Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2021.DOI*

# Comprehensive Analysis of Nature-Inspired Algorithms for Parkinson's Disease Diagnosis

**SHAKILA SHAFIQ[1], (Member, IEEE), SABBIR AHMED[1], (Member, IEEE), M SHAMIM KAISER[1], (Senior Member, IEEE), MUFTI MAHMUD[2], (Senior Member, IEEE), MD. SHAHADAT HOSSAIN[3], (Senior Member, IEEE), KARL ANDERSSON[4], (Senior Member, IEEE)**

[1]Institute of Information Technology, Jahangirnagar University, Savar, Dhaka-1342, Bangladesh (e-mail: shakilaju46, sabbir.iit.ju@gmail.com, mskaiser@juniv.edu).
[2]Department of Computer Science, Nottingham Trent University, Nottingham, U.K. (e-mail:muftimahmud@gmail.com).
[3]Department of Computer Science and Engineering, University of Chittagong, University-4331, Chattogram, Bangladesh
[4]Pervasive and Mobile Computing Laboratory, Luleå University of Technology, 931 87 Skellefteå, Sweden

Corresponding author: Mufti Mahmud (e-mail: muftimahmud@gmail.com) Karl Andersson (email:karl.andersson@ltu.se ).

**ABSTRACT**
**Background:**
Parkinson's disease (PD) is a prominent neurodegenerative disease that damages the neurons of the substantia nigra, causing irreversible impairments leading to involuntary movements. As this disease disrupts patients' daily activities in a mature stage, early detection of the disease is crucial. Several methods based on nature-inspired (NI) algorithms have been proposed for PD detection and patient management. As there are several NI algorithms for feature selection, a mapping with an individual machine learning (ML) classifier is necessary to obtain optimal performance of the detection pipeline.
**Method:**
To fill this gap, in this work, 13 NI algorithms and 11 ML classifiers were selected, and critical comparisons were performed regarding their combined performance in detecting PD. Each NI algorithm was employed to select an optimal feature set which was then classified by the 11 ML classifiers keeping the same parameters. This generated 143 NI-ML pairs, which were carefully compared to find the best-performing pairs considering several assessment criteria such as accuracy, cross-validation mean score, precision, recall and F1-score.
**Results:**
The results of the extensive comparative analysis allowed the ranking of the algorithms in the 50th, 75th and 95th percentile to identify the best-performing pairs. The analyses revealed that 12 NI-ML models obtained a testing accuracy of over 91%, which is above the 95th percentile value. The Flower Pollination Algorithm and Extreme Gradient Boost Algorithm pair obtained the highest testing accuracy of 93%.
**Conclusion:**
This study revealed the remarkable performance of the boosting algorithms promoting explainable machine learning in PD detection.

**INDEX TERMS** Parkinson's Disease, Nature-inspired algorithms, Machine Learning classifiers, Feature selection, Classification

## I. INTRODUCTION

PARKINSON'S disease (PD) is a foremost neurodegenerative disease that is caused due to the loss of sensory cells in the substantia nigra, a portion in the midbrain [1]–[3]. This disease affects the nerve cells of this portion that produce dopamine which is further used by our nervous system to exchange messages between sensory cells. Dopamine has a vital role in our daily behaviour and physical movements.

Due to less dopamine production, imbalanced physical movements such as spontaneous tremors and rhythmic muscular tightening cause shaking of several body parts while moving. Again, tremor, the primary manifestation of PD, is also caused by slow movement (bradykinesia) and muscle stiffness (hypertonia). These involve the Basal Ganglia, a brain area related to training, speech and gait patterns. Other than the pre-motor system manifestations, some more early symptoms are– gastrointestinal problems (such as constipation and sluggish food transit through the intestines from the stomach), loss of patient's smelling sense (hyposmia), sleeping disorder (insomnia), Willis-Ekbom disease and immoderate daytime drowsiness, erectile dysfunction in males, lower rate of lubrication in females, trouble attaining orgasm in both males and females, and affective disorders. As dopamine is also involved in emotional feelings, sometimes the patients with PD may experience problems like anxiety, depression, and lack of interest and emotional attachment to something.

