deviation can assist in breaking the equality. A lower standard deviation indicates a more stable classifier. Table 14 presents the standard deviation of accuracy scores for each algorithm on 15 runs. As the SHO classifier shows the lowest deviation in accuracy score, we can say it is a feasible solution for the SDP problem.

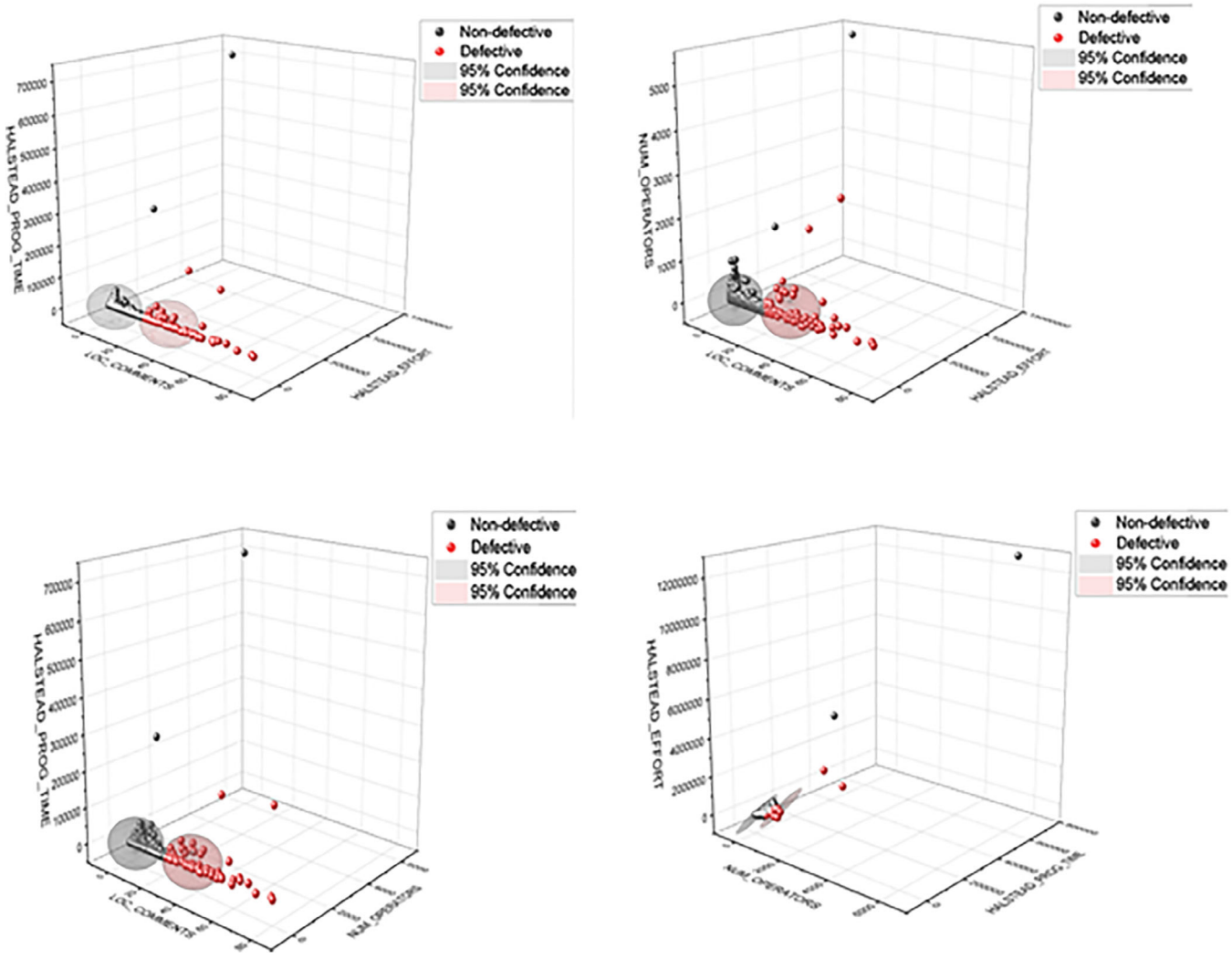

FIGURE 11 Contours of marginal distributions of spotted hyena optimizer metrics taken three at a time and scatter plots of PC3 data

