



International Journal of Computational Science and Engineering

ISSN online: 1742-7193 - ISSN print: 1742-7185

<https://www.inderscience.com/ijcse>

Feature selection using Tasmanian devil optimisation algorithm for software fault prediction

Himansu Das, Muskan, Hrishikesh Kumar

DOI: [10.1504/IJCSE.2025.10068003](https://doi.org/10.1504/IJCSE.2025.10068003)

Article History:

Received:	12 August 2023
Last revised:	21 January 2024
Accepted:	23 January 2024
Published online:	21 December 2024

Feature selection using Tasmanian devil optimisation algorithm for software fault prediction

Himansu Das*, Muskan and Hrishikesh Kumar

School of Computer Engineering,
KIIT University,
Bhubaneswar, India
Email: das.himansu2007@gmail.com
Email: muskansona999@gmail.com
Email: hrishikeshkumar09@gmail.com

*Corresponding author

Abstract: Software quality is improved through early software flaw identification, which is made possible by software fault prediction (SFP). Early software problem identification reduces the amount of money and work needed to fix those vulnerabilities. This paper presents a novel FSTDO model, a feature selection (FS) approach built upon Tasmanian devil optimisation (TDO). The proposed FSTDO helps to select the optimal feature subset that yields maximum classification accuracy. The proposed FSTDO algorithm is juxtaposed with four well-known FS algorithms, such as FS using ACO, FS using GA, FS using PSO and FS using DE. Four popular classifiers KNN, NB, DT, and QDA, were used with the FSTDO to assess the quality of its outcomes. The experimental outcomes reveal that the proposed FSTDO has more potential in comparison with the competing FS models as it delivers enhanced performance in terms of accuracy, and the selection of optimal features is also significantly viable.

Keywords: software fault prediction; SFP; feature selection; optimisation; Tasmanian devil optimisation; TDO; machine learning; classifiers; ant colony optimisation; ACO; particle swarm optimisation; PSO; K-nearest neighbour; KNN.

Reference to this paper should be made as follows: Das, H., Muskan and Kumar, H. (2025) 'Feature selection using Tasmanian devil optimisation algorithm for software fault prediction', *Int. J. Computational Science and Engineering*, Vol. 28, No. 1, pp.32–55.

Biographical notes: Himansu Das works as an Associate Professor in the School of Computer Engineering, KIIT University, Bhubaneswar, India. He has received his PhD in Engineering degree from the Veer Surendra Sai University of Technology (VSSUT), Odisha. His research interests include the field of data mining, soft computing, and machine learning.

Muskan has received his BTech from the KIIT University, Bhubaneswar, India. His research interests include the fields of machine learning, computer vision and artificial intelligence.

Hrishikesh Kumar has received his BTech from the KIIT University, Bhubaneswar, Odisha, India. His research interests include the fields of machine learning, computer vision and artificial intelligence.

1 Introduction

Software fault prediction (SFP) (Catal and Diri, 2009) is a significant and critical activity during the early phases of software development for boosting software quality and lowering maintenance effort. Early defect detection can lead to faster problem resolution and the delivery of maintainable software. If the issue is discovered after the development process has begun, the developers will incur more costs and effort since they will have to redesign some of the software modules. SFP enhances software quality by anticipating faults using past data in the form of datasets.

FS (Das et al., 2021, 2022; Padhi et al., 2022; Li et al., 2017) is a technique for using only relevant data to diminish the number of variables fed to a machine learning (ML)

model (classifier) (Jordan and Mitchell, 2015; Mothilal et al., 2020). It is one of the most important tasks in pre-processing to overcome classification (Das et al., 2020a, 2020b) difficulties. FS reduces noise in the data and improves the classifiers' predicted performance. It enables us to choose only the most helpful characteristics from the complete dataset for the ML model. Types of FS include Wrapper and Filter based methods (Wah et al., 2018). In filter-based FS algorithms, the procedure of choosing the features is independent of the ML algorithm. The features are chosen using this strategy based on how well they perform in various statistical tests to determine whether they are correlated with the result variable. This is a major limitation of the filter-based approach as the usefulness of

the selected feature subset in training the classifier is not taken into consideration. However, in the wrapper-based approach of FS, 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. So, wrapper-based FS is used in the presented work as they are useful for providing greater performance since they employ the target classifier as the FS technique.

The technique of selecting the optimal result among numerous possible solutions for a problem statement in terms of its constraints is known as optimisation (Simon, 2013). Science and technological advancements have resulted in the introduction of new and difficult optimisation issues, together with newly acquired information about past optimisation difficulties. Undivulged search space, on-derivative objective functions, search space that's discontinuous, large dimensions, and non-convexity are all characteristics of many of these issues. As a result, conventional approaches and mathematical analysis have failed to tackle real-world optimisation issues successfully. As a result, researchers and scientists frequently offer novel optimisation solution methods known as metaheuristic algorithms (Abdel-Basset et al., 2018). In order to address optimisation issues, it is built on randomised searcher agents and employs arbitrary operators, without the requirement for derivative knowledge. Because of features such as the ability to avoid local optimisation, rapid execution, stellar results, and adaptability to optimisation issues in various domains, metaheuristic algorithms have grown in popularity over conventional approaches.

Metaheuristic algorithms have stochasticity characteristics that solve optimisation problems through methodologies inspired by natural occurrences such as physics laws, biological sciences, game rules, and different processes of evolution. These kinds of algorithms first provide plausible solutions based on the problem constraints. They then enhance these possible solutions iteratively based on the algorithm update phases. Two crucial characteristics of the metaheuristic algorithms to search and find a solution are exploration and exploitation. Exploration means conducting a global search in the problem-solving space for analysing the various areas of the solution space and avoids becoming trapped in the optimal local area, whereas exploitation refers to the idea of performing a local search in the vicinity of the discovered solution in order to discover a better one. To achieve acceptable performance, metaheuristic algorithms must strike an appropriate balance between exploitation and exploration. As metaheuristic algorithms use a random search method, the answer they propose may not be the global ideal. As a result, the solution produced by metaheuristic algorithms is referred to as a quasi-optimal solution. Because of their unpredictability and diverse nature in the search process, the performance of metaheuristic algorithms varies. As a result, metaheuristic algorithms may provide several solutions to a given optimisation issue. As a result, researchers have been

developing several metaheuristic algorithms to give improved quasi-optimal solutions to optimisation problems.

Based on their major inspiration, metaheuristic algorithms are categorised into four types inspired by: evolution, physics, swarm, and games. The development of evolutionary-based algorithms has been driven by the integration of biological sciences with the concepts of natural selection and Darwin's theory of evolution. Genetic algorithm (GA) (Holland, 1992) and differential evolution (DE) (Karaboğa and Ökdem, 2004) can be considered as the most renowned evolutionary algorithms. The random operators of mutation, selection, and crossover play a crucial role in the design of GA and DE in updating the population. Natural behaviour of numerous animal species, aquatic animals, birds, and other living creatures opened the path for the creation of swarm-based algorithms. The most well-known and commonly used swarm-based algorithms are PSO (Kennedy and Eberhart, 1995) and ant colony optimisation (ACO) (Al Salami, 2009). PSO has imitated animal behaviour such as swarming fish or birds. In determining the shortest path, whereas ACO has been effectively replicating the ants' natural behaviour. In nature, animals' hunting strategies constitute an optimising process.

The main research topic in any metaheuristic technique study is if it is still required to suggest new metaheuristic algorithms to more successfully address optimisation challenges in light of the multiple techniques that have been established. According to researchers' analysis of the literature, the simulation of the behaviour of Tasmanian demons had not previously been used in the development of any metaheuristic algorithm. The Tasmanian devil's natural consumption patterns point to an optimisation process in achieving this species' primary objective, namely, food supply. This inspired the researchers to create a novel optimiser by replicating its feeding strategy leading to the birth of Tasmanian devil optimisation (TDO) algorithm (Dehghani et al., 2022). Within this work, a novel wrapper-based FS method utilising the TDO algorithm is proposed.

The motivation for proposing the FSTDO model stems from the fact that the performance of all previously available FS algorithms, such as FSGA (Shahamat and Pouyan, 2015), FSDE (Khushaba et al., 2009), FSACO (Deriche, 2009), and FSPSO (Ahmad, 2015), is entirely reliant on the settings of their hyperparameters. So, a new algorithm that is less dependent on its hyperparameters for performance and outperforms the previous algorithms is proposed. For the experiments, a combination of FSTDO with K-nearest neighbour (KNN) classifier (Yigit, 2013), naive Bayes (NB) classifier (Rish, 2001), decision tree (DT) classifier (Safavian and Landgrebe, 1991) and QDA classifier (Tharwat, 2016) were employed. In this paper, the fitness value, (i.e., error) was used as an assessment criterion.

The findings indicate that applying TDO for FS enhanced the accuracy of all four classifiers for the SFP datasets. The FS using optimisation results demonstrate the strong ability of FSTDO in exploration and exploitation, as

well as the creation of a correct balance between these two indicators to successfully solve FS issues. FSGA, FSPSO, FSDE, and FSACO were used to assess the calibre of the FSTD0 results. In all the simulation studies, similar hyperparameter settings were used for FSTD0 and all the other competing FS algorithms. The simulation output showcases a higher capability of the proposed FSTD0 methodology than the four competitor FS algorithms and is a lot productive. Compared to the competing four FS algorithms, FSTD0 gave higher accuracy of classification and was better in selecting the optimal feature subset in majority of test cases. The experimental results were also statistically analysed using Friedman Test and Holm test to validate their statistical significance. The main contributions of this work are stated below:

- 1 A new wrapper-based FS model named FSTD0 is proposed.
- 2 SFP is carried out utilising four well-known classifiers – KNN, QDA, NB, and DT – on 12 open-source software fault datasets.
- 3 The experimental results of FSTD0 have been compared with four well-known FS models such as FSDE, FSACO, FSPSO, and FSGA.
- 4 To confirm the statistical validity of the experimental data, a statistical analysis utilising the Holm and Friedman test was carried out.

The remainder of this paper is laid out in the following format: Section 2 acknowledges the literature survey or past related research works, followed by Section 3 which introduces a TDO-based FS methodology and its implementation, Section 4 displays the experiment and computational results, Section 5 walks us through statistical analysis and discussions and Section 6 gives a conclusion to the work.

2 Literature review

This particular segment describes earlier works and studies conducted for numerous wrapper-based FS (Leng et al., 2010) techniques over the last couple decades. Because of the existence of a significant amount of noisy, irrelevant, and redundant features in high-dimensional data, the presence of many features impairs classification performance (Osisanwo et al., 2017). In implementation of FS, the attributes are provided in an easier way to speed up the classifiers' learning process. Every FS approach's design should take into account a number of variables, including the search strategy and evaluation technique for wrapper methods.

Another type of FS approach, filter-based FS (Porkodi, 2014), uses statistical criteria to choose the features. Due to the relevance of the feature, the top rank of the characteristics is often chosen for analysis. Choosing the features based on the established statistical criteria, a variety of filter-based FS approaches were developed, including

similarity-based methods, information-based approaches, along with statistical-based approaches. However, it does not take the ML algorithms (Mahesh, 2020) into account and the usefulness of the selected features in training the ML algorithms is not evaluated. Due to the aforementioned factors, filter-based techniques performed well in terms of speed and computational cost. On the other hand, the wrapper-based FS techniques have been constructed on ML algorithms for learning to assess the quality of features chosen using these parameters:

- a it seeks out the most pertinent subset of features.
- b it computes the model fitness for the selected features.

In contrast to filter techniques, the computational cost of the issue grows in this method since learning of a model with a fresh collection of characteristics each time depending on the fitness value. The model is then put to the test using the test set, and the error is calculated. As the wrapper-based techniques take the ML algorithms into account for selecting the features, it gives better accuracy than the filter-based approaches and hence the wrapper-based FS approaches are used in this paper. In learning algorithms, the methods which relies on wrapper procedure is used to compute the efficacy of selecting optimal features because selecting such ideal features is a bit time-consuming operation. To pick the most pertinent subset of features, it employs a search process that employs several optimisation techniques. The search method can either be linear or random. To find the best features, the mechanism which searches sequentially can add or delete features sequentially in the feature field. However, because it adjoins or removes features in sequential sequence, it may capture in local optima. In random search, randomness is incorporated into the search process to automatically diversify it from the local optima. Several optimisation algorithms, including ACO, PSO, DE and GA are being used as a random search method to identify best attribute subset.

FS methods that are wrapper-based are yet a mystery among scholars, since several of them are always working to build new FS methods utilising various optimisation strategies. Many surveys have been conducted on various FS approaches and their importance in the categorisation process. These studies examined the many methodologies used in FS methods initially as well as the problems with its own set of goals accompanied by obstacles. The FS methodology can be either supervised or unsupervised. Class levels were taken into account in the procedure for making decisions in the supervised FS method scenario to choose optimum features. The following constitute a few examples of this technique applied in literary works. The first evolutionary method which is most frequently utilised for FS to select the best features based on the course of natural evolution is GA (Kumar et al., 2010 Mirjalili and Mirjalili, 2019). It adopted an approach based on local search to discover the connection between the selected group of defining features. A binary encoded GA was suggested by Lu et al. (2008) to determine the optimal characteristics for picture annotation. Some other previous

research conducted on FS using GA is given in Kaya and Pehlivan (2015), Yoshida et al. (2001) and Yildiz and Doğru (2019).