PD is the second most commonly diagnosed ailment that affects older adults, specifically those over 60, after Alzheimer's disease [4]. The pervasiveness of PD can be observed in a recent study by the Parkinson's Foundation [5]. As per the Parkinson's Foundation, over 10 million people live with PD globally. PD affects roughly 4% of people before they reach the age of 50, although the likelihood of being affected increases with age [5]. As a result, PD is worrisome not just for the elderly but also for adults. In the United States, the yearly expenditures of PD are projected to reach about \$11 billion, where direct costs of \$6.2 billion are included [6]. Most costs are expended in the latter phases of PD when manifestations are more dominant than ever before [7]. Hence, from a strictly financial viewpoint, any technique that identifies early symptoms of PD (i.e., milder and less acute) would be beneficial in reducing treatment cost. The same can be stated about the quality of healthcare. Because this condition creates serious issues and affects patients' everyday activities in the late stages, early diagnosis of the disease is critical to living healthier lives for as long as feasible.

There is no particular test for diagnosing PD. It is diagnosed by neurologists following the patient's medical history, discussing the patient's signs and manifestations, and a physical and neurological examination. A specialised single-photon emission computerised tomography (SPECT) scan, also known as dopamine transporter scan (DaTscan), may also be recommended by doctors [6]. Even if it may reinforce the assumption that the patient has PD, the manifestations and neurological examination can precisely determine the actual diagnosis. Most patients do not need a DaTscan as it may lead to inaccurate results for those with any condition related to impaired dopamine nerve terminals in the corpus striatum [8]. For screening out other ailments that might be causing patients' complaints, doctors may recommend lab tests like blood tests. Imaging tests, including MRIs, brain ultrasounds, and PET scans, can also be performed to confirm a diagnosis of other disorders [8]. Diagnosing PD may take some time and may involve periodic follow-ups with neurologists specialised in movement disorders to examine the patients' status and manifestation over time [8].

A significant number of diagnostic methods are increasingly being developed. A massive amount of data is processed and stored. A new variety of wireless and wired medical devices are being introduced into clinics, hospitals, and other healthcare institutions. Besides medical experts, computer scientists are also looking to develop robust techniques to detect it at an earlier phase. Because the earlier it is detected, the more likely the patients will benefit from medication. In recent years artificial intelligence (AI) has been applied in diverse problem domains to solve various challenging problems including student engagement [9], virtual reality exposure therapy [10], text classification [11]–[14], cyber security [15]–[18], neurological disease detection [3], [19], [20] and management [21]–[26], elderly care [27], [28], biological data mining [29], [30], fighting pandemic [31]–[37], and healthcare service delivery [38]–[40]. Many nature-inspired (NI) algorithms have been successfully used as diagnostic tools in recent years. These NI algorithms, inspired by various common natural observations or phenomena like behaviours of fish, birds, insects, animals, plants, humans, and natural events, are developed and used to construct PD diagnosis approaches. The feature selection process selects a small subset of optimal features from an enormous collection of features while retaining system performance, resulting in better accuracy. Due to the involvement of many features in machine learning (ML) works, various approaches involving NI algorithms have been developed to address the challenge of reducing the large feature space by removing the inessential and extraneous features [29], [30]. As a result, the training process has an incisive structure without losing the predicted accuracy achieved by employing just the most essential features. Thus, the problem's computing cost and time complexity can be decreased if nature-inspired approaches are used.

Many NI algorithms have been commonly used for feature selection to achieve a better classification with ML techniques for PD detection. For instance, Genetic Algorithms (GA), Ant-Colony Optimisation (ACO) with Support Vector Machine (SVM) [41] [42], GA and Binary Particle Swarm Optimisation (BPSO) with 11 Machine Learning techniques [43], Binary Grey Wolf Optimisation (BGWO) [44] with three ML methods: K-Nearest Neighbors (K-NN), SVM and Naive Bayes (NB) [45]. But some recently proposed algorithms, e.g. Flower Pollination Algorithm (FPA), Jaguar Algorithm (JA), Sine Cosine Algorithm (SCA), and Salp Swarm Algorithm (SSA), which are well enough optimisation algorithms, have not been studied yet in this scope. This study focused on the commonly used NI algorithms and the algorithms mentioned above that have not been studied before; DE, FPA, and SSA performed well with a mentionable outcome. Figure 1 depicts the 13 NI optimisation algorithms used in this study. The contributions of this study are the following:

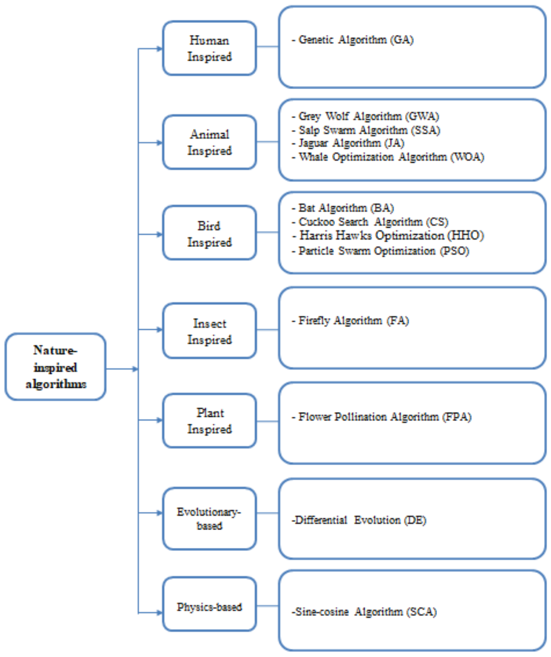

Nature-inspired algorithms
Human Inspired
- Genetic Algorithm (GA)
Animal Inspired
- Grey Wolf Algorithm (GWA)
- Salp Swarm Algorithm (SSA)
- Jaguar Algorithm (JA)
- Whale Optimization Algorithm (WOA)
Bird Inspired
- Bat Algorithm (BA)
- Cuckoo Search Algorithm (CS)
- Harris Hawks Optimization (HHO)
- Particle Swarm Optimization (PSO)
Insect Inspired
- Firefly Algorithm (FA)
Plant Inspired
- Flower Pollination Algorithm (FPA)
Evolutionary-based
-Differential Evolution (DE)
Physics-based
-Sine-cosine Algorithm (SCA)

**FIGURE 1.** Thirteen widely acknowledged nature-inspired algorithms of seven different classes used in this study for feature selection.

- We employed thirteen NI feature selection algorithms, such as Bat Algorithm (BA), Cuckoo Search (CS), Differential Evolution (DE), Firefly algorithm (FA), Flower Pollination Algorithm (FPA), Genetic Algorithm (GA), Jaguar Algorithm (JA), Particle Swarm Optimisation (PSO), Grey Wolf Optimisation (GWO), Harris Hawks Optimisation (HHO), Sine-cosine Algorithm (SCA), Salp Swath Algorithm (SSA) and Whale Optimisation Algorithm (WOA) on a PD dataset. To the best of the author's knowledge, DE, FPA, JA, SCA, or SSA algorithms have not been studied for feature selection in PD.
- In addition to commonly used ML classifiers for PD diagnosis, six classifiers were experimented with, namely Adaptive Boosting Algorithm (AdaB), Gradient Boosting Algorithm (GradB), Extreme Boosting Algorithm (XGB), Stochastic Gradient Descent Algorithm (SGD), Gaussian Naive Bayes (GNB), and Bernoulli Naive Bayes (BNB) which were not previously used in PD diagnosis.
- In each of the NI algorithm's feature selection techniques, the performances of 11 ML classifiers were tested in a separate experiment using the same set of parameters to analyse the effect of feature selection on classification accuracy. This is also compared to the performance of ML techniques without feature selection.
- The results of different ML classifiers were compared for each NI algorithm's feature selection with the same parameters as above.
- Finally, the best NI-ML pairs for PD classification testing performance are ranked to find the best ones.

## II. LITERATURE REVIEW

NI algorithms are among the most popular methods to solve optimisation problems which belong to a group of evolutionary problem-solving techniques inspired by nature. These include Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), Bat Algorithm (BA), Cuckoo Search (CS), Differential Evolution (DE), Firefly algorithm (FA), Harris Hawks Optimisation (HHO), Grey Wolf Optimiser (GWO), Jaguars Algorithm (JA), and Whale Optimisation Algorithm (WOA). The GA mimics natural selection in which the fittest individuals are chosen for reproduction to generate offspring [46]. The PSO is based on swarming where the optimal solution is searched in the solution space [47], [48]. The BA is motivated by the echolocation technique of the microbat [49]. The CS algorithm simulates cuckoo species that lay eggs in the nest of individuals of other species [50]. The DE is a population-based algorithm that works iteratively to improve candidate solutions [51]. The FA is a recently created algorithm inspired by the firefly's flashing characteristic [52]. The HHO algorithm is proposed from the patterns of Harris hawks' chasing the prey, attacking suddenly and grasping [53]. The GWO is a recently proposed algorithm that imitates social actions and the haunting strategy of the grey wolves [54]. The JA is also a newly developed NI algorithm based on the behaviour of jaguars which has great potential in exploitation and exploration [55]. The WOA was proposed in 2016, encouraged by humpback whales' predicate as they prey uniquely [56].