TABLE 12 KC2 confusion matrix using the proposed classifier

	Predicted	
Actual	**Non-defective**	**Defective**
Non-defective	119	1
Defective	23	7

5.3 | Experimental discussion

As per the experimental results, the meta-heuristic SHO algorithm is applied to four different datasets, KC1, KC2, JM1, and PC3 to find the fittest classification model (standard rules) among individuals. This experiment used the SHO as a classifier to improve SDP for with-in projects. The proposed system begins the search process with a population of random if-then rules. Everyone is composed of conditional and decision parts. The conditional part (antecedent) is the software metrics explained in Section 5.1. The decision is whether the rule label defective or non-defective. The random sets of rules are organized afterward by the fitness function, which keeps only the good individuals. Then, the SHO algorithm makes others follow the best ones by exploration and exploitation processes. Support and confidence were used as the MOF during the experiments to locate the fittest rules. The support equation finds the rules, which cover the largest number of instances where the confidence measures the ratio between the instances that satisfy the whole rule to those which satisfy the antecedent or conditional part of the rule. The MOF finds the optimal rules, which match the largest number of dataset instances.

TABLE 13 Comparative investigation of the proposed classifier with algorithms for KC2

Algorithm/weighted average	*P* (%)	*SP* (%)	*FPR* (%)	*S* (%)	*F_M* (%)	*ACC* (%)
ANN	82.0	42.1	57.9	83.6	80.7	83.6
ANN + PCA	78.2	37.3	62.7	81.4	78.0	81.4
NB	81.1	66.3	33.7	76.5	78.1	76.5
Bagging	65.6	19.0	81.0	81.0	72.5	81.0
K-NN	76.4	38.8	61.2	79.5	77.4	79.5
C4.5	65.6	19.0	81.0	81.0	72.5	81.0
Random forest	65.6	19.0	81.0	81.0	72.5	81.0
SVM	76.5	36.9	63.1	80.0	77.4	80.0
Proposed classifier	84.5	38.5	61.5	84.0	80.0	84.0

Abbreviation: SVM, support vector machine.