The biggest difficulty in the GA is premature convergence or being caught in a local optimum owing to the starting weight. To overcome this issue, another population-based optimisation method PSO (Fernández Martínez and García Gonzalo, 2008) came into picture paving way for PSO-based FS (Tu et al., 2007) approaches which used several co-evolution PSO to determine the most suited features. Can et al. (2013) proposed a new model for detecting software failures called P-SVM model using PSO and SVM. This model took benefit of nonlinear computing ability of SVM and optimisation of parameter of PSO. It adopted an optimised SVM for SFP and used the PSO technique to determine the optimal parameter for the SVM. Zheng and Wang (2018) introduced a method called FS-JMIE where JMIE (a metric) searches feature subset performance. He then applies binary particle swarm optimisation (BPSO) (Khanesar et al., 2007) for subset searching. Shehu et al. (2021) proposed a time-based PSO variant. In order to accomplish this, they created a time-constant in the update function of velocity. It helped them to prevent early convergence, especially in the dataset of videos' emotional frames. This PSO variant is compared with the two standard PSO algorithm. This helped them categorise emotion from (CK+) dataset.

Similarly, a DE-based FS method (Khushaba et al., 2011) was suggested, accompanied by a statistical repair mechanism, to select the optimal feature subset in datasets having varying dimensionality. Recommendation to use DE-based FS method in intrusion detection system for choosing the optimal features was made by Almasoudy et al. (2020). Al-Ani et al. (2013) proposed a novel FS method which utilises DE in a new way for identifying pertinent feature subsets. They used a technique that includes distributing the features among a group of wheels in an effort to narrow the search space. In the same manner, another strategy termed FSACO approach (Al-Ani, 2005) has been addressed to pick the best features. Performance is increased by utilising a number of unsupervised FS methods. By considering the significance of specific characteristics in each iteration, an unsupervised FS proposition based on ACO (Dadaneh et al., 2016) had been created to determine the subset that is pertinent optimal features. The similarity index between the characteristics is used to reduce these redundant feature sets. In order to assess if two features are comparable, it used inter-feature validity, which efficiently eliminates feature redundancy. An instance-based classification approach called pattern matching (PMC) was developed by Sreeja and Sankar (2015). By matching the pattern in the training dataset, PMC's primary principle picks the unlabelled samples. To increase the classification performance of PMC, they suggested an FS algorithm based on ACO. Saraç and Özel (2014) suggested an ACO approach for classification of web pages. Categorisation of text using ACO was discussed by Aghdam et al. (2008). Despite the fact that the majority

of the dimensions are useless, the main issue with text categorisation is its great dimensionality. They used classifier and feature subset length as heuristic data for ACO. It eliminated the need for any prior information in this way. Unsupervised FS approach employing ACO (UFSACO) was proposed by Tabakhi et al. (2014). Through multiple iterations, their strategy attempted to determine the ideal feature subset without employing a learning algorithm. This method could be used with high dimensional datasets in various applications (Long et al., 2023; Diao et al., 2023; Liang et al., 2022, 2023; Zhang et al., 2023; Xu et al., 2021) due to its low computational complexity.

All of the aforementioned techniques, whether supervised or unsupervised FS approaches, have drawbacks that cause the model's performance to degrade. All of the approaches presented thus far seek to maximise a combination of FS cost, classification error rate, also called as cost of misclassification. A major limitation of all the discussed FS methods is that they heavily depend on their hyper parameter values for their performance. If their hyper parameter values are not adjusted properly, it will result in low performance by these models. So, to overcome the aforementioned issue, a unique FS technique is designed using a new optimisation algorithm called TDO to pick the most acceptable features while eliminating unnecessary, repetitive, and noisy features.

3 Methodology

3.1 Problem statement: feature selection

FS is a method for employing only pertinent data to use fewer input variables (features) into our ML model. It is among the most important duties in classification process. FS reduces data noise and improves the classifiers' predicted performance. It permits to narrow down the whole dataset's properties to just those that would benefit the ML model.

FS is a technique for narrowing down a big feature collection to a reasonable subset of attributes to improve classification model performance. It merely selects the characteristics that are important for making decisions. It generates an efficient subset of features by getting rid of redundant or noisy information in order to lessen the processing cost of the problem. This optimum feature subset selection is an NP-hard problem (Tindell et al., 1992) that cannot be solved in polynomial time. A thorough search is necessary to find the best appropriate subset of the features, which is an NP-hard problem. With an increase in the number of characteristics, this search method leading to the continuous growth of the solutions in solution space. It is critical to choose the best answer from the available options, which is a difficult process. It eventually improves the performance of FS issues. The classification process productivity ought to be enhanced through the generation of the optimal feature subset.

A general block diagram of FS using an optimisation algorithm is presented in Figure 1. This optimum FS consists of four phases, which are as follows:

- Features subset is created.
- Comparison and evaluation of fitness applying the aforementioned subsets of attributes.
- Check for satisfaction of the termination criteria, alternatively, reprise steps one and two.
- Assessment of conclusions using an ideal selection of traits. Finally, optimal feature subset is given as output.

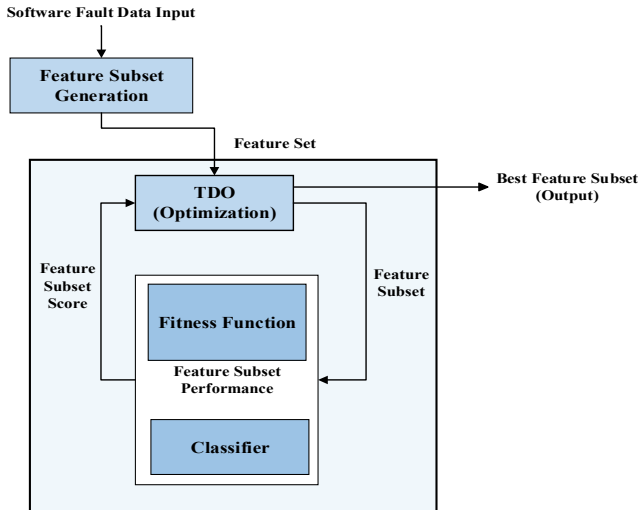
The dataset is split into two portions, known as the train and test sets. The optimisation method is fed the training data in order to discover the most suited optimal features. The classification technique assesses the model's performance using the best features of the train and test data. The FS process can be mathematically represented using equation (1). In equation (1), the aim is to minimise the error and, in every iteration, the error gets reduced. As a result, the classification accuracy becomes higher.

$$Lu(Y) = \min Error(k) \quad (1)$$

Here, $Y = |K|$, where $k \subseteq K$.

Here, K is the original feature set and k is its subset. k will optimise the objective function $Lu(Y)$.

Figure 1 Block diagram of FS using optimisation algorithm (see online version for colours)



As already discussed, FS is used to pick the required features and reduce irrelevant features thus decreasing the effects related to the curse of dimensionality hence increasing efficiency. It can be categorised into:

- Filter method – it uses a statistical computation to compute the relevancy of the predictors which are outside of the predictive models and retain only the predictors that pass certain criteria.
- Wrapper method – it evaluates several models using technique which adds or remove predictors to get the required combination of features that will maximise the

effectiveness of the FS model. It trains the ML model using subset of features and iteratively draws inferences from the previously trained model and decides whether to add or remove features from the subset.

The following are the primary distinctions between the filter and wrapper approaches for FS: Wrapper techniques assess the utility of a subset of characteristics by iteratively training a model that depends on it, whereas filter strategies evaluate the utility of a feature set based on how it correlates with the dependent variable.

FS encoding: the binary encoding of the features is the basis on which the features in this work are chosen. Each column of the dataset is a potential feature and the feature subset is selected from the set of all the columns (features). All the features in a feature subset or solution of a population are represented by either 0 or 1. A value of 0 implies that the associated attribute will not be included in the feature subset whereas a value of 1 indicates that it will. Figure 2 displays an illustration of the binary encoding of eight features. According to Figure 2, the 1st, 3rd, 4th, 7th, and 8th features will be included in the feature subset and the rest of the characteristics will not be selected.

Figure 2 Binary representation of features

Features	1	0	1	1	0	0	1	1
Index	1	2	3	4	5	6	7	8

3.2 TDO algorithm

TDO is a new optimisation technique that executes on the modelling of the natural behaviour showcased by Tasmanian devil (TD). It is a carnivorous and marsupial wild animal which is a member of the Dasyuridae family (Miller et al., 2011) and native to Tasmania. The fundamental source of motivation for TDO is the TD's eating operation in two techniques: live prey hunting and flesh-eating that's utilised to address various science optimisation challenges. They are opportunistic predators that, while capable of hunting food, prefer to feed on carrion if it is available. They have two eating techniques. In the first method, the TD feeds on carrion. The subsequent method entails murdering the subject in order to hunt and consume it. Just as the goal of the optimisation process is to get a suitable answer, the goal of this creature's feeding process is to find a food source.

The two key principles of TD are exploration and exploitation, which are applied in this method to thoroughly examine the problem-solving space and find the best solution. The foraging behaviour of this creature in looking for food in various locations actually represents an exploratory indicator in the optimisation process to investigate optimal regions of foraging space. In contrast to this, its prey tracking process, which takes place only in a specific area, resembles the exploitation index in local searches aimed at converging on optimal solutions. It's suggestive of creating a mathematical model of the method this animal employs to locate food sources that tends to

create optimisers that produce ideal answers to the optimisation challenge.

The detailed explanation of the proposed FSTDO including mathematical modelling, pseudo code of the model and the flowchart is showcased in the upcoming sections.

3.3 Proposed FSTDO

Controlling factors for population-based algorithms include population size, generation number, mutation rate, crossover rate, social and cognitive characteristics, and starting amount. To acquire the best outcome, these parameters must be correctly set. These algorithm-specific parameters are critical for determining the efficacy of the optimisation method; otherwise, bad tuning of these parameters may trap in a local optimum and raise the computing cost of the issue. To address the above difficulties, the TDO model is utilised. TDO can be combined with FS approach to create the best feature subset to boost classification performance. This suggested FSTDO technique is detailed below.

3.3.1 Initialisation

FSTDO is a stochastic algorithm that is population-based with TD as searcher agents. The primary population of TD is produced at random based on the task restrictions. Elements of the FSTDO group that are adept at addressing problems, depending upon where they are in the search area, searchers input potential values for the issue variables. As a result, every individual in the population is represented by a vector that has the same number of elements as the variables in the problem. Conclusively, a matrix in equation (2) can be used to model the set of FSTDO members.

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_i \\ \vdots \\ Y_N \end{bmatrix}_{N \times M} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,j} & \cdots & y_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{i,1} & \cdots & y_{i,j} & \cdots & y_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,j} & \cdots & y_{N,m} \end{bmatrix}_{N \times M} \quad (2)$$

Here,

Y population of TD

N number of TD

Y_i i^{th} candidate solution

M number of variables in the supplied issue

$y_{i,j}$ candidate value of TD for j^{th} variable.

By entering the variable values in the objective function (fitness function), the objective function values of the problem may be calculated. As a consequence, a vector is used to model the acquired values of the fitness function represented using equation (3).

$$G = \begin{bmatrix} G_1 \\ \vdots \\ G_i \\ \vdots \\ G_N \end{bmatrix}_{N \times 1} = \begin{bmatrix} G(Y_1) \\ \vdots \\ G(Y_i) \\ \vdots \\ G(Y_N) \end{bmatrix}_{N \times 1} \quad (3)$$

Here,

G vector belonging to objective function.

G_i value of objective function acquired by i^{th} candidate solution.

Fitness function: in this work, the fitness function is actually an equation used for calculating the prediction error of the classifier. The fitness function value (fitness score) for an individual is determined by contrasting the projected output with the actual output of the testing data (the testing class label). The fitness score is calculated for each candidate solution by using equation (4).

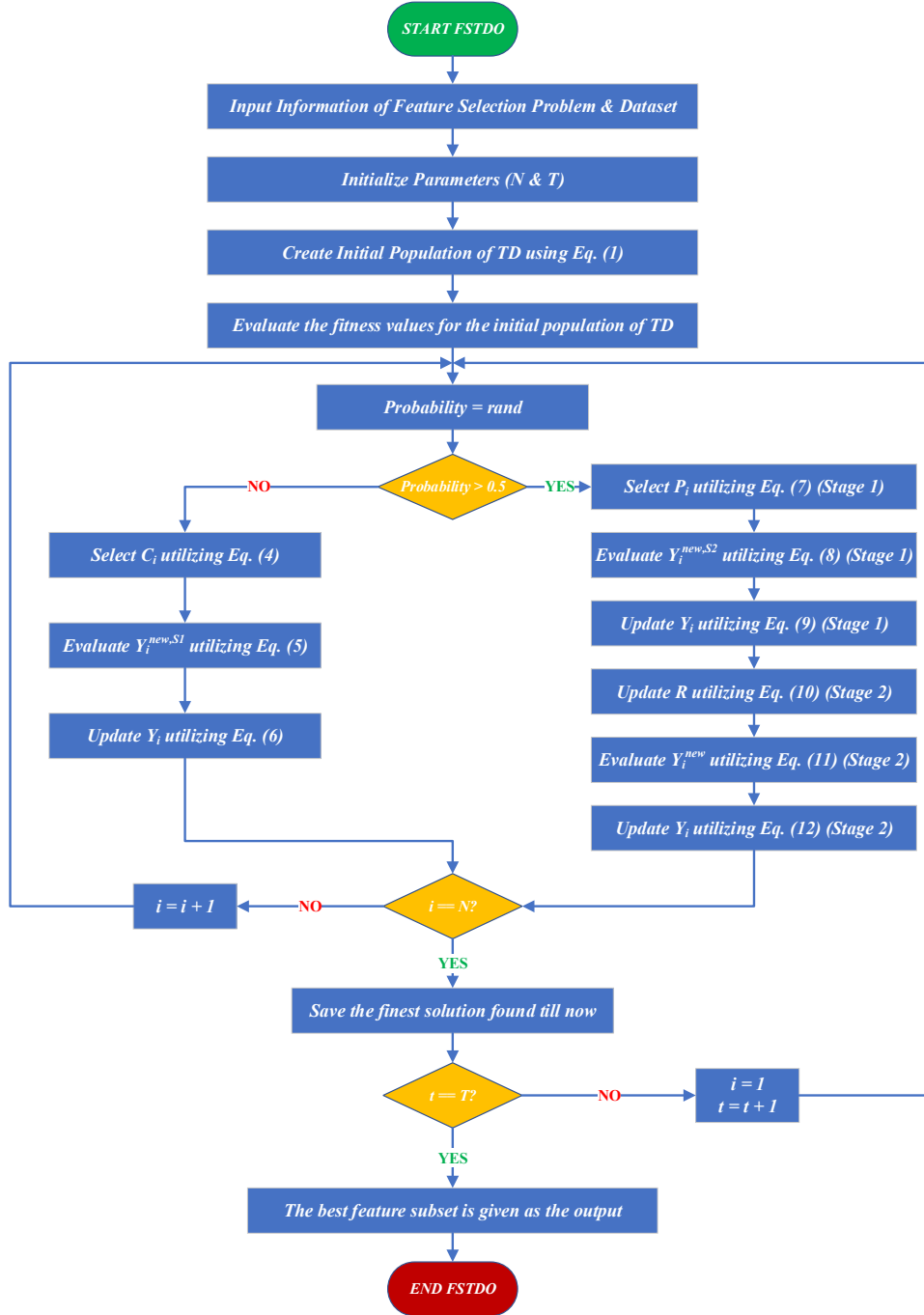
$$G(Y_i) = \frac{\sum \text{Error}}{\text{Pr}} \quad (4)$$

Here, *Error* is the number of faulty predictions and *Pr* represents total number of predictions. In order to get the best feature subset, the algorithm aims to minimise this objective function value.