Among many NI algorithms, GA and PSO have been used more to reduce irrelevant data features. Similarly, Support Vector Machine (SVM) is the most common ML technique. Soumaya et al. [41] used GA for feature selection and SVM for PD classification. Goyal et al. [57] implemented a two-stage feature selection where in the first stage GA was used by recursive feature elimination based on SVM classification in the second stage. Afterwards, Pasha et al. [43] employed GA and BPSO for dimensionality reduction of data on 11 ML models for classification, namely LR, lSVM, rSVM, GNB, GPC, KNN, DT, RFMLP, AB and QDA. After preprocessing the data, Ul Haq et al. [42] united the ACO algorithm with Relief for feature selection and SVM was applied for PD classification.

Besides these popular algorithms, many NI algorithms have been used for feature selection with ML classifiers. Rajalaxmi et al. [45] selected the optimal features by binary grey wolf optimisation (BGWO) [44] and compared the features in the sigmoid function for these three ML methods: KNN, SVM and NB. Though SVM showed the highest accuracy of 93.88%, it could not select the minimum number of features. In contrast, KNN selected the minimum number of features but came up second with an accuracy of 92.86%. Then the number of selected features and accuracy obtained from BGWO were compared with two original and two modified versions of bio-inspired algorithms, namely GA, PSO, Binary Bat Algorithm (BBA) and Modified Cuckoo Search Algorithm (MCS). However, the proposed algorithm did not outperform as BBA produced the highest accuracy of 93.6%.

An optimised variety of the crow search algorithm (OCSA) was introduced by Gupta et al. [58] via three standard ML algorithms, namely KNN, DT, and Random Forest (RF). Then the number of selected features, accuracy, and computation time were compared between OCSA via K-NN, DT, RF and actual chaotic crow search algorithm (CCSA), where 20 benchmark datasets were considered. The study concluded that the proposed algorithm gives a better outcome with RF. Another study was carried out by Gupta et al. [59] where they compared an optimised version of a nature-inspired algorithm with its original one in the same year. For feature selection, the authors brought out an optimised cuttlefish algorithm (OCFA) compared with the traditional cuttlefish algorithm, also justified by two ML algorithms: DT and KNN, on selected features where four datasets were used amidst which speech dataset came up with the highest accuracy of 92.194%. A modified grey wolf optimisation (MGWO) algorithm was proposed by Sharma et al. [60] for feature selection where the classification of PD was determined with three frequently used classifiers viz. KNN, DT and RF experimented on four different datasets. After comparing the detection rate, accuracy, and false alarm rate, RF outperformed the other two classifiers. Lastly, the features were selected by MGWO and OCFA. Furthermore, the accuracy comparison between them was highlighted where MGWO selected lesser features and had a higher accuracy of 94.83%. Later that year, Sharma et al. [61] introduced a modified version of the ALO (MALO) algorithm, a binary variant of the algorithm for eliminating irrelevant features to revamp PD classification examined with three classifiers viz. KNN, RF, and DT. The study tested with two datasets: speech and voice; no handwriting dataset was included like in the previous studies mentioned above. Finally, the minimised number of features, accuracy and computation time of MALO were compared with traditional CFA and OCFA. Though MALO performed with the highest accuracy for the speech dataset, it could not outperform the voice dataset. Afterwards, Sehgal et al. [62] presented a Modified Grasshopper Optimisation Algorithm (MGOA) for detecting the optimal set of features. This study used the same four datasets and three classifiers mentioned in [60] and also found RF as the best performer in terms of detection rate, accuracy, and false alarm rate amongst the three classifiers. In addition, with OCFA, this study included MGWO for performance comparison, where the proposed MGWO selected the lowest number of features and highest accuracy. Later in the same year, Durgut et al. [63] used binary versions of the artificial bee colony algorithm to reduce irrelevant features and deployed KNN for classification and SVM in the second stage to compare individual results. Dash et al. [64] employed a chaotic firefly algorithm integrated with a kernel-based NB algorithm for discriminant features, and five classifiers were brought for classification.

#### 1) Research gap

It can be concluded from the literature review that the following different criteria have led to the studies:

- One or two NI algorithms for feature selection.
- Few ML algorithms for classification.
- Very few numbers of best-performed NI-ML pairs.

As mentioned in the literature review, a very recent study where Pasha et al. employed 11 ML classifiers but investigated only two NI algorithms for feature selection, where GA and AdaBoost were the best pair with an accuracy of 90.7%. Most of the studies investigated one or a few NI algorithms, which does not reveal the significant role of NI algorithms in dimensionality reduction. Moreover, no studies were conducted using boosting algorithms for PD classification to the author's best knowledge. Therefore, addressing these problems, this study overcomes the research gaps by considering 13 NI algorithms for feature selection. It is worth mentioning that 5 of these NI algorithms have not been studied previously, among whom FPA, SSA, and DE rose with a notable result. This study revealed the great effect of boosting algorithms for PD classification for the first time. So, this research work will establish a substantial opportunity for the researchers of this area to understand which NI algorithm performs better with which ML classifiers and their domination in more accurate PD detection. As a result, this study will play a significant role in PD diagnosis at an early stage.