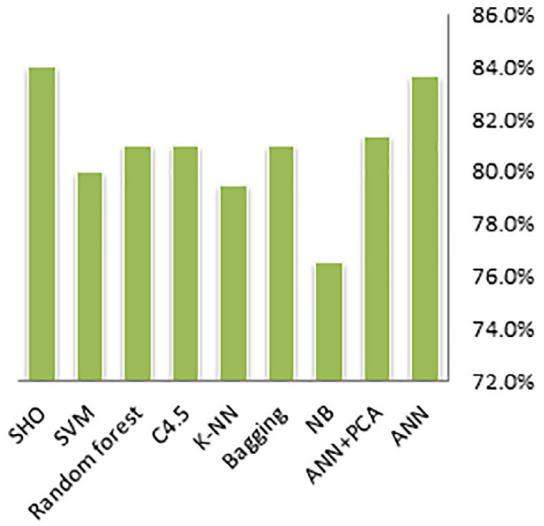

FIGURE 12 Comparative analysis in terms of accuracy for KC2

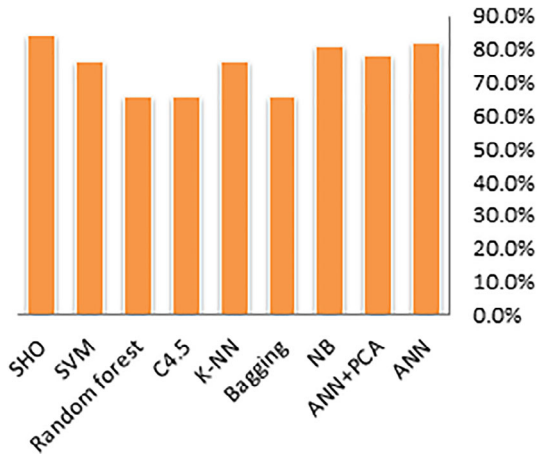

FIGURE 13 Comparative analysis in terms of weighted average precision for KC2

The experimental study showed that the proposed algorithm performs better than popular techniques of data mining in terms of accuracy score 85.2, 81.8, 89.6, and 84.0 and precision score 85.3, 85.1, 94.8, and 84.5 for KC1, JM1, PC3, and KC2, respectively. Moreover, the standard deviation is computed for the SHO algorithm and compared with the popular data mining techniques mentioned in the literature. The experiment