An illustration of the fitness calculation is provided in Table 1. The fitness values in the table are calculated using equation (4). It can be seen from the table that the 1st individual has a fitness score of 0.429 and the 2nd individual has a fitness score of 0.286. So, the 2nd individual is a better solution as it has a lower fitness score.

Table 1 An illustration of fitness calculation

	Projected output	Actual output	Fitness using equation (4)
1	Y	Y	Fitness = 3/7 = 0.429
	Y	Y	
	N	Y	
	Y	N	
	N	N	
	N	Y	
	Y	Y	
Number of faulty predictions = 3			Total predictions = 7
2	Y	Y	Fitness = 2/7 = 0.286
	N	Y	
	Y	Y	
	Y	N	
	N	N	
	Y	Y	
	Y	Y	
Number of faulty predictions = 2			Total predictions = 7

Figure 3 Flowchart of FSTD0 (see online version for colours)**Algorithm 1** Pseudo-code for FSTD0

```

1  Input: Enter details of the FS problem.
2  Initialise the parameters: number of iterations (T) and population size (N).
3  Initialisation of TD positions with some random values (0 or 1) and evaluate the fitness function.
4  for  $t = 1$  to  $T$ 
5      for  $i = 1$  to  $N$ 
6          if probability < 0.5, probability = rand
7              Strategy 1: (exploration phase): Feeding on carrion
8                  Choose carrion for the  $i^{\text{th}}$  TD using equation (5).
9                  Determine the new status (position) of the TD using equation (6).
10                 Use the transfer function for binary conversion of the new position using equation (14) and equation (15).
  
```

```

11      | Update the  $i^{\text{th}}$  TD with the help of equation (7).
12      | else
13      | Strategy 2 (exploitation phase): Feeding on prey
14      |   Stage 1: Prey selection and attack
15      |   With the  $i^{\text{th}}$  TD, choose prey using equation (8).
16      |   Determine the new status (position) of the TD using equation (9).
17      |   Use the transfer function for binary conversion of the new position using equation (14) and equation (15).
18      |   Update the  $i^{\text{th}}$  TD using equation (10).
19      |   Stage 2: Pursuit of prey
20      |   Update the radius of the neighbourhood using equation (11).
21      |   Determine the new status of the  $i^{\text{th}}$  TD in the vicinity of Y using equation (12).
22      |   Use the transfer function for binary conversion of the new position using equation (14) and equation (15).
23      |   Use this to update the  $i^{\text{th}}$  TD with the help of equation (13).
24      | end if
25      | end for  $i = 1$  to  $N$ 
26      | Conserve the finest solution offered up to this point.
27      | end for  $t = 1$  to  $T$ 
28      | Output: FSTDO's optimal feature subset to the specified FS issue.

```

Examining the metrics generated from the objective function allows one to assess the possible solutions. The population's most qualified candidate solution is the one that yields the minimum value for the objective function. At each iteration, the superior individual of the population is updated depending on current values. FSTDO's population update algorithm is based on two TD feeding techniques. Each of the TD can devour carrion or hunt for prey. FSTDO assumes that the likelihood of selecting any of these tactics is equal to 50%. According to this theory, each TD is updated in each iteration of the FSTDO based on only one of these two tactics mentioned below.

3.3.2 Strategy 1: exploration phase

TD occasionally chooses to eat local carrion than hunting. Around the TD live other raptors who hunt enormous prey but are unable to consume it all. Unless the TD shows around, these creatures might not be able to consume enough of their prey. The TD loves to eat this carrion in these situations. The activity of a TD searching its habitat for carrion resembles an algorithm search procedure in a problem-solving environment. The TD tactic truly exemplifies the potential of FSTDO exploration in identifying the initial ideal location by scanning several regions of the search field.

The notions contained in the TD carcass-eating method are mathematically represented using equations (5), (6) and (7). The location of the remainder of the population in the search region is presumed to represent carrion places in the FSTDO design for each TD. Equation (5) simulates a random selection of one of these conditions to let the l^{th} population member chosen as the target carrion for the i^{th} TD. As a result, l must be picked at random from 1 to N .

$$\begin{aligned}
 C_i &= Y_l \\
 i &= 1, 2, 3, \dots, N \\
 l &\in \{1, 2, 3, \dots, N \mid l \neq i\}
 \end{aligned} \tag{5}$$

Here, C_i represents the carrion which got chosen by i^{th} TD.

A new position in the search space is determined for the TD relying on the selected carrion. In this strategy's TD motion simulation, if the carrion's objective function value is higher, the TD goes towards it; otherwise, it travels away from it. This TD moving method may be seen in equation (6). After computing the fresh spot for TD, provided the result of the objective function is superior at this updated position, it is embraced in the last step of the first tactic; otherwise, TD stays in its previous place that is depicted in equation (7).

$$y_{i,j}^{\text{new},S1} = \begin{cases} y_{i,j} + r \times (c_{i,j} - I \times y_{i,j}), & G_{c_i} < G_i \\ y_{i,j} + r \times (y_{i,j} - c_{i,j}), & \text{otherwise} \end{cases} \tag{6}$$

$$Y_i = \begin{cases} Y_i^{\text{new},S1}, & G_i^{\text{new},S1} < G_i \\ Y_i, & \text{otherwise} \end{cases} \tag{7}$$

Here

$Y_i^{\text{new},S1}$ i^{th} TD's new status depending on the first strategy.

$y_{i,j}^{\text{new},S1}$ value of j^{th} variable for i^{th} TD.

$G_i^{\text{new},S1}$ value of objective function for i^{th} TD.

G_{c_i} i^{th} TD's objective function value for the chosen carrion.

r random floating number in the interval $[0, 1]$.

I either 1 or 2.

3.3.3 Strategy 2: exploitation phase

Hunting and consuming prey is the TD's second method of feeding. There are two distinct phases to the TD's actions during the assault. It searches the area for the prey in the first phase before attacking it. The subsequent step has it chasing down the target in order to halt it and begin devouring. The modelling of the initial phase is comparable to the modelling of the initial strategy, i.e., carcass selection. Equations (8) to (10), then, depict the initial phase of prey selection and attack. In the second method, the location of additional population members is assumed as prey location while updating the i^{th} TD. The l^{th} member of the population is chosen at random as prey, where l is a random natural integer between 1 to N and not equal to i . The process of selection of prey is reproduced in equation (8).

$$\begin{aligned} P_i &= Y_l \\ i &= 1, 2, 3, \dots, N \\ l &\in \{1, 2, 3, \dots, N \mid l \neq i\} \end{aligned} \quad (8)$$

In this instance, P_i is the TD's chosen victim.

Once the location of the prey has been identified, a new position for the TD is chosen. If the fitness function value of the chosen prey is higher while calculating this new position, the TD goes closer to it; otherwise, it travels farther away from it. This process' modelling is shown in equation (9). The previous position for the TD is replaced if the new position makes the value of the target function better. The second strategy's phase is depicted in equation (10).

$$y_{i,j}^{\text{new},S2} = \begin{cases} y_{i,j} + r \times (p_{i,j} - X \times y_{i,j}), & G_{P_i} < G_i \\ y_{i,j} + r \times (y_{i,j} - p_{i,j}), & \text{otherwise} \end{cases} \quad (9)$$

$$Y_i = \begin{cases} Y_i^{\text{new},S2}, & G_i^{\text{new},S2} < G_i \\ Y_i, & \text{otherwise} \end{cases} \quad (10)$$

Here,

$Y_i^{\text{new},S2}$ i^{th} TD's new status depending on the second strategy.

$y_{i,j}^{\text{new},S2}$ value of j^{th} variable for i^{th} TD.

$G_i^{\text{new},S2}$ value of objective function for i^{th} TD.

G_{P_i} i^{th} TD's value of fitness function of the selected prey.

r any random floating number in the interval of $[0, 1]$.

X random number, either 1 or 2.

The second stage and the imitation of chasing after prey are the key distinctions between the second and the first strategy. The local search inside the search space is akin to the pursuit of prey near to the assault area. This TD behaviour shows how the FSTDO may be used to get access

to superior candidate solutions. The TD pursues the victim in the area of the attacked area to mimic this pursuit procedure. The FSTDO uses equation (11) to equation (13) to imitate the prey chasing stage. At this point, the range that the TD tracks its prey may be determined using the radius of this neighbourhood as in equation (11). In light of the chasing activity in the neighbourhood, a new position for the TD may be determined as simulated in equation (12). If the newly computed position gives the objective function a better value than its old position, the TD will accept it. The TD's location is updated in equation (13).

$$R = 0.01 \left(\frac{1-t}{M} \right) \quad (11)$$

$$y_{i,j}^{\text{new}} = y_{i,j} + (2r-1) \times R \times y_{i,j} \quad (12)$$

$$Y_i = \begin{cases} Y_i^{\text{new}}, & G_i^{\text{new}} < G \\ Y_i, & \text{otherwise} \end{cases} \quad (13)$$

Here,

R radius of the attack point

M the max number of iterations

Y_i^{new} new status of i^{th} TD in Y_i locality

$y_{i,j}^{\text{new}}$ value of i^{th} TD for j^{th} variable

G_i^{new} objective function value for i^{th} TD.

t iteration counter.

The first iteration of the algorithm concludes after all FSTDO members have been updated. The location of TDs and the goal function are given new values. Following this, the algorithm moves on to the following iteration, and the FSTDO population update procedure goes on till the method's iterations are complete, in accordance with equations (5) till (13). Over these cycles, FSTDO updates and saves the top candidate solution. When the algorithm has been fully applied, FSTDO presents the top potential solution as the answer to the problem.

The transfer function (TF) is needed after every such equation mentioned above where the continuous valued position (or newly calculated position) is obtained and hence that position needs to be converted into its corresponding binary value as binary values can only be used while calculating the fitness by using equation (14) and (15). The TF is used after equation (6), equation (9) and equation (12) to convert the newly calculated positions of the TDs to their corresponding binary values.

$$\text{TransferFunction} = \frac{1}{1 + \exp((-10) \cdot (M_{i,r}^{P1} - 0.5))} \quad (14)$$

$$M_{i,r}^{P1} = \begin{cases} 1, & \text{if TransferFunction} > \text{rand} \\ 0, & \text{else} \end{cases} \quad (15)$$

Here,

$r = 1, 2, 3, \dots, D$

rand random floating number between 0 and 1.

TransferFunction probability of $M_{i,r}^{Pl}$ (continuous value of the r^{th} feature).

The value of $M_{i,r}^{Pl}$ is updated to its binary form using its probability. If *TransferFunction* is greater than *rand*, then the value is updated to 1. Else the value is updated to 0.

3.3.4 Output

Finally, after a stopping criterion is reached, the FSTDO algorithm gives the optimal subset of features as output in the form of indices of those features. The working of the FSTDO is explained using the algorithm and flowchart as presented in Algorithm 1 and Figure 3 respectively.

4 Result analysis

4.1 Datasets used

The proposed FSTDO's performance indicators have been examined and validated using 12 open-source National Aeronautics and Space Administration (NASA) software defects datasets from the PROMISE repository (Boetticher, 2007): JM1, CM1, KC1, MC1, MW1, PC1, KC3, PC3, MC2, PC2, PC4 and PC5. Each of the 12 software defect datasets represents a NASA software system or a subsystem. Each row represents a software module (which can refer to a function) and contains static code metrics or quality metrics in the form of attributes or features (columns). The last column is the output/target class. The values in the target class are in the form of 'Y' or 'N' where, 'Y' represents that the software module is defective and 'N' represents that the software module is not defective. Table 2 has a thorough description of these datasets. Table 3 represents the number of instances and features in each of the datasets.

Few reasons for selecting these particular datasets are:

- Software engineering PROMISE Repository information is accessible to the general population in an effort to promote researchers to take advantage of them for experimental purposes for producing improved software engineering prediction models.
- Any ML-based prediction model will produce unsatisfactory results using the raw PROMISE repository datasets since the majority of it has many redundant, extraneous and noisy features. So, performing FS on these datasets is necessary and it improves the accuracy by a great extent for SFP.