## III. METHODOLOGY

### A. DATASET

Many studies have used publicly available datasets from different repositories for PD diagnosis employing NI algorithm for feature selection and ML algorithms for classification. In this research study, the data set is collected from UCI ML Repository [65] which was also used in some previous studies mentioned in the literature section [43], [63]. The dataset is prepared from 252 patients whose speeches were recorded at the Department of Neurology at Istanbul University, where 188 (107 men and 81 women) were PD patients from the age 33 to 87 and 64 (23 men and 41 women) were healthy individuals from the age 41 to 82. The entire data collection process was brought out following the instructions of expert physicians. Then the dataset is prepared by the sustained phonation of the vowel 'a' collected from each patient, asking them to repeat it three times. During this process, the microphone was set to 44.1 kHz. Then several speech signal processing algorithms like Time-Frequency features, Wavelet Transform features, Vocal Fold based features, Mel Frequency Cepstral Coefficients (MFCC), and Tunable Q-factor Wavelet Transforms (TWQT) were extracted from the patient's speech recordings. Among the 754 features, 752 features have floating-point values, two features have binary values, and 1 part has a certain ranged value (0-2). The last feature is one of the features having binary values which represent a class or the decision of this dataset. The

ML classifiers can use this variable to identify PD patients clinically. The main characteristics of the dataset are:

- Type of task: Classification
- Characteristics of attributes: Integer, Real
- Instances or row count: 756
- Attributes or column count: 754
- Missing Value: N/A

## B. NATURE-INSPIRED ALGORITHMS

In reality, it's unusual that all of the variables in a dataset are relevant for developing a machine learning model. Adding extraneous variables lowers the model's generalisation ability and reduces a classifier's overall accuracy. Moreover, adding more features to a model escalates the total complexity of the model. A feature selection algorithm is used when the number of features in the data is too large to compute, or additional features result in worse evolution metrics. Since machine learning works entail many parts, various nature-inspired algorithms have shown promising outcomes in PD diagnosis. In this article, we have experimented with 13 feature selection algorithms described along with their mathematical models in this current section.

**Genetic Algorithm (GA)**

In the 1970s, John Holland [66] introduced a new meta-heuristic optimisation procedure which is known as a genetic algorithm (GA). However, it was improved by Goldberg et al. [67] in 1989. GA generally provides optimal or near-to-optimal results by exploiting several effective operators like crossover, mutation and selection. In GA, a set of positions has termed a gene stored in a chromosome (known as a solution). Considering all old chromosomes, crossover generates new chromosomes according to the crossover probability. The mutation operator handles the diversity of the solutions and changes the initial value of one or more chromosomes based on a mutation probability.

**Bat Algorithm (BA)**

Bat algorithm was proposed by Yang [49] in 2010 based on micro bats' echolocation behaviour. There are four parameters involved with the bat algorithm, namely, velocity (v), position (x), pulse rate (r), and loudness (A). In each iteration, the frequency of each bat is adjusted by eq 1 and the velocity and position will be updated by eq 2 and eq 3 respectively.

$$f_i = f_{\min} + \left(f_{\max} - f_{\min}\right)\beta \tag{1}$$

$$v_i^t = v_i^{t-1} + \left(x_i^t - x_{\text{gbest}}^t\right) f_i \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{3}$$

If $rand > r_i$ where $0 \leq rand \leq 1$, a best solution will be selected. Then around the selected best solution, a local solution will be generated. If $rand < A_i$ and $f(x_i) < f(x_*)$, then the solution is accepted, $r_i$ is increased and $A_i$ is reduced by eq 4 and eq 5. Then the bats will be ranked and the current best solution will be generated.

$$r_i^{t+1} = r_i^0 \left[1 - e^{\gamma t}\right] \tag{4}$$

and

$$A_i^{t+1} = \alpha A_i^t, \tag{5}$$

where $\alpha = \gamma = 0.9$.