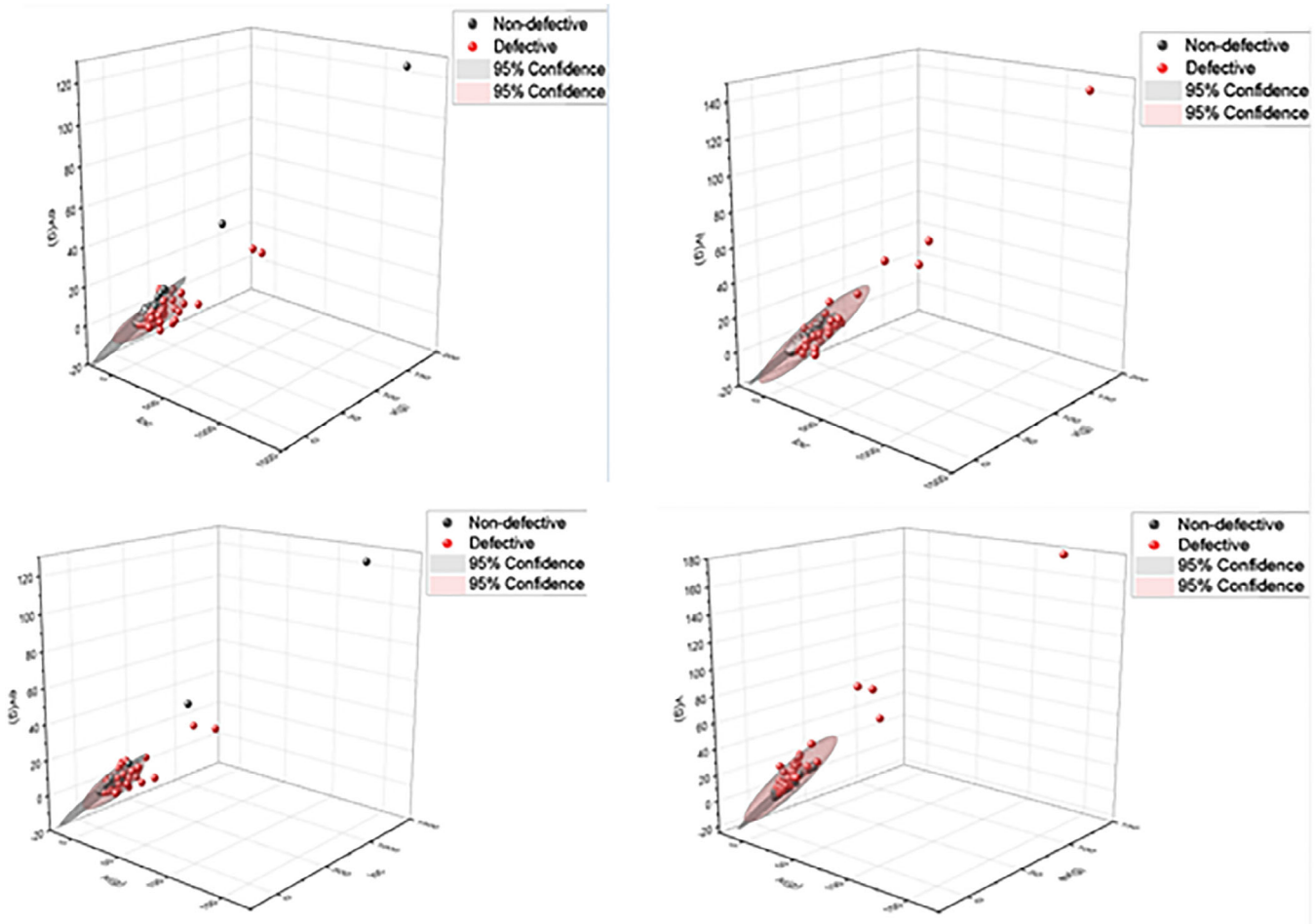

FIGURE 14 Contours of marginal distributions of spotted hyena optimizer metrics taken three at a time and scatter plots of KC2 data

TABLE 14 Standard deviation according to the accuracy

	Datasets					
	KC1		JM1		KC2	
Algorithms	JM1	KC2	KC1	KC2	KC1	JM1
ANN	0.056	0.079	0.071	0.089	0.073	0.040
ANN + PCA	0.054	0.079	0.072	0.088	0.078	0.041
NB	0.064	0.082	0.082	0.088	0.084	0.043
Bagging	0.063	0.080	0.071	0.084	0.080	0.035
K-NN	0.075	0.083	0.091	0.091	0.092	0.053
C4.5	0.073	0.081	0.084	0.081	0.084	0.037
Random forest	0.058	0.080	0.073	0.082	0.083	0.052
SVM	0.061	0.078	0.071	0.079	0.061	0.041
Proposed classifier	0.041	0.080	0.071	0.081	0.051	0.032

Abbreviation: SVM, support vector machine.

shows that the SHO is the highest in accuracy and precision and the lowest in standard deviation. Therefore, the SHO meta-heuristic algorithm proved its ability as a feasible and stable classifier.

Through experiments, the proposed classifier reaches its optimum solution through the first set of iterations and keeps steady on the optimum solution during the remainder of the iterations. For example, for JM1, PC3, and KC2, the algorithm reaches its optimum solution after at

most 4% of the total number iterations, while for KC1 the proposed algorithm converges after 17 iterations (approximately 30%). This is a relatively fast rate of convergence.

In the uncertainty quantification study of the SHO, the GMM approach is used to estimate the joint probability density of data distribution in the output test file. The contours drew by the GMM adequately captured the scatter plot of the test file containing the selected most important metrics and the predicted class. This is evidence of a feasible uncertainty performance of the SHO classifier.

According to our experiments, the SHO proved its ability to find optimal classification rules in a relatively fast convergence time and better than the mentioned literature studies. Nevertheless, we think the accuracy rate and precision should be increased by enhancing the quality of the datasets used for the study. Moreover, the testing process is a very overly problem and up till now the prediction model could guide the test team to find the defects, but we need a complete automated defect prediction process.

6 | CONCLUSIONS AND FUTURE WORK

Various techniques were proposed here for improving SDP. The challenge of classification accuracy arises in connection with predicting many datasets. This study used the SHO algorithm to improve SDP. This new meta-heuristic algorithm was used as a classifier for defect prediction within projects. Support and confidence were used as the MOF during the experiments to locate the fittest rules. An experimental study showed that the proposed algorithm performs better than popular techniques of data mining in terms of accuracy and precision with a mean of 85.2 and 87.4, respectively. In addition, other measurements were determined for the proposed classifier and contrasted with those of several other algorithms. These comparisons demonstrated that the proposed classifier can be used to predict software defects efficiently.