For the purpose of evaluating fitness (error), these datasets were randomly split into training and testing groups of 80% and 20%, respectively.

4.2 Experimental condition

The experiments were carried out with Python 3.9.10 in Jupyter Notebook 6.4.7. The system hardware specifications were: an NVIDIA MX130 GPU, an Intel Core i5-8250U processor with a frequency of 1.80 GHz, and 16 GB of RAM.

Table 2 Datasets details used in the experiment

Sl. no.	Dataset	Programming language	Description of the software system/subsystem
1	KC1	C++	Management of storage to receive and analyse ground data
2	CM1	C	Instrument of spacecraft
3	PC2	C	Dynamic altitude control systems simulator
4	JM1	C	Ground-based real-time forecasting system
5	MC1	C and C++	An experiment of the combustion of a space shuttle
6	PC1	C	An earth orbiting satellite's flight software
7	MW1	C	A combustion experiment in zero gravity
8	MC2	C	A visual navigation system
9	PC4	C	An earth orbiting satellite's flight software
10	PC3	C	An earth orbiting satellite's flight software
11	KC3	JAVA	Management of storage for ground data
12	PC5	C++	System to boost cockpit security

Table 3 Datasets with number of instances and features

S no.	No. of datasets	No. of instances	No. of features
1	JM1	7,782	22
2	CM1	327	38
3	KC1	1,183	22
4	MC1	1,988	39
5	PC4	1,287	38
6	MW1	253	38
7	PC1	705	38
8	MC2	125	40
9	PC2	745	37
10	KC3	194	40
11	PC3	1,077	38
12	PC5	1,711	39

In the context of ML, hyperparameters are the configurations and settings of a ML algorithm that must be established prior to training the model on a dataset. Choosing the best hyperparameters may boost the

performance of the resulting model significantly. The act of picking the optimum hyperparameters is known as hyperparameter tuning, and it may be accomplished using an array of strategies such as grid search, random search, and Bayesian optimisation. Below mentioned is the list of all the hyperparameters utilised in the algorithms (FSGA, FSDE, FSPSO, FSACO and FSTDO) in a tabular format (Table 4).

On the basis of Table 4, it is observed that two hyperparameters are constant in all the FS models, i.e., population size (30) and total iterations (200), to achieve most optimal overall performance. These two parameters are fixed for all the FS models in order to avoid getting biased experimental results and thus make a proper comparison. In order to assure the stability of the findings, execution of each algorithm has been carried out multiple times (ten times each) using an initial random population and the average of the findings is taken. Greater the values of population size and total iterations, the higher the performance, but the longer the computing time.

Table 4 Controlling parameters or hyperparameters for five mentioned models in the experiment

<i>Hyperparameters</i>	<i>FSGA</i>	<i>FSDE</i>	<i>FSPSO</i>	<i>FSACO</i>	<i>FSTDO</i>
Population size	30	30	30	30	30
No. of iterations	200	200	200	200	200
CR	0.8	0.9	-	-	-
$W_{\max}$	-	-	0.9	-	-
C2	-	-	2	-	-
MR	0.01	-	-	-	-
SF	-	0.8	-	-	-
$W_{\min}$	-	-	0.4	-	-
Beta	-	-	-	0.1	-
Alpha	-	-	-	1	-
C1	-	-	2	-	-
Rho	-	-	-	0.2	-

The crossover rates (CR) for FSGA and FSDE are set to 0.8 and 0.9 respectively. FSGA has another hyperparameter called mutation rate (MR) which is set to a value of 0.01. Similarly, for FSDE the value of the scaling factor (SF) is set to 0.8. In FSACO, the values of rho, beta and alpha were fixed at 0.2, 0.1 and 1 respectively. Similarly, in FSPSO, the values of $W_{\max}$ and $W_{\min}$ (initial weights) were adjusted to 0.9 and 0.4 respectively and the values of both the acceleration factors (C2 and C1) were set to 2. These hyperparameter values for the FSGA, FSPSO, FSDE and FSACO models have to be properly tuned to get accurate results. A slight mistake in tuning these hyperparameters can lead to poor performance by the algorithm and also can make the FS model biased. So, it can be said that except FSTDO all the other FS models are heavily dependent on the values of their respective hyperparameters for their performance.

Moreover, the parameters used for the four classifiers (NB, DT, KNN and QDA) used in our experiments are listed below:

- *NB classifier*: All the default parameters have been used and no parameters of the classifier were modified.
- *KNN classifier*: Number of neighbours (n_neighbours) has been set to 3. Rest of the parameters were not modified and default values were used.
- *DT classifier*: Default values of all the parameters have been used except for criteria and random state. The values of the parameters criteria (criterion) and random state (random_state) were set to 'entropy' and 0 respectively.
- *QDA classifier*: Default values of all the parameters have been used and no parameters were modified.

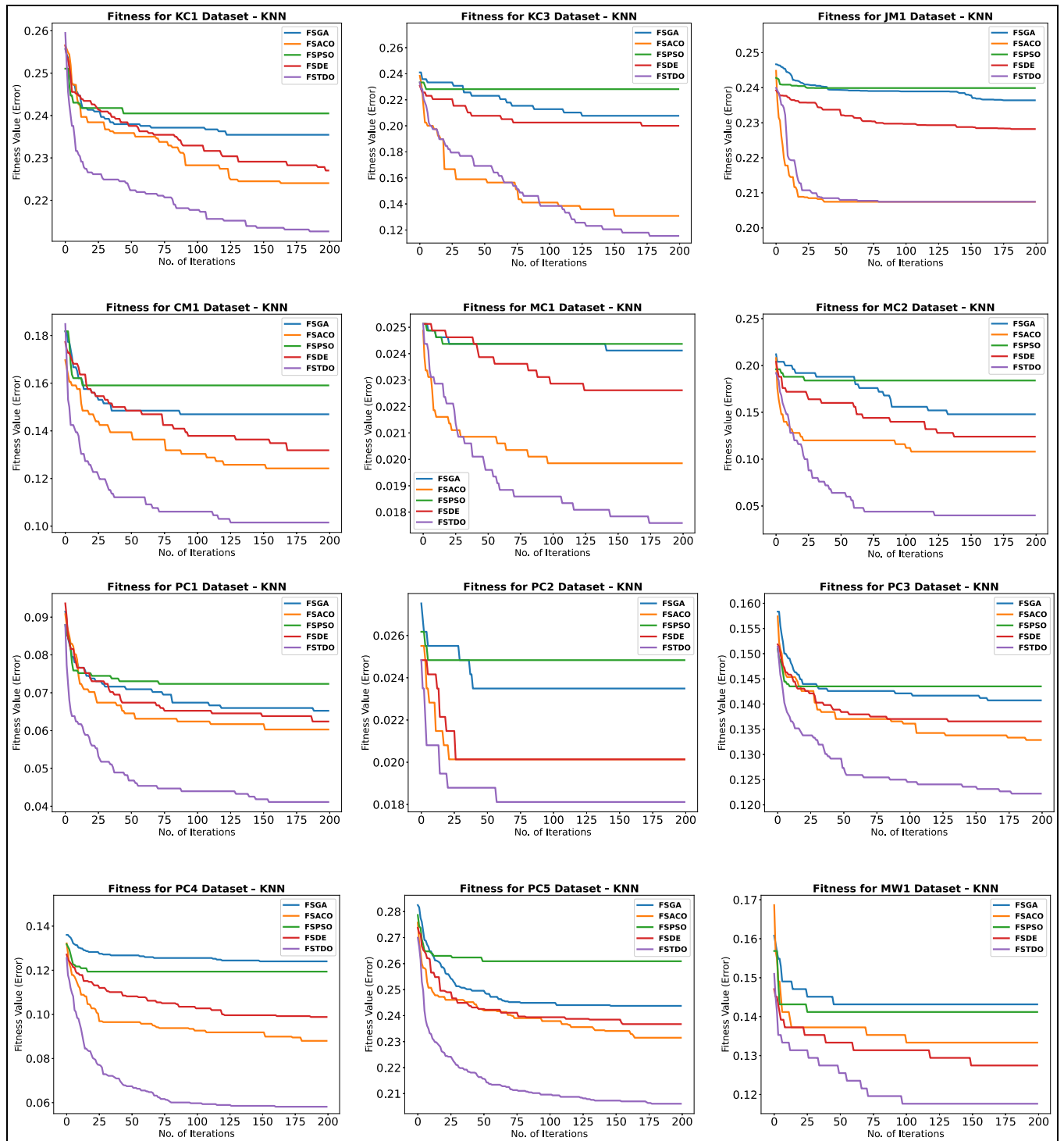
4.3 Experimental results

The following segment examines the results obtained from the experiments performed on FSPSO FSACO, FSDE, FSGA and FSTDO. The performance of the FS models is evaluated and compared based on the following indicators:

- Classification accuracy: higher accuracy of classification indicates better performance.
- Number of chosen features: smaller feature subset indicates more optimal performance.
- Fitness plots: lower fitness values indicate better performance.

Owing to the stochastic character of the aforementioned procedures, multiple trials were conducted using a random population.

By adjusting the governing parameters and algorithm-specific parameters, FS methods like FSGA, FSACO, FSDE, FSPSO, as well as FSTDO aim to improve the performance and accuracy. For all the FS models, the governing parameters, such as population size as well as number of iterations, are set to 30 and 200, respectively. Nevertheless, all of the FS techniques differ in terms of the only regulating factors that are particular to the algorithm, such as starting weight, selection operator, crossover probability, mutation probability, etc. Nonetheless, the emphasis should be given on the suggested FSTDO technique giving better performance with less dependence on hyper parameters. Most notably, the suggested FSTDO technique decreases dimensionality more than other FS approaches. The kind of data, number of chosen features, governing parameters of optimisation methods, and classifiers all affect how FS techniques behave.

Figure 4 Fitness plots for NB classifier comparing the five FS algorithms using 12 datasets (see online version for colours)**Table 5** Classification accuracy and FS comparison of the five different algorithms using 12 datasets

Datasets	Classifiers	Without FS (%)	FSDE (%)	FSPSO (%)	FSGA (%)	FSACO (%)	FSTDO (%)
KC1	KNN	69.6202	76.4557	74.6410	76.4556	77.5949	78.7342
	DT	72.1518	77.0886	73.1223	76.0337	76.8354	77.7215
	NB	74.2615	77.2152	76.1181	77.2151	77.4684	78.0169
	QDA	74.2616	78.1012	76.2869	77.9324	77.8481	78.6498
Average number of features		22	8.2	8.3	9.4	4.5	5.2
KC3	KNN	74.3588	80.0	76.1538	79.2307	86.9231	88.4616

Table 5 Classification accuracy and FS comparison of the five different algorithms using 12 datasets (continued)

<i>Datasets</i>	<i>Classifiers</i>	<i>Without FS (%)</i>	<i>FSDE (%)</i>	<i>FSPSO (%)</i>	<i>FSGA (%)</i>	<i>FSACO (%)</i>	<i>FSTDO (%)</i>
KC3	DT	76.9230	90.0	81.0256	87.6922	85.1282	92.8026
	NB	66.6666	76.9230	71.5380	76.1537	79.4872	81.7949
	QDA	76.9231	86.9231	79.7436	87.6923	86.4102	90.5129
	Average number of features	40	17.7	16.9	18.8	8.9	8.7
JM1	KNN	73.3461	77.1805	75.0610	76.3584	79.2550	79.2549
	DT	69.9421	78.7347	72.9352	73.2884	79.4861	79.6403
	NB	78.9980	79.8458	79.1586	79.6211	79.8909	80.1541
	QDA	75.8509	79.7174	79.0494	79.8265	79.7880	79.9422
	Average number of features	22	4.9	8.9	10.3	3	3.5
CM1	KNN	75.7575	86.8181	83.3333	85.3030	87.4242	89.8485
	DT	80.3030	89.6969	83.1818	88.9393	87.2727	92.4242
	NB	77.2727	83.4848	81.9696	83.1817	84.5454	85.6060
	QDA	83.3333	88.4848	85.0000	88.4848	88.1818	92.2727
	Average number of features	38	18.2	14.5	17.8	12.1	11.5
MC1	KNN	96.4824	97.7386	97.5628	97.5878	98.0151	98.2412
	DT	97.7386	98.5678	98.4924	98.6682	98.3417	99.5478
	NB	95.7286	97.7135	96.3065	97.7135	97.7386	97.9146
	QDA	97.4874	97.7387	97.4876	97.7387	97.7638	98.2412
	Average number of features	39	19.2	12.4	19.2	13.4	6.1
MC2	KNN	76.0	87.6	80.0	85.2	89.2	96.0
	DT	68.0	90.8	75.2	89.6	86.0	99.6
	NB	92.0	95.6	92.6	93.2	96.0	97.6
	QDA	84.0	96.0	88.4	95.2	95.2	99.2
	Average number of features	40	18.4	17.2	18.4	7.2	6.9
PC1	KNN	89.3617	93.7588	90.8510	93.4751	93.9716	95.8865
	DT	88.6524	95.0354	92.0567	95.0354	93.9716	96.3120
	NB	87.2340	91.2056	89.6453	90.6382	92.6241	93.5460
	QDA	86.5248	93.2624	89.7163	93.4752	92.9078	94.4681
	Average number of features	38	19.1	16.2	18.9	10.8	9.0
PC2	KNN	96.6442	97.9866	97.4496	97.6510	97.9866	98.1879
	DT	95.3020	98.6577	96.5101	98.3221	97.2483	98.6577
	NB	93.9597	96.9798	95.8395	96.3087	97.3154	97.3826
	QDA	97.3154	97.5839	97.3825	97.3825	98.1208	98.7919
	Average number of features	37	14.7	15.2	17.1	10.7	9.4
PC3	KNN	82.4074	86.3425	84.7685	85.9259	86.7129	87.7778
	DT	78.7037	86.3888	82.9167	86.5740	84.4444	88.0093
	NB	68.9814	86.8518	80.8333	86.8056	87.0833	87.9629
	QDA	62.0370	86.4352	83.3796	86.7592	86.8981	89.2593
	Average number of features	38	16.6	13.5	17.2	11.6	10.0
PC4	KNN	84.4961	90.1162	86.6279	87.5969	91.1628	94.1860
	DT	91.0852	93.5271	92.6356	93.5271	92.4031	95.6589
	NB	86.8217	91.5891	89.1085	91.7442	91.3953	93.2558
	QDA	47.6744	92.7132	86.9767	92.4806	91.8217	94.0310
	Average number of features	38	17.6	14.4	18.6	14.2	11.8
PC5	KNN	67.0553	76.3265	71.9820	75.6268	76.8513	79.3878
	DT	72.5947	77.7259	73.5276	77.8134	75.1603	79.5918
	NB	70.5539	71.5743	70.8163	71.4577	72.1865	73.2069
	QDA	69.3878	72.5656	70.5831	72.9154	71.1079	75.3936
	Average number of features	39	18.4	15.7	19.3	14.8	11.0