**Cuckoo Search Algorithm (CS)**

CS is a meta-heuristic algorithm introduced by Yang and Deb [50] in 2009. This algorithm is inspired by some cuckoo species that lay eggs in the nest of other birds, also called host birds. After initialising the CS parameters namely the number of available host nests, i.e., population (n), probability of finding the cuckoo's egg by the host bird (Pa) and maximum number of iterations ($\text{Max}_t$), random walk is done by cuckoo using Levy Flight (eq 6) to generate the new solution in the host nest and the fitness, $F_i$ is generated. Levy Flight is performed as:

$$x_i^{t+1} = x_i^t + \alpha \oplus Levy(\lambda), \tag{6}$$

where $\alpha$ is the step size, $\lambda$ is the Levy exponent ($\lambda$ =1.5), $x_i^t$ current location. Random step length is calculated from Levy distribution. Levy($\lambda$) is calculated as eq 7.

$$Levy \;\sim\; u = t^{-\lambda} \;\; (1 < \lambda \leq 3) \tag{7}$$

After that, a nest is chosen randomly from n, and the fitness of the cuckoo is compared with the host nest. If the fitness of the cuckoo is greater than the fitness of the host, then the host egg is replaced with a cuckoo egg, i.e., the current solution. If the host bird identifies the cuckoo egg, the worst case is generated, and the egg is thrown. A new nest is built near the old one, which can be mathematically expressed as eq 8.

$$x_i^t = x_i^t + \epsilon\left(x_i^t - x_j^t\right) \tag{8}$$

where $\epsilon$ is a random distribution from 0 to 1; in this case, a new solution is generated again by the Levy Flight. The current solution is ranked in each iteration until the stopping condition is met.

**Differential Evolution Algorithm (DE)**

In the 1990s, DE was introduced by Storn and Price [51] [68]. The chosen values of the parameters greatly impact the performance of the optimisation of this algorithm. The three significant parameters are: the population size denoted as NP, the crossover probability, CR and the differential weight, F.

Let $f : R^n \rightarrow R$ denote the fitness function to be minimised. Let $x \in R^n$ indicate an agent in the population. The fundamental DE algorithm can be outlined below:

- Select the parameters. $F \in [0, 2]$, $CR \in [0, 1]$ and $NP \geq 4$
- All agents x will be initialised with the random position in the state space.

- The following steps will be repeated until a termination requirement is met (e.g., the number of iterations completed or enough fitness is achieved):
- Select three agents, namely an (also called base vector), b and c, randomly from the population. These agents have to be different from each other and also from x.
- Select an index $R \in \{1, \ldots, n\}$ randomly where n denotes the dimensionality of the problem.
- New position $y = [y_1, \ldots, y_n]$ of the agent is computed as below:
- For every $i \in \{1, \ldots, n\}$, select a random number, $r_i$ between 0 to 1
- If $r_i < CR$ or $i = R$ then set $y_i = a_i + F \times (b_i - c_{i)}$ else set $y_i = x_i$.
- If $f(y) \leq f(x)$ then substitute the agent x within the population with the improved candidate solution y.
- Repeat this for each agent x in the population.
- From the population, select the agent who has the best fitness. Then Please return it to be the best-formed candidate solution.

The trial vector generation technique as well as the control parameter selection has a significant impact on DE's performance in a certain optimisation problem.

**Firefly Algorithm (FA)**

Firefly is a meta-heuristic algorithm inspired by the fireflies' flashing behaviour which was created by Yang et al. [52] in 2010. There are three idealised assumptions in the basic FA:

- Fireflies are attracted to each other regardless of their gender.
- A brighter firefly attracts a less bright firefly. When the distance between two fireflies increases, brightness and attractiveness decrease. If a firefly has the same brightness as others or there is no such firefly brighter than it, then it moves randomly.
- The objective function determines a firefly's brightness.

The decrease in brightness due to distance can be expressed as the following the inverse square law,

$$I < \frac{1}{r^2} \tag{9}$$

The light intensity, $I$ with a distance of $r$, can be calculated as eq 10.

$$I = I_0 e^{-\gamma r^2} \tag{10}$$

, where $I_0$ is light intensity respect the source. Likewise, the brightness can be calculated as eq 11

$$\beta = \beta_0 e^{-\gamma r^2} \tag{11}$$

In this algorithm, a feasible solution to an optimisation problem is symbolised as the position of a firefly. The position vector of a firefly at $t^{\text{th}}$ iteration is updated by eq 12.

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{\text{ij}}^2} \left( x_j^t - x_i^t \right) + \alpha \epsilon_i^t, \tag{12}$$

where $\beta_0 > 1$ is the attractiveness of $x_j$ at $r_{\text{ij}} = 0$. even though there is no optimal solution explicit in the equation, the fittest solution is chosen from a population of n solutions at each iteration. Here, $\gamma$ is an algorithm parameter that controls how much the updating process is affected by the distance between two fireflies, whereas controls the random movement's step length, rand is a random number from a uniform distribution with values ranging from 0 to 1.