The proposed system gives a binary decision, either defective or non-defective, but it is known that there is an area of ambiguity between the two decisions. This is because the software metrics can have unstable or conflicting values. The system should absorb these light changes in values and work accordingly. Hence, this calls for attention to uncertainty in software defection decisions. Future work should include an upgrading of the SDP systems to cover uncertainty fields like fuzzy (Zadeh & Aliev, 2018) and rough (Riaz et al., 2019) theories. These fields give more space for metrics values by degrees of membership, lower and upper approximations. Moreover, the SDP can include exploring the application of a feature reduction methodology before training the classifier to ensure that only the key features are concentrated on. Testing this classifier for its ability to determine defect types would also be of interest.

ACKNOWLEDGEMENT

This work is not funded and is submitted for obtaining a master's degree in Software Engineering.

CONFLICT OF INTEREST

The authors declare there is no conflict of interest.

DATA AVAILABILITY STATEMENT

Research data are not shared.

ORCID

Mahmoud A. Elsabagh https://orcid.org/0000-0001-8704-3887

Mona G. Gafar https://orcid.org/0000-0001-7592-840X

REFERENCES

Abdar, M., Acharya, U. R., Sarrafzadegan, N., & Makarenkov, V. (2019). NE-nu-SVC: A new nested ensemble clinical decision support system for effective diagnosis of coronary artery disease. *IEEE Access*, *7*, 167605–167620.

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., & Nahavandia, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion.*, *76*, 243–297.

Alaia, E. B., Harbaoui, I., Borne, P., & Bouchriha, H. (2018). A comparative study of the PSO and GA for the m-MDPDPTW. *International Journal of Computers Communications & Control*, *13*(1), 8–23.

Aljamaan, H. I., & Elish, M. O. (2009). An empirical study of bagging and boosting ensembles for identifying faulty classes in object-oriented software. *IEEE Symposium on Computational Intelligence and Data Mining*, *2009*, 187–194.

Bansal, J. C. (2019). Particle swarm optimization. In J. C. Bansal (ed.), *Evolutionary and swarm intelligence algorithms* (pp. 11–23). Springer.

Cartis, C., & Scheinberg, K. (2018). Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Mathematical Programming*, *169*(2), 337–375.

Dam, H. K., Pham, T., Ng, S. W., Tran, T., Grundy, J., Ghose, A., Kim, T., & Kim, C.-J. (2018). A deep tree-based model for software defect prediction. *ArXiv*, *1802*, 00921.

Das, S., & Saha, P. (2021). Performance of swarm intelligence based chaotic meta-heuristic algorithms in civil structural health monitoring. *Measurement*, *169*, 108533.

Deb, K. (2014). Multi-objective optimization. In K. Deb (ed.), *Search methodologies* (pp. 403–449). Springer.

Dhiman, G., & Kumar, V. (2017). Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*, *114*, 48–70.

Eid, H. F., Garcia-Hernandez, L., & Abraham, A. (2021). Spiral water cycle algorithm for solving multi-objective optimization and truss optimization problems. *Engineering with Computers*, *2021*, 1–11.

Elsabagh, M. A., Farhan, M. S., & Gafar, M. G. (2020). Cross-projects software defect prediction using spotted hyena optimizer algorithm. *SN Applied Sciences*, *2*(4), 538. https://doi.org/10.1007/s42452-020-2320-4

Gandomi, A. H., Yang, X.-S., Talatahari, S., & Alavi, A. H. (2013). Metaheuristic algorithms in modeling and optimization. *Metaheuristic Applications in Structures and Infrastructures*, *2013*, 1–24.

Hammouri, A., Hammad, M., Alnabhan, M., & Alsarayrah, F. (2018). Software bug prediction using machine learning approach. *International Journal of Advanced Computer Science and Applications*, *9*(2), 78–83.

Hosseinabadi, A. A. R., Vahidi, J., Saemi, B., Sangaiah, A. K., & Elhoseny, M. (2019). Extended genetic algorithm for solving open-shop scheduling problem. *Soft Computing*, *23*(13), 5099–5116.