Table 5 Classification accuracy and FS comparison of the five different algorithms using 12 datasets (continued)

Datasets	Classifiers	Without FS (%)	FSDE (%)	FSPSO (%)	FSGA (%)	FSACO (%)	FSTDO (%)
MW1	KNN	78.4313	87.2549	84.7058	85.6863	86.6666	88.2353
	DT	74.5098	87.8431	82.7451	87.0588	85.2941	91.3725
	NB	76.4705	83.5294	78.8235	82.1569	87.2549	92.1569
	QDA	80.3921	88.6274	84.3137	86.6666	90.3921	92.5490
Average number of features		38	13.5	12.9	17.2	8.7	8.2

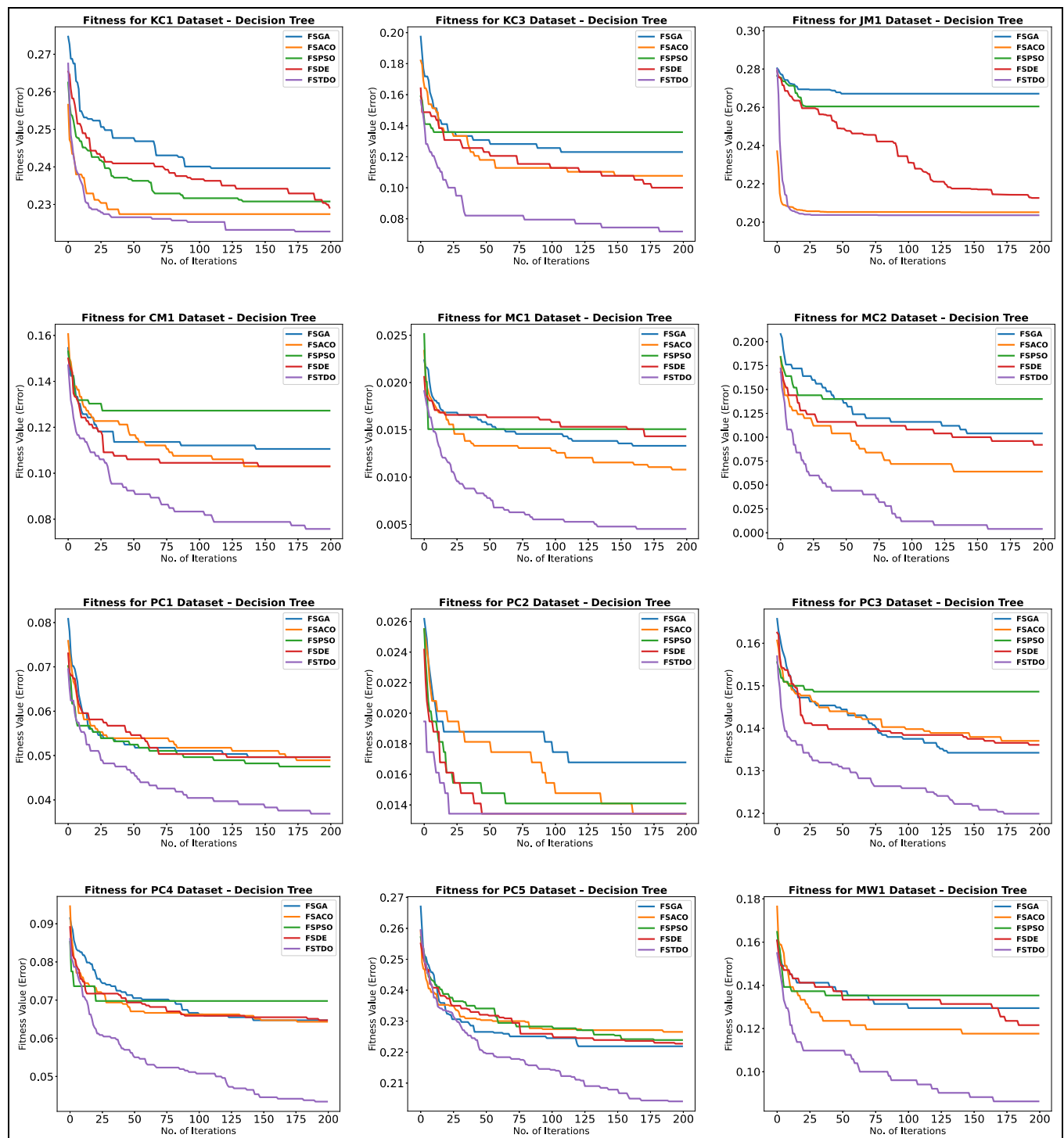
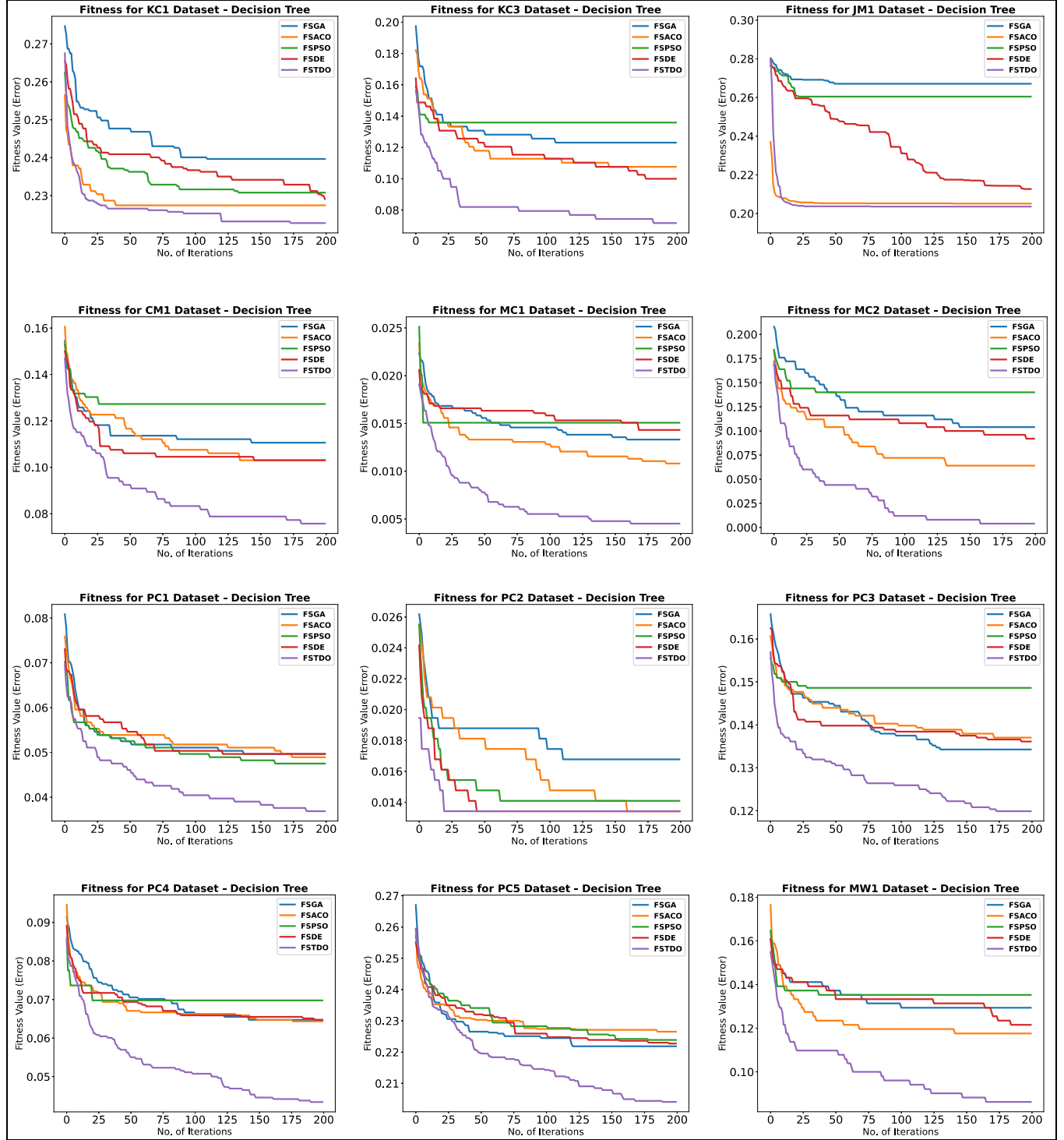
Figure 5 Fitness plots for KNN classifier comparing the five FS algorithms using 12 datasets (see online version for colours)

Figure 6 Fitness plots for DT classifier comparing the five FS algorithms using 12 datasets (see online version for colours)**Table 6** Friedman rank for all the six mentioned FS algorithm models for 12 datasets

Datasets	Classifiers	Without FS (%)	FSDE (%)	FSPSO (%)	FSGA (%)	FSACO (%)	FSTDO (%)
KC1	KNN	69.6202 (6)	76.4557 (3)	74.6410 (5)	76.4556 (4)	77.5949 (2)	78.7342 (1)
	DT	72.1518 (6)	77.0886 (2)	73.1223 (5)	76.0337 (4)	76.8354 (3)	77.7215 (1)
	NB	74.2615 (6)	77.2152 (3)	76.1181 (4)	77.2152 (3)	77.4684 (2)	78.0169 (1)
	QDA	74.2616 (6)	78.1012 (2)	76.2869 (5)	77.9324 (3)	77.8481 (4)	78.6498 (1)
Average rank for each model		6	2.5	4.75	3.5	2.75	1

Table 6 Friedman rank for all the six mentioned FS algorithm models for 12 datasets (continued)