**Flower Pollination Algorithm (FPA)**

FPA is a bio-inspired algorithm; however, it is not based on swarm intelligence. FPA was developed using flowering plants' pollination processes and features by Yang [69] in 2012. In FPA, a pollen particle's location is expressed as a solution vector, and the pollination process over a long distance can be represented as eq 13.

$$x_i^{t+1} = x_i^t + \gamma L(\lambda) \left( g_* - x_i^t \right), \tag{13}$$

where $g_*$ is the fittest solution found thus far at $t^{\text{th}}$ iteration and $\gamma$ is a scaling parameter. $L(\lambda)$ can be regarded as a randomly generated vector taken from a Levy distribution with an exponent of $\gamma$ in the above equation. Other pollination characteristics, like flower constancy, can be expressed by eq 14

$$x_i^{t+1} = x_i^t + \epsilon \left( x_j^t - x_k^t \right), \tag{14}$$

where $x_j^t$ and $x_k^t$ are pollen of identical plant species from the distinct flowers. If $x_j^t$ and $x_k^t$originate from the identical species or are picked from the identical population, this is referred to as a local random walk if we take $\epsilon$ as a random number from a uniform distribution with values ranging from 0 to 1. FPA is simplified by producing only one pollen gamete in each flower.

**Grey Wolf Optimisation (GWO)**

GWO is a metaheuristic algorithm proposed by Mirjaliali et al. [54] in 2014. GWO is inspired by grey wolves' special social hierarchy and hunting mechanism, where their ability to work in a social pack increases their potential.

- **Chasing or pursuing the prey:** When the target enters the territory, the wolves will chase the target towards the waiting wolves to make the kill.
- **Encircling the prey:** During hunting, the wolves encircle the prey. This can be expressed by eq 15 and eq 16.

$$\overrightarrow{D} = \left| \overrightarrow{C}.\overrightarrow{X_p} - \overrightarrow{X}(t) \right|, \tag{15}$$

and

$$\overrightarrow{X}(t+1) = \left| \overrightarrow{X_p}(t) - \overrightarrow{A}.\overrightarrow{D} \right|, \tag{16}$$

where t is the current iteration, $\overrightarrow{X_p}(t)$ is the position of the prey, $\overrightarrow{X}$ is the position of the grey wolf and $\overrightarrow{A}, \overrightarrow{C}$ are coefficient vectors.

The above equations are used to update the grey wolves' position in compliance with the position of the prey.

- **Hunting the prey:** The hunting process is guided by the alpha wolf i.e. the fittest solution. The mathematical model for the hunting process can be depicted as follows:

$$\overrightarrow{D_\alpha} = \left|C_1.\overrightarrow{X_\alpha} - \overrightarrow{X}(t)\right|, \tag{17}$$

$$\overrightarrow{D_\beta} = \left|C_2.\overrightarrow{X_\beta} - \overrightarrow{X}(t)\right|, \tag{18}$$

$$\overrightarrow{D_\delta} = \left|C_3.\overrightarrow{X_\delta} - \overrightarrow{X}(t)\right|, \tag{19}$$

$$\overrightarrow{X_1} = \left|\overrightarrow{X_\alpha} - A_1\overrightarrow{D_\alpha}\right|, \tag{20}$$

$$\overrightarrow{X_2} = \left|\overrightarrow{X_\beta} - A_2\overrightarrow{D_\beta}\right|, \tag{21}$$

and

$$\overrightarrow{X_3} = \left|\overrightarrow{X_\delta} - A_3\overrightarrow{D_\delta}\right|, \tag{22}$$

where $X_\alpha$, $X_\beta$ and $X_\delta$ are the position of alpha, beta and delta wolf respectively. The positions of the grey wolves are updated by eq 23

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{23}$$

- **Attacking the prey:** Alpha wolf will finish the hunt by attacking the prey. The mathematical expression for this process is as follows:

$$\overrightarrow{A} = 2.\overrightarrow{a}.\overrightarrow{r_1} - \overrightarrow{a}, \tag{24}$$

and

$$\overrightarrow{C} = 2.\overrightarrow{r_2} \tag{25}$$

When the prey stops moving, wolves will attack it to finish the hunting process. This is modeled by decreasing the $\overrightarrow{a}$ from 2 to 0 during the iterations. As $\overrightarrow{a}$ decreases, $\overrightarrow{A}$ also decreases. $A < 1$ leads the wolf to attack the prey.