Jayanthi, R., & Florence, L. (2019). Software defect prediction techniques using metrics based on neural network classifier. *Cluster Computing*, *22*(1), 77–88.

Kaur, R., & Sharma, E. S. (2018). Various techniques to detect and predict faults in software system: Survey. *International Journal of Future Revolution in Computer Science & Communication Engineering (IJFRSCE)*, *4*(2), 330–336.

Khan, B., Naseem, R., Shah, M. A., Wakil, K., Khan, A., Uddin, M. I., & Mahmoud, M. (2021). Software defect prediction for healthcare big data: An empirical evaluation of machine learning techniques. *Journal of Healthcare Engineering*, *2021*, 1–16.

Khan, M. Z. (2020). Hybrid ensemble learning technique for software defect prediction. *International Journal of Modern Education and Computer Science*, *12*(1), 1–10.

Kimaev, G., Chaffart, D., & Ricardez-Sandoval, L. A. (2020). Multilevel Monte Carlo applied for uncertainty quantification in stochastic multiscale systems. *AICHE Journal*, *66*(8), e16262.

Kompa, B., Snoek, J., & Beam, A. L. (2021). Second opinion needed: Communicating uncertainty in medical machine learning. *NPJ Digital Medicine*, *4*(1), 1–6.

Koru, A. G., & Liu, H. (2005). Building effective defect-prediction models in practice. *IEEE Software*, *22*(6), 23–29.

Kumar, V., Chhabra, J. K., & Kumar, D. (2014). Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems. *Journal of Computational Science*, *5*(2), 144–155.

Kumar, V., & Kaur, A. (2020). Binary spotted hyena optimizer and its application to feature selection. *Journal of Ambient Intelligence and Humanized Computing*, *11*(7), 2625–2645.

Kuncheva, L. I., Skurichina, M., & Duin, R. P. W. (2002). An experimental study on diversity for bagging and boosting with linear classifiers. *Information Fusion*, *3*(4), 245–258.

Lai, M. H. C. (2020). Bootstrap confidence intervals for multilevel standardized effect size. *Multivariate Behavioral Research*, *2020*, 1–21.

MATLAB. (2016). version 7.10.0 (R2016a). Natick, Massachusetts: The MathWorks Inc.

Mierswa, I, & Klinkenberg, R. (2018). RapidMiner Studio (9.1) [Data science, machine learning, predictive analytics]. Retrieved from https://rapidminer.com/

Nam, J. (2014). *Survey on software defect prediction*. Department of Compter Science and Engineerning, The Hong Kong University of Science and Technology, Techcnical Report.

Okutan, A., & Yıldız, O. T. (2014). Software defect prediction using Bayesian networks. *Empirical Software Engineering*, *19*(1), 154–181.

Pławiak Paweł and Abdar, M., Pławiak, J., Makarenkov, V., & Acharya, U. R. (2020). DGHNL: A new deep genetic hierarchical network of learners for prediction of credit scoring. *Information Sciences*, *516*, 401–418.

Pourpanah, F., Lim, C. P., & Saleh, J. M. (2016). A hybrid model of fuzzy ARTMAP and genetic algorithm for data classification and rule extraction. *Expert Systems with Applications*, *49*, 74–85.

Qodmanan, H. R., Nasiri, M., & Minaei-Bidgoli, B. (2011). Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence. *Expert Systems with Applications*, *38*(1), 288–298.

Raukas, H. (n.d.). Some approaches for software defect prediction.

Ren, J., Qin, K., Ma, Y., & Luo, G. (2014). On software defect prediction using machine learning. *Journal of Applied Mathematics*, *2014*, 2014.

Riaz, M., Davvaz, B., Firdous, A., & Fakhar, A. (2019). Novel concepts of soft rough set topology with applications. *Journal of Intelligent \& Fuzzy Systems*, *36*(4), 3579–3590.