Datasets	Classifiers	Without FS (%)	FSDE (%)	FSPSO (%)	FSGA (%)	FSACO (%)	FSTDO (%)
KC3	KNN	74.3588 (6)	80.0 (3)	76.1538 (5)	79.2307 (4)	86.9231 (2)	88.4616 (1)
	DT	76.9230 (6)	90.0 (2)	81.0256 (5)	87.6922 (3)	85.1282 (4)	92.8026 (1)
	NB	66.6666 (6)	76.9230 (3)	71.5380 (5)	76.1537 (4)	79.4872 (2)	81.7949 (1)
	QDA	76.9231 (6)	86.9231 (3)	79.7436 (5)	87.6923 (2)	86.4102 (4)	90.5129 (1)
Average rank for each model		6	2.75	5	3.25	3	1
JM1	KNN	73.3461 (6)	77.1805 (3)	75.0610 (5)	76.3584 (4)	79.2550 (2)	79.2549 (1)
	DT	69.9421 (6)	78.7347 (3)	72.9352 (5)	73.2884 (4)	79.4861 (2)	79.6403 (1)
	NB	78.9980 (6)	79.8458 (3)	79.1586 (5)	79.6211 (4)	79.8909 (2)	80.1541 (1)
	QDA	75.8509 (6)	79.7174 (4)	79.0494 (5)	79.8265 (2)	79.7880 (3)	79.9422 (1)
Average rank for each model		6	3.25	5	3.5	2.25	1
CM1	KNN	75.7575 (6)	86.8181 (3)	83.3333 (5)	85.3030 (4)	87.4242 (2)	89.8485 (1)
	DT	80.3030 (6)	89.6969 (2)	83.1818 (5)	88.9393 (3)	87.2727 (4)	92.4242 (1)
	NB	77.2727 (6)	83.4848 (3)	81.9696 (5)	83.1817 (4)	84.5454 (2)	85.6060 (1)
	QDA	83.3333 (6)	88.4848 (2)	85.0000 (5)	88.4848 (2)	88.1818 (3)	92.2727 (1)
Average rank for each model		6	2.5	5	6.5	2.75	1
MC1	KNN	96.4824 (6)	97.7386 (3)	97.5628 (5)	97.5878 (4)	98.0151 (2)	98.2412 (1)
	DT	97.7386 (6)	98.5678 (3)	98.4924 (4)	98.6682 (2)	98.3417 (5)	99.5478 (1)
	NB	95.7286 (5)	97.7135 (3)	96.3065 (4)	97.7135 (3)	97.7386 (2)	97.9146 (1)
	QDA	97.4874 (4)	97.7387 (3)	97.4876 (4)	97.7387 (3)	97.7638 (2)	98.2412 (1)
Average rank for each model		5.25	3	4.25	3	2.75	1
MC2	KNN	76.0 (6)	87.6 (3)	80.0 (5)	85.2 (4)	89.2 (2)	96.0 (1)
	DT	68.0 (6)	90.8 (2)	75.2 (5)	89.6 (3)	86.0 (4)	99.6 (1)
	NB	92.0 (6)	95.6 (3)	92.6 (5)	93.2 (4)	96.0 (2)	97.6 (1)
	QDA	84.0 (6)	96.0 (2)	88.4 (5)	95.2 (3)	95.2 (3)	99.2 (1)
Average rank for each model		6	2.5	5	3.5	2.75	1
PC1	KNN	89.3617 (6)	93.7588 (3)	90.8510 (5)	93.4751 (4)	93.9716 (2)	95.8865 (1)
	DT	88.6524 (6)	95.0354 (2)	92.0567 (4)	95.0354 (2)	93.9716 (3)	96.3120 (1)
	NB	87.2340 (6)	91.2056 (3)	89.6453 (5)	90.6382 (4)	92.6241 (2)	93.5460 (1)
	QDA	86.5248 (6)	93.2624 (3)	89.7163 (5)	93.4752 (2)	92.9078 (4)	94.4681 (1)
Average rank for each model		6	2.75	4.75	3	2.75	1
PC2	KNN	96.6442 (5)	97.9866 (2)	97.4496 (4)	97.6510 (3)	97.9866 (2)	98.1879 (1)
	DT	95.3020 (5)	98.6577 (1)	96.5101 (4)	98.3221 (2)	97.2483 (3)	98.6577 (1)
	NB	93.9597 (6)	96.9798 (3)	95.8395 (5)	96.3087 (4)	97.3154 (2)	97.3826 (1)
	QDA	97.3154 (5)	97.5839 (3)	97.3825 (4)	97.3825 (4)	98.1208 (2)	98.7919 (1)
Average rank for each model		5.25	2.25	4.25	3.25	2.25	1
PC3	KNN	82.4074 (6)	86.3425 (3)	84.7685 (5)	85.9259 (4)	86.7129 (2)	87.7778 (1)
	DT	78.7037 (6)	86.3888 (3)	82.9167 (5)	86.5740 (2)	84.4444 (4)	88.0093 (1)
	NB	68.9814 (6)	86.8518 (3)	80.8333 (5)	86.8056 (4)	87.0833 (2)	87.9629 (1)
	QDA	62.0370 (6)	86.4352 (4)	83.3796 (5)	86.7592 (3)	86.8981 (2)	89.2593 (1)
Average rank for each model		6	3.25	5	3.25	2.5	1
PC4	KNN	84.4961 (6)	90.1162 (3)	86.6279 (5)	87.5969 (4)	91.1628 (2)	94.1860 (1)
	DT	91.0852 (6)	93.5271 (2)	92.6356 (3)	93.5271 (2)	92.4031 (4)	95.6589 (1)
	NB	86.8217 (6)	91.5891 (3)	89.1085 (5)	91.7442 (2)	91.3953 (4)	93.2558 (1)
	QDA	47.6744 (6)	92.7132 (2)	86.9767 (5)	92.4806 (3)	91.8217 (4)	94.0310 (1)
Average rank for each model		6	2.5	4.5	2.75	3.5	1
PC5	KNN	67.0553 (6)	76.3265 (3)	71.9820 (5)	75.6268 (4)	76.8513 (2)	79.3878 (1)
	DT	72.5947 (6)	77.7259 (3)	73.5276 (5)	77.8134 (2)	75.1603 (4)	79.5918 (1)
	NB	70.5539 (6)	71.5743 (3)	70.8163 (5)	71.4577 (4)	72.1865 (2)	73.2069 (1)

Table 6 Friedman rank for all the six mentioned FS algorithm models for 12 datasets (continued)

Datasets	Classifiers	Without FS (%)	FSDE (%)	FSPSO (%)	FSGA (%)	FSACO (%)	FSTDO (%)
PC5	QDA	69.3878 (6)	72.5656 (3)	70.5831 (5)	72.9154 (2)	71.1079 (4)	75.3936 (1)
Average rank for each model		6	3	5	3	3	1
MW1	KNN	78.4313 (6)	87.2549 (2)	84.7058 (5)	85.6863 (4)	86.6666 (3)	88.2353 (1)
	DT	74.5098 (6)	87.8431 (2)	82.7451 (5)	87.0588 (3)	85.2941 (4)	91.3725 (1)
	NB	76.4705 (6)	83.5294 (3)	78.8235 (5)	82.1569 (4)	87.2549 (2)	92.1569 (1)
	QDA	80.3921 (6)	88.6274 (3)	84.3137 (5)	86.6666 (4)	90.3921 (2)	92.5490 (1)
Average rank for each model		6	2.5	5	3.75	2.75	1

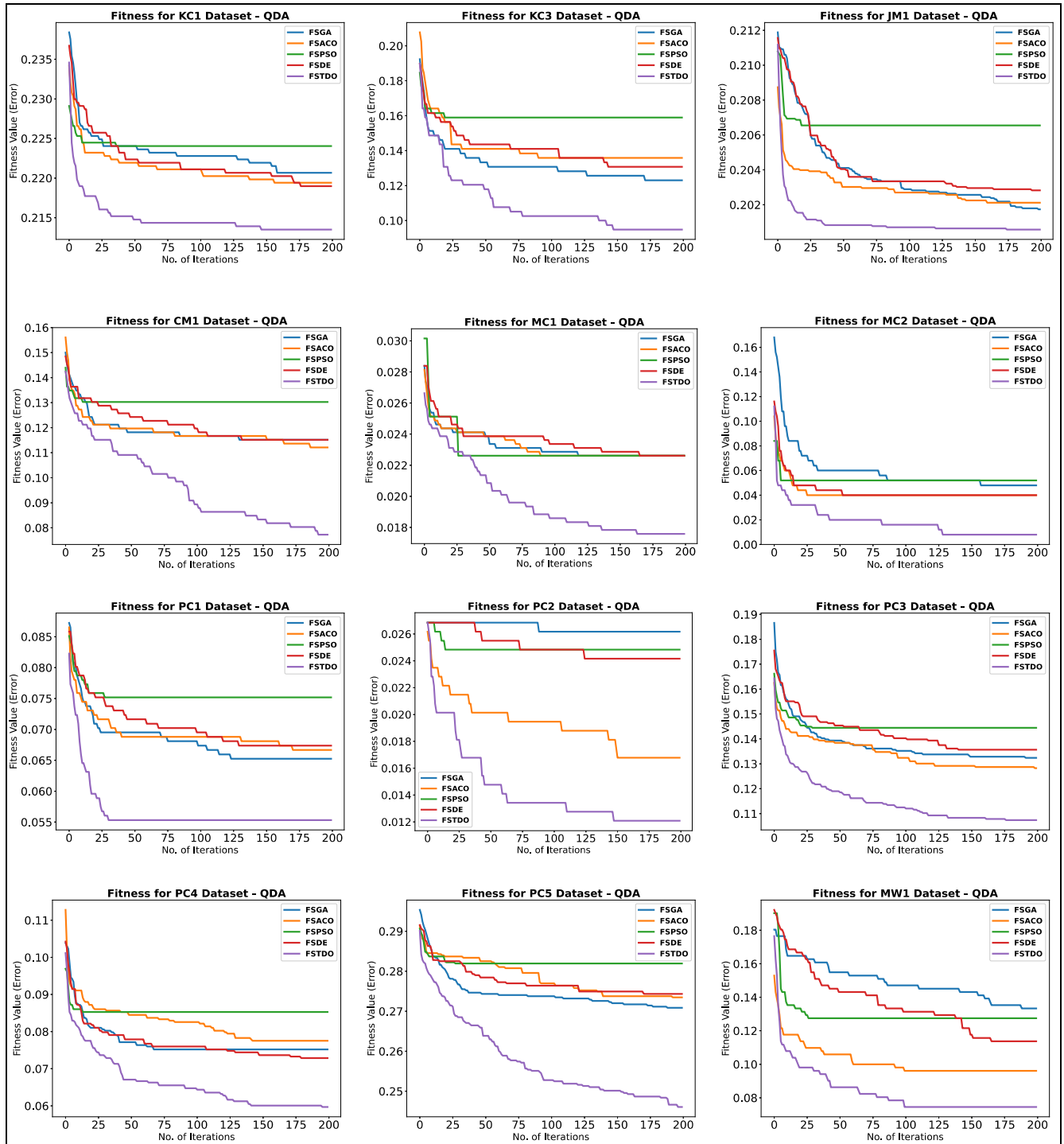
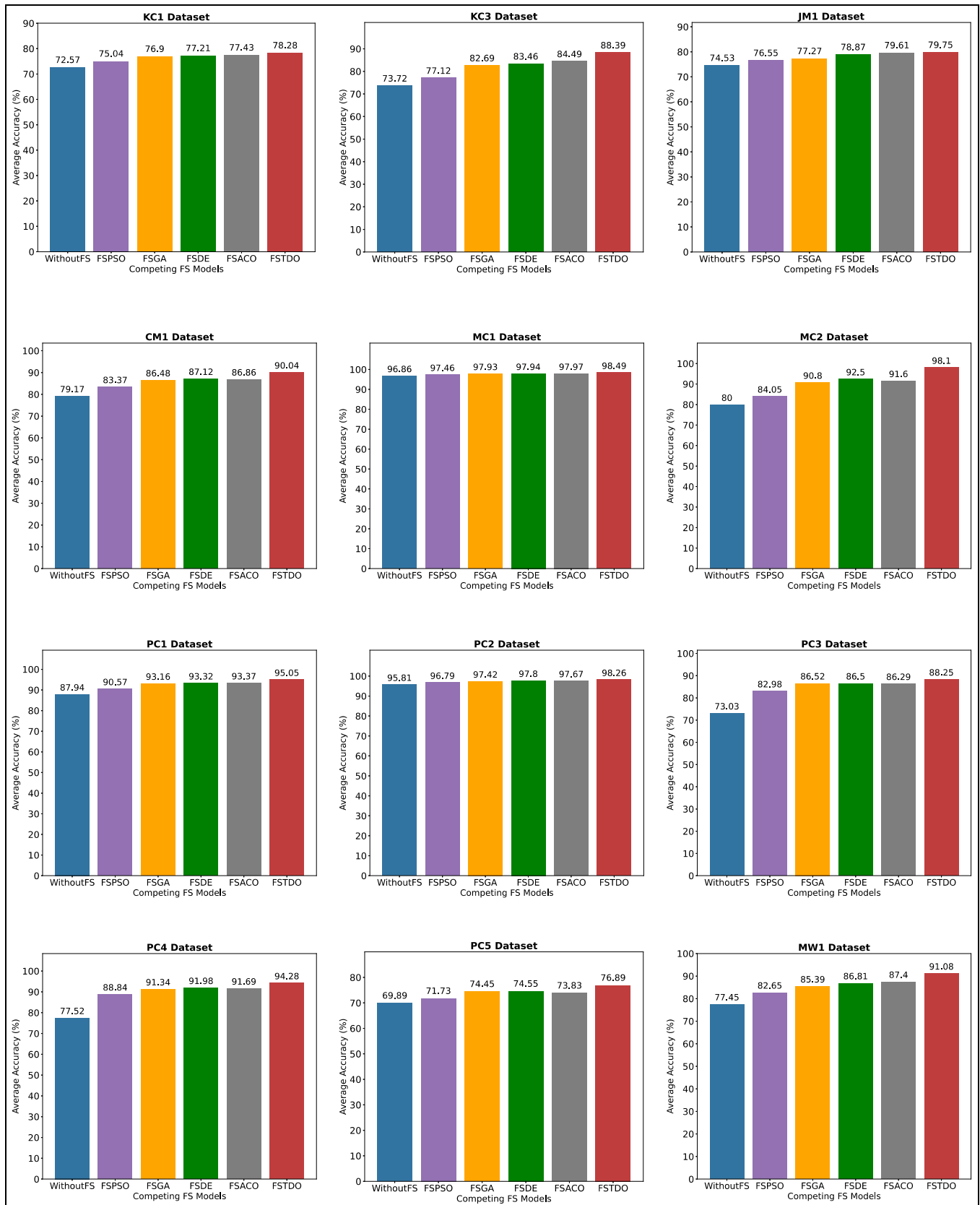
Figure 7 Fitness plots for QDA classifier comparing the five FS algorithms using 12 datasets (see online version for colours)

Figure 8 Graphical comparison among the average accuracies of the five FS models using 12 datasets (see online version for colours)

The trials were carried out on the aforementioned 12 datasets outlined in Table 2 and Table 3. Four classifiers namely KNN, NB, DT and QDA were used for performing

the experiments. Table 5 records the average classification accuracy and the average number of features selected for each dataset and FS approach. In Table 5, the number of

selected features is fractional as they are the average number of features obtained from the experiments conducted on four different classifiers. The results acquired suggest that the proposed FSTDO model outperforms all the competing FS models in majority of test cases. As contrasted to other models (FSGA, FSDE, FSPSO, and FSACO) that obtain comparable accuracy values with modest deviations, the suggested FSTDO methodology provides better accuracy results for practically almost all datasets. However, for KNN classifier in JM1 dataset, the classification accuracy of FSTDO and FSACO were almost same. Similarly, in PC2 dataset for DT classifier, FSTDO and FSDE had the same classification accuracy. Except these two exceptions, FSTDO gave higher classification accuracy for all the rest of the test cases.

Similarly, FSTDO showed better performance when it comes to the optimal number of selected features for majority of test cases. However, for KC1 and JM1 dataset, FSACO was better in reducing the dimension and in selecting the optimal features. Except these two cases, FSTDO outperformed all the other competing algorithms in selecting the optimal subset of features in all the rest of the test cases.

Following the experiments, the fitness (error) plots across the various datasets generated using FSGA, FSACO, FSPSO, FSDE and FSTDO for each classifier namely, NB, KNN, DT and QDA are portrayed in Figures 4, Figure 5, Figure 6, and Figure 7. It is clear from these graphs that for almost all the test cases, fitness value for FSTDO is lower than the other FS algorithms. As classification error was taken as fitness value, so it means that less fitness value leads to high accuracy. So, it can be said that FSTDO is the most effective FS algorithm among these other competing FS algorithms used for our experiment.

However, there were some corner cases where the other FS algorithms were as effective as FSTDO. In case of KNN classifier, it can be clearly seen from graph that for JM1 dataset, the fitness of FSACO is almost similar to FSTDO as shortly after a few iterations, both their graphs are coinciding. Similarly in case of DT classifier for PC2 dataset, after some iterations the fitness plot of FSDE and FSTDO coincides showing that they were having same fitness values. Also, towards the end the fitness plot of FSACO also coincided with FSDE and FSTDO.

All these seemingly minor discrepancies in the results might be attributed to the existence of data that is redundant, non-relevant details, or noise in the data collected for the experiment. In such instances, the offered FSTDO algorithm may be unable of producing superior results. However, for all the other test cases, the proposed FSTDO algorithm fared superior to the competing FS algorithms.

4.4 Discussion

Table 5 illustrates the accuracy (%) of the classification models and the average number of optimal features selected by the FS models for all the 12 datasets using FSGA, FSACO, FSPSO, FSDE and FSTDO for each classifier. The bold values in Table 5 list the best classification accuracy

and the optimal number of chosen characteristics among all the obtained results.

The results from the experiment demonstrate that utilising FS, prediction accuracy boosts for all datasets in all four classifiers. Moreover, the proposed FSTDO model showed the most promising accuracy in comparison to four other competitor algorithms. FSTDO model also reduced the dimensions of the datasets most effectively as it just keeps the pertinent characteristics and discards the non-essential features along with noise.

Figure 4, Figure 5, Figure 6, and Figure 7 displays the comparison of the fitness values (error in prediction) for all the FS models with all the classifiers. It is seen that the proposed FSTDO model showed the best fitness values. It means that the error in prediction accuracy was lowest for the proposed FSTDO approach.

The comparison among average accuracies of the FS models for the 12 SFP datasets is presented visually in the form of bar graphs in Figure 8. For this comparison, the average accuracy is calculated based on the accuracies obtained from the four different classifiers (KNN, DT, NB QDA). The bar graphs clearly indicate the superior performance and effectiveness of the proposed FSTDO model compared to the other FS models.

Since the global optimal solution could only be provided by FSTDO with great exploitation power, simulation results demonstrate that FSTDO is also the best FS model among all the competitors. The study of the findings in this table demonstrates the dominance of FSTDO over the four competing FS algorithms by showing that it was able to deliver outcomes that were much more competitive across all datasets.

4.5 Time complexity

The time complexity of the FSTDO model is discussed in this section. The time complexity for the initialisation of FSTDO is $O(A.k)$, where A is the number of individuals of the population of TD and k is the number of features. FSTDO solves the FS problem in M number of iterations. The time complexity involved in the update process of the individuals of the population on their way to the prey or carcass is $O(A.k.M)$. The time complexity involved in the second strategy of the process of chasing the prey is $O(A_{s2}.k.M)$, where the number of TD that have used the second strategy of feeding is represented by A_{s2} . So, the total time complexity of FSTDO is $O((A.k).((1 + M) + (M.A_{s2})))$.

5 Statistical analysis and discussion

Statistical analysis (Demšar, 2006) is a classic approach for quantifying, analysing, interpreting, and drawing outcomes based on data. It may compare the performance (classification accuracy) of various current models to that of the proposed model. There are two types of statistical tests:

Table 7 Depiction of FS algorithms along with their average ranks for 12 datasets ($AvgRank_{Datasets}$)

Datasets	Without FS	FSDE	FSPSO	FSGA	FSACO	FSTDO
KC1	6	2.5	4.75	3.5	2.75	1
KC3	6	2.75	5	3.25	3	1
JM1	6	3.25	5	3.5	2.25	1
CM1	6	2.5	5	6.5	2.75	1
MC1	5.25	3	4.25	3	2.75	1
MC2	6	2.5	5	3.5	2.75	1
PC1	6	2.75	4.75	3	2.75	1
PC2	6	2.25	4.25	3.25	2.25	1
PC3	6	3.25	5	3.25	2.5	1
PC4	6	2.5	4.5	2.75	3.5	1
PC5	6	3	5	3	3	1
MW1	6	2.5	5	3.75	2.75	1
$AvgRank_{Datasets}$	5.938 (6)	2.729 (2)	4.791 (5)	3.521 (4)	2.75 (3)	1 (1)

- a parametric tests (Pajevic and Bassier, 2003) which involve parameters
- b non-parametric tests (Gibbons and Chakraborti, 2011) which are performed using hypothesis.

In this paper, we focused at the Friedman test (Friedman, 1937, 1940). In order to measure the statistically significant aspects among the models, this non-parametric rank-based test is carried out. It is employed to examine the outcomes obtained when the same parameters are implemented in a variety of contexts related with multiple models, including the suggested model for ranking. Here, only three or more aligned groups are scrutinised. Based on the model's accuracy, Friedman test allocates a rating to each model. The highest accuracy value receives the lowest rank while the highest rank is provided to the model with smallest accuracy. Moreover, in this statistical test, all the models are considered to be equal according to the H_0 (null hypothesis). This suggests that there is no discernible difference amongst them; on the flip side, the H_1 (alternative hypothesis) contends that the models are incompatible. If and only if the p -value falls below the significance level, the null hypothesis is eliminated. It indicates that there are at least two distinct models. In Table 6, the average rank of each model or $AvgRank_{Models}$ (Without FS, FSGA, FSDE, FSACO, FSPSO and FSTDO) is calculated in respect to various classifiers (NB, KNN, DT, and QDA). It is evaluated by summing up the ranks of the particular model obtained from each classifier for every given dataset divided by total number of classifiers as shown in equation (16). Similar to this, Table 7 uses equation (17) to compute the average rank of several models or $AvgRank_{Datasets}$ in relation to the aforementioned datasets. By dividing the total average rank of models by the number of datasets, average rankings for each FS models are calculated.

$$AvgRank_{Models} = \sum \text{ranks of classifiers} / TC \quad (16)$$

$$AvgRank_{Datasets} = \sum AvgRank_{Models} / TD \quad (17)$$

Here, TC is total number of classifiers and TD is total number of datasets.

From Table 6 itself, it can be clearly declared that FSTDO is the most efficient classification model till this part of the experiment as its Friedman Rank is constantly 1. Next, the $AvgRank_{Datasets}$ are calculated and presented in Table 7 to further verify this statement.

The average ranks obtained from Table 6 for each of the 6 mentioned models are represented below as:

- $AvgRank_{Datasets}$ for TDO = $AR_1 = 1$.
- $AvgRank_{Datasets}$ for DE = $AR_2 = 2.729$.
- $AvgRank_{Datasets}$ for ACO = $AR_3 = 2.75$.
- $AvgRank_{Datasets}$ for GA = $AR_4 = 3.521$.
- $AvgRank_{Datasets}$ for PSO = $AR_5 = 4.791$.
- $AvgRank_{Datasets}$ for Without FS = $AR_6 = 5.938$.

Now, the X_F^2 is calculated using the average rankings of the models, and the reported value of 44.982 is obtained using equation (18).

$$X_F^2 = \frac{12N}{P(P+1)} \times \left[\sum_j AvgRank_{Datasets}^2 - \frac{P(P+1)^2}{4} \right] \quad (18)$$

$$F_F = \frac{X_F^2 (N-1)}{N(P-1) - X_F^2} \quad (19)$$

Here N and P respectively, stand for the number of datasets which is 12 and the number of models whose value equates to 6, which are taken into account. We evaluate the Friedman statistics as 32.947 as stated in equation (19). The degree of freedom is computed as $(6 - 1)$ and $[(12 - 1) * (6 - 1)]$ which is equal to $(5, 55)$. The critical value is also evaluated using degrees of freedom $(5, 55)$ and significance

level ($\alpha = 0.05$) and it amounts to 2.383. The density plot with degree of freedom (5, 55) is diagrammed in Figure 9.

Figure 9 Density plot (see online version for colours)

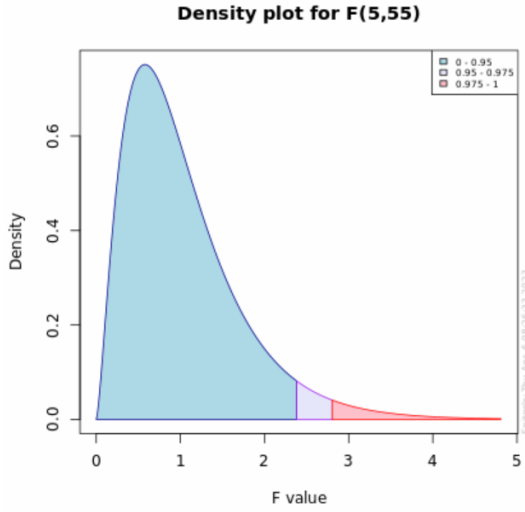


Table 8 Holm evaluation findings

<i>S no.</i>	<i>Models with FS</i>	<i>z-value</i>	<i>p-value</i>	$(\alpha/P-i)$
1	FSTDO: FSDE	2.263	0.01182	0.0506
2	FSTDO: FSACO	2.291	0.01098	0.0251
3	FSTDO: FSGA	3.3	0.00048	0.0166
4	FSTDO: FSPSO	6.270	0.00001	0.0125
5	FSTDO: without FS	6.463	0.00001	0.01

It should be emphasised well enough that because the critical value is significantly lower than the Friedman statistics, the null hypothesis is discarded resulting in adoption of alternative hypothesis. Therefore, the post hoc test experiment was carried out with the help of Holm procedure (Iman and Davenport, 1980; García et al., 2010). By employing the Holm method, the *z-value* and *p-value* are taken into consideration to examine how well each unique model performs in comparison to other models. Equation (20) is used to compute *z-value*. Thereafter, the obtained *z-value* and normal distribution table are used to compute the *p-value*.

$$z = \frac{AR_i - AR_j}{\sqrt{\frac{P(P+1)}{6N}}} \quad (20)$$

Here,

P Number of FS models employed for the execution of the experiment, i.e., 6.

N Number of datasets used for the experiment, i.e., 12.

AR_i Average rank or $AvgRank_{Datasets}$ for i^{th} model chosen at a time.

AR_j Average rank or $AvgRank_{Datasets}$ for j^{th} model chosen at a time.

The $\alpha(P-i)$, *p-value* and *z-value*, of all models has been contrasted to FSTDO and the outcomes are summarised in Table 8.

Table 8 demonstrates that all the *p-values* are much lower than $\alpha(P-i)$. This indicates that FSTDO is statistically significant and outperforms all the other FS models. Thus, the statistical analysis showed that the proposed FSTDO is the superior model as well as most efficient for FS. In future, all the statistical methods will be applied in several other classification algorithms (Chaturvedi et al., 2023; Kumar and Srivastava, 2020; Kumar et al., 2018; Kumar, 2022) for the validation of the work.

6 Conclusions and future work

In this paper, a novel FS algorithm named FSTDO is proposed for SFP. The data used for predicting the software defects are very high dimensional and to extract the relevant information from that data, FS is used. For the experiments, 12 open-source software defect datasets from the PROMISE repository of NASA were used. Four different classifiers, namely: KNN, DT, QDA, and NB were used for conducting the experiments on each of the 12 SFP datasets. Moreover, the proposed FSTDO model's performance was compared with four well-known FS algorithms: FSGA, FSPSO, FSACO, and FSDE to inspect its quality. According to the simulation findings, the FSTDO model increases the prediction accuracy for all four classifiers across all datasets, outperforming the other four algorithms. Additionally, it chooses the most appropriate and optimal number of features while ignoring noise and unneeded features. One drawback of this proposed method is lose information if such unnecessary, redundant, and disruptive data are eliminated. However, the experimental results of the FSTDO model were also statistically validated using Friedman and Holm tests, and it was found that the results of FSTDO are statistically significant. This showed the superiority of the proposed FSTDO model over the competitors in solving FS problems.

In future, the proposed FS algorithm may be used for work that might make this algorithm more efficient. This FS algorithm may be further improved and different classifiers can be embedded to provide even better accuracy. Furthermore, various new optimisation algorithms can be integrated with wrapper-based FS approaches to solve FS problems. These FS models can have various real-life applications like weather forecasting, hazardous asteroid classification, disease prediction, and space debris classification.