Sammut, C., & Webb, G. I. (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.

Seifert, E. (2014). *OriginPro 9.1: Scientific data analysis and graphing software software review*. ACS Publications.

Shafiullah, D. S., Vergara, P. P., Haque, A., Nguyen, P. H., & Pemen, A. J. M. (2020). Gaussian mixture based uncertainty modeling to optimize energy management of heterogeneous building neighborhoods: A case study of a Dutch university medical campus. *Energy and Buildings*, *224*, 110150.

Shan, C., Chen, B., Hu, C., Xue, J., & Li, N. (2014). *Software defect prediction model based on LLE and SVM*.

Singh, P. D., & Chug, A. (2017). *Software defect prediction analysis using machine learning algorithms* (pp. 775–781). 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence.

Software Defect Dataset: OpenML (n.d.). Retrieved from https://www.openml.org/search?type=data

Software Defect Dataset: Promise repository. (n.d.). Retrieved from http://promise.site.uottawa.ca/SERepository/dataetspage.html

Srivastav, A., Tewari, A., & Dong, B. (2013). Baseline building energy modeling and localized uncertainty quantification using Gaussian mixture models. *Energy and Buildings*, *65*, 438–447. https://doi.org/10.1016/j.enbuild.2013.05.037

Wang, S., Rao, R. V., Chen, P., Zhang, Y., Liu, A., & Wei, L. (2017). Abnormal breast detection in mammogram images by feed-forward neural network trained by Jaya algorithm. *Fundamenta Informaticae*, *151*(1–4), 191–211.

Wei, H., Hu, C., Chen, S., Xue, Y., & Zhang, Q. (2019). Establishing a software defect prediction model via effective dimension reduction. *Information Sciences*, *477*, 399–409.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.
Xu, Z., Xuan, J., Liu, J., & Cui, X. (2016). *Michac: Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering* (pp. 370–381). 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER).
Zadeh, L. A., & Aliev, R. A. (2018). *Fuzzy logic theory and applications: Part I and part II*. World Scientific Publishing.
Zhang, Y., Wang, S., Ji, G., & Phillips, P. (2014). Fruit classification using computer vision and feedforward neural network. *Journal of Food Engineering*, *143*, 167–177.

AUTHOR BIOGRAPHIES

Mahmoud M. Elsabag: B. Sc. "Excellent with Honors" Graduation Project: Smart Home, software Engineer at Information and Development center, Kafrelsheikh University. Working as a Teaching Assistant at Software Engineering Department, at Faculty of Computers and Information, Kafrelsheikh University, Egypt. Now Working as a Teaching Assistant at Machine Learning and Information Retrieval Department, at Faculty of Artificial Intelligence, Kafrelsheikh University, Egypt E-mail Addresses: Mahmoud_Mohsen@fci.kfs.edu.eg.

Marwa S. Farhan holds a Ph.D. in Information Systems. She is an associate professor, information systems department, Faculty of Informatics and Computer science, British university in Egypt, Egypt. Her research interest focus on Big data and data analytics, Advanced Database Management, and software engineering.

Mona G. Gafar received the B.Sc., M.Sc., and Ph.D. degrees from the Faculty of Computers and Information, Mansoura University, in 2006, 2010, and 2014, respectively. She is currently an Assistant Professor with the Department of Computer Science, College of Science and Humanities in Al-Sulail, Prince Sattam bin Abdulaziz University, Saudi Arabia. In addition, she is an associate professor with the Department of Machine Learning and Information Retrieval Department, Faculty of Artificial Intelligence, Kafrelsheikh University, Egypt. Her research interests involve artificial intelligence, data mining, software engineering and applications of modern optimization techniques.

How to cite this article: Elsabagh, M. A., Farhan, M. S., & Gafar, M. G. (2021). Meta-heuristic optimization algorithm for predicting software defects. *Expert Systems*, e12768. https://doi.org/10.1111/exsy.12768