References

- Abdel-Basset, M., Abdel-Fatah, L. and Sangaiah, A.K. (2018) 'Metaheuristic algorithms: a comprehensive review', *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, pp.185–231, Academic Press, Elsevier, USA.
- Aghdam, M.H., Ghasem-Aghaei, N. and Basiri, M.E. (2008) 'Application of ant colony optimization for feature selection in text categorization', in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, IEEE, June, pp.2867–2873.
- Ahmad, I. (2015) 'Feature selection using particle swarm optimization in intrusion detection', *International Journal of Distributed Sensor Networks*, Vol. 11, No. 10.
- Al Salami, N.M. (2009) 'Ant colony optimization algorithm', *UbiCC Journal*, Vol. 4, No. 3, pp.823–826.
- Al-Ani, A. (2005) 'Feature subset selection using ant colony optimization', *International Journal of Computational Intelligence*, Vol. 2, No. 1, pp.53–58.
- Al-Ani, A., Alsukker, A. and Khushaba, R.N. (2013) 'Feature subset selection using differential evolution and a wheel based search strategy', *Swarm and Evolutionary Computation*, Vol. 9, pp.15–26.
- Almasoudy, F.H., Al-Yaseen, W.L. and Idrees, A.K. (2020) 'Differential evolution wrapper feature selection for intrusion detection system', *Procedia Computer Science*, Vol. 167, pp.1230–1239.
- Boetticher, G. (2007) *The PROMISE Repository of Empirical Software Engineering Data* [online] <http://promisedata.org/repository> (accessed 20 May 2023).
- Can, H., Jianchun, X., Ruide, Z., Juelong, L., Qiliang, Y. and Liqiang, X. (2013) 'A new model for software defect prediction using particle swarm optimization and support vector machine', in *2013 25th Chinese Control and Decision Conference (CCDC)*, IEEE, May, pp.4106–4110.
- Catal, C. and Diri, B. (2009) 'A systematic review of software fault prediction studies', *Expert Systems with Applications*, Vol. 36, No. 4, pp.7346–7354.
- Chaturvedi, S., Kumar, N. and Kumar, R. (2023) 'A PSO-optimized novel PID neural network model for temperature control of jacketed CSTR: design, simulation, and a comparative study', *Soft Comput.*, <https://doi.org/10.1007/s00500-023-09138-0>.
- Dadaneh, B.Z., Markid, H.Y. and Zakerolhosseini, A. (2016) 'Unsupervised probabilistic feature selection using ant colony optimization', *Expert Systems with Applications*, Vol. 53, pp.27–42.
- Das, H., Naik, B. and Behera, H.S. (2020a) 'A hybrid neuro-fuzzy and feature reduction model for classification', *Advances in Fuzzy Systems*, Vol. 2020, pp.1–15.
- Das, H., Naik, B. and Behera, H.S. (2020b) 'An experimental analysis of machine learning classification algorithms on biomedical data', in *Proceedings of the 2nd International Conference on Communication, Devices and Computing: ICCDC 2019*, Springer Singapore, pp.525–539.
- Das, H., Naik, B. and Behera, H.S. (2021) 'Optimal selection of features using artificial electric field algorithm for classification', *Arabian Journal for Science and Engineering*, Vol. 46, No. 9, pp.8355–8369.
- Das, H., Naik, B. and Behera, H.S. (2022) 'A Jaya algorithm based wrapper method for optimal feature selection in supervised classification', *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 6, pp.3851–3863.
- Dehghani, M., Hubálovský, Š. and Trojovský, P. (2022) 'Tasmanian devil optimization: a new bio-inspired optimization algorithm for solving optimization algorithm', *IEEE Access*, Vol. 10, pp.19599–19620.
- Demšar, J. (2006) 'Statistical comparisons of classifiers over multiple data sets', *The Journal of Machine Learning Research*, Vol. 7, pp.1–30.
- Deriche, M. (2009) 'Feature selection using ant colony optimization', in *2009 6th International Multi-Conference on Systems, Signals and Devices*, IEEE, March, pp.1–4.
- Diao, C., Zhang, D., Liang, W., Li, K-C., Hong, Y. and Gaudiot, J-L. (2023) 'A novel spatial-temporal multi-scale alignment graph neural network security model for vehicles prediction', *IEEE Transactions on Intelligent Transportation Systems*, January, Vol. 24, No. 1, pp.904–914, DOI: 10.1109/TITS.2022.3140229.
- Fernández Martínez, J.L. and García Gonzalo, E. (2008) 'The generalized PSO: a new door to PSO evolution', *Journal of Artificial Evolution and Applications*, Article ID 861275, pp.1–15.
- Friedman, M. (1937) 'The use of ranks to avoid the assumption of normality implicit in the analysis of variance', *Journal of the American Statistical Association*, Vol. 32, No. 200, pp.675–701.
- Friedman, M. (1940) 'A comparison of alternative tests of significance for the problem of m rankings', *The Annals of Mathematical Statistics*, Vol. 11, No. 1, pp.86–92.
- García, S., Fernández, A., Luengo, J. and Herrera, F. (2010) 'Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power', *Information Sciences*, Vol. 180, No. 10, pp.2044–2064.
- Gibbons, J.D. and Chakraborti, S. (2011) 'Nonparametric statistical inference', in *International Encyclopedia of Statistical Science*, Springer, Berlin, Heidelberg, pp.977–979.
- Holland, J.H. (1992) 'Genetic algorithms', *Scientific American*, Vol. 267, No. 1, pp.66–73.
- Iman, R.L. and Davenport, J.M. (1980) 'Approximations of the critical region of the fbiectkan statistic', *Communications in Statistics-Theory and Methods*, Vol. 9, No. 6, pp.571–595.
- Jordan, M.I. and Mitchell, T.M. (2015) 'Machine learning: trends, perspectives, and prospects', *Science*, Vol. 349, No. 6245, pp.255–260.
- Karaboğa, D. and Ökdem, S. (2004) 'A simple and global optimization algorithm for engineering problems: differential evolution algorithm', *Turkish Journal of Electrical Engineering and Computer Sciences*, Vol. 12, No. 1, pp.53–60.
- Kaya, Y. and Pehlivan, H. (2015) 'Feature selection using genetic algorithms for premature ventricular contraction classification', in *2015 9th International Conference on Electrical and Electronics Engineering (ELECO)*, IEEE, November, pp.1229–1232.
- Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimization', in *Proceedings of ICNN'95-International Conference on Neural Networks*, IEEE, November, Vol. 4, pp.1942–1948.

- Khanesar, M.A., Teshnehlab, M. and Shoorehdeli, M.A. (2007) 'A novel binary particle swarm optimization', in *2007 Mediterranean Conference on Control & Automation*, IEEE, June, pp.1–6.
- Khushaba, R.N., Al-Ani, A. and Al-Jumaily, A. (2009) 'Feature subset selection using differential evolution', in *Advances in Neuro-Information Processing: 15th International Conference, ICONIP 2008*, Springer Berlin Heidelberg, Auckland, New Zealand, 25–28 November 2008, Revised Selected Papers, Part I 15, pp.103–110.
- Khushaba, R.N., Al-Ani, A. and Al-Jumaily, A. (2011) 'Feature subset selection using differential evolution and a statistical repair mechanism', *Expert Systems with Applications*, Vol. 38, No. 9, pp.11515–11526.
- Kumar, M., Husain, D., Upreti, N. and Gupta, D. (2010) *Genetic Algorithm: Review and Application*, SSRN: 3529843.
- Kumar, R. (2022) 'Memory recurrent Elman neural network-based identification of time-delayed nonlinear dynamical system', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 53, No. 2, pp.753–762.
- Kumar, R. and Srivastava, S. (2020) 'Externally recurrent neural network based identification of dynamic systems using Lyapunov stability analysis', *ISA Transactions*, Vol. 98, pp.292–308.
- Kumar, R., Srivastava, S., Gupta, J.R.P. and Mohindru, A. (2018) 'Self-recurrent wavelet neural network-based identification and adaptive predictive control of nonlinear dynamical systems', *International Journal of Adaptive Control and Signal Processing*, Vol. 32, No. 9, pp.1326–1358.
- Leng, J., Valli, C. and Armstrong, L. (2010) 'A wrapper-based feature selection for analysis of large data sets', *2010 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)*, IEEE.
- Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J. and Liu, H. (2017) 'Feature selection: a data perspective', *ACM computing surveys (CSUR)*, Vol. 50, No. 6, pp.1–45.
- Liang, W., Li, Y., Xie, K., Zhang, D., Li, K.-C., Souri, A. and Li, K. (2022) 'Spatial-temporal aware inductive graph neural network for C-ITS data recovery', *IEEE Transactions on Intelligent Transportation Systems*, DOI: 10.1109/TITS.2022.3156266.
- Liang, W., Yang, Y., Yang, C., Hu, Y., Xie, S., Li, K.-C. and Cao, J. (2023) 'PDPChain: a consortium blockchain-based privacy protection scheme for personal data', *IEEE Transactions on Reliability*, June, Vol. 72, No. 2, pp.586–598, DOI: 10.1109/TR.2022.3190932.
- Long, J., Liang, W., Li, K.-C., Wei, Y. and Marino, M.D. (2023) 'A regularized cross-layer ladder network for intrusion detection in industrial internet-of-things', *IEEE Transactions on Industrial Informatics*, February, Vol. 19, No. 2, pp.1747–1755, DOI: 10.1109/TII.2022.3204034.
- Lu, J., Zhao, T. and Zhang, Y. (2008) 'Feature selection based-on genetic algorithm for image annotation', *Knowledge-Based Systems*, Vol. 21, No. 8, pp.887–891.
- Mahesh, B. (2020) 'Machine learning algorithms-a review', *International Journal of Science and Research (IJSR)*, Vol. 9, pp.381–386.
- Miller, W., Hayes, V.M., Ratan, A., Petersen, D.C., Wittekindt, N.E., Miller, J. and Schuster, S.C. (2011) 'Genetic diversity and population structure of the endangered marsupial *Sarcophilus harrisii* (Tasmanian devil)', *Proceedings of the National Academy of Sciences*, Vol. 108, No. 30, pp.12348–12353.
- Mirjalili, S. and Mirjalili, S. (2019) 'Genetic algorithm', *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pp.43–55, Springer, Berlin/Heidelberg, Germany.
- Mothilal, R.K., Sharma, A. and Tan, C. (2020) 'Explaining machine learning classifiers through diverse counterfactual explanations', in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, January, pp.607–617.
- Osisanwo, F.Y., Akinsola, J.E.T., Awodele, O., Hinmikaiye, J.O., Olakanmi, O. and Akinjobi, J. (2017) 'Supervised machine learning algorithms: classification and comparison', *International Journal of Computer Trends and Technology (IJCTT)*, Vol. 48, No. 3, pp.128–138.
- Padhi, B.K., Chakravarty, S., Naik, B., Pattanayak, R.M. and Das, H. (2022) 'RHSOFS: feature selection using the rock hyrax swarm optimization algorithm for credit card fraud detection system', *Sensors*, Vol. 22, No. 23, p.9321.
- Pajevic, S. and Basser, P.J. (2003) 'Parametric and non-parametric statistical analysis of DT-MRI data', *Journal of Magnetic Resonance*, Vol. 161, No. 1, pp.1–14.
- Porkodi, R. (2014) 'Comparison of filter based feature selection algorithms: an overview', *International Journal of Innovative Research in Technology & Science*, Vol. 2, No. 2, pp.108–113.
- Rish, I. (2001) 'An empirical study of the naive Bayes classifier', in *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, August, Vol. 3, No. 22, pp.41–46.
- Safavian, S.R. and Landgrebe, D. (1991) 'A survey of decision tree classifier methodology', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 3, pp.660–674.
- Saraç, E. and Özel, S.A. (2014) 'An ant colony optimization based feature selection for web page classification', *The Scientific World Journal*, Vol. 2014, Article ID 649260, pp.1–16.
- Shahamat, H. and Pouyan, A.A. (2015) 'Feature selection using genetic algorithm for classification of schizophrenia using fMRI data', *Journal of AI and Data Mining*, Vol. 3, No. 1, pp.30–37.
- Shehu, H.A., Browne, W. and Eisenbarth, H. (2021) 'Particle swarm optimization for feature selection in emotion categorization', in *2021 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, June, pp.752–759.
- Simon, D. (2013) *Evolutionary Optimization Algorithms*, John Wiley & Sons.
- Sreeja, N.K. and Sankar, A. (2015) 'Pattern matching based classification using ant colony optimization based feature selection', *Applied Soft Computing*, Vol. 31, pp.91–102.
- Tabakhi, S., Moradi, P. and Akhlaghian, F. (2014) 'An unsupervised feature selection algorithm based on ant colony optimization', *Engineering Applications of Artificial Intelligence*, Vol. 32, pp.112–123.
- Tharwat, A. (2016) 'Linear vs. quadratic discriminant analysis classifier: a tutorial', *International Journal of Applied Pattern Recognition*, Vol. 3, No. 2, pp.145–180.
- Tindell, K.W., Burns, A. and Wellings, A.J. (1992) 'Allocating hard real-time tasks: an NP-hard problem made easy', *Real-Time Systems*, Vol. 4, No. 2, pp.145–165.
- Tu, C.J., Chuang, L.Y., Chang, J.Y. and Yang, C.H. (2007) 'Feature selection using PSO-SVM', *IAENG International Journal of Computer Science*, Vol. 33, No. 1, pp.1–6.

- Wah, Y.B., Ibrahim, N., Hamid, H.A., Abdul-Rahman, S. and Fong, S. (2018) 'Feature selection methods: case of filter and wrapper approaches for maximising classification accuracy', *Pertanika Journal of Science & Technology*, Vol. 26, No. 1, pp.329–340.
- Xu, Z., Liang, W., Li, K.C., Xu, J., Zomaya, A.Y. and Zhang, J. (2021) 'A time-sensitive token-based anonymous authentication and dynamic group key agreement scheme for industry 5.0', *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 10, pp.7118–7127.
- Yigit, H. (2013) 'A weighting approach for KNN classifier', in *2013 International Conference on Electronics, Computer and Computation (ICECCO)*, IEEE, November, pp.228–231.
- Yildiz, O. and Doğru, I.A. (2019) 'Permission-based android malware detection system using feature selection with genetic algorithm', *International Journal of Software Engineering and Knowledge Engineering*, Vol. 29, No. 2, pp.245–262.
- Yoshida, H., Leardi, R., Funatsu, K. and Varmuza, K. (2001) 'Feature selection by genetic algorithms for mass spectral classifiers', *Analytica Chimica Acta*, Vol. 446, Nos. 1–2, pp.483–492.
- Zhang, S., Hu, B., Liang, W., Li, K.C. and Gupta, B.B. (2023) 'A caching-based dual K-anonymous location privacy-preserving scheme for edge computing', *IEEE Internet of Things Journal*, Vol. 10, No. 11.
- Zheng, K. and Wang, X. (2018) 'Feature selection method with joint maximal information entropy between features and class', *Pattern Recognition*, Vol. 77, pp.20–29.