**Searching for prey:** If $A > 1$, then this will diverse the wolves from the prey, and they will search for other prey. C assists in putting some more weight on the prey to make it difficult to the wolves to find it i.e. if $A > 1$, then emphasize and if $C < 1$, then reduce importance.

**Harris Hawks Optimisation (HHO)**

HHO is a population-based recent algorithm proposed by Heidari et al. [53] in 2019. It is inspired by exploring the prey, surprise pounce and attacking strategies of Harris hawks. HHO has different phases which are described below.

Exploration phase: Harris hawks is an intelligent bird that can trace and detect the prey with its powerful eyes. If no prey is found, Harris hawks will wait, observe and monitor the site. In this algorithm, Harris hawks is a candidate solution and the best candidate solution in each step is the prey. Harris hawks haunt randomly on some sites and wait to detect prey following two strategies.

1. Solution is generated based on the position of other family members and prey which is modelled in eq 26 for the condition q< 0.5.

2. Solution is generated based on perching tall trees which is inside the family's home range. This is modeled as eq 26 for the condition q>0.5.

$$X(t+1) = \begin{cases} X_{\text{rand}}(t) - r_1\left|X_{\text{rand}}(t) - 2r_2X(t)\right| & q \geq \\ (X_{\text{rabbit}}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < \end{cases} \tag{26}$$

]

where $X(t+1)$ is the position vector of hawks, $X_{\text{rabbit}}(t)$ is the position of prey, $X(t)$ is the hawks' current position vector, $r_1$, $r_2$, $r_3$, $r_4$ and q are random numbers within (0,1), LB and UB are the upper and lower bounds of variables, $X_{\text{rand}}(t)$ is a randomly selected hawk from the current population and $X_m(t)$ is the average position of the the current population of hawks. The average position of current hawks' population is obtained by eq 27

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \tag{27}$$

where $X_i(t)$ indicates the location of each hawk in iteration t and N denote the total number of hawks.

Transition: After the exploration phase, this algorithm transfer to the exploitation phase where there are different strategies based on the prey's escaping behaviour. The energy of the prey is mathematically modelled as eq 28

$$E = 2E_0(1 - \frac{t}{T}) \tag{28}$$

where E is the prey's escaping energy, and T indicates the maximum number of iterations, t is the current iteration, and E0 is the initial state of its energy inside the interval [-1, 1].

Exploitation phase: The Harris hawks perform the surprise pounce in this phase. As the prey attempt to escape, a parameter r is considered to express the chance of escaping. When r>0.5, the prey has a chance of successfully escaping and r<0.5 the prey is not successful to escape. The following four attacking strategies by this predator are proposed in this algorithm.

1. Soft besiege: If r≥0.5 and |E|≥0.5, the prey has sufficient escaping energy. When it attempts to escape, the Harris hawks surround it softly to make it more tired and then perform the attack. This can be mathematically modelled as eq 29

$$X(t+1) = \Delta X(t) - E|JX_{\text{rabbit}}(t) - X(t)| \tag{29}$$

$$\Delta X(t) = X_{\text{rabbit}}(t) - X(t) \tag{30}$$

Where $\Delta X(t)$ is the difference between the position vector of the prey and the current location, $r_5$ is a random number inside (0, 1), and J = 2(1-$r_5$) represents the random jump the energy of the prey all over the escaping stage.

2. Hard besiege: If r≥0.5 and |E|<0.5, the prey is tired and has low escaping energy. The Harris hawks perform the sudden attack. The current positions are updated using eq 31

$$X(t+1) = X_{\text{rabbit}}(t) - E|\Delta X(t)| \tag{31}$$

3. Soft besiege with progressive rapid dives: In this case, |E|≥0.5 like the soft besiege but r<0.5. As the prey has sufficient energy to effectively escape a soft besiege is made before the sudden pounce. This strategy is more intelligent than the previous one. The Harris Hawks can decide their next move to perform a soft besiege following eq 32.

$$Y = X_{\text{rabbit}}(t) - E|JX_{\text{rabbit}}(t) - X(t)| \tag{32}$$

It is considered that hawks will dive using eq 33

$$Z = Y + S \times LF(D) \tag{33}$$

where $D$ is the dimension of the problem and $S$ is a random vector by size $1 \times D$ and $LF$ is the levy flight function.

This behaviour can be mathematically modelled as eq 34

$$X(t+1) = \begin{cases} \text{Y} & \text{if F}(Y) < F(X(t)) \\ \text{Z} & \text{if F}(Z) < F(X(t)) \end{cases} \tag{34}$$

3. Hard besiege with progressive rapid dives: When |E|<0.5 and r<0.5, the prey has not sufficient energy to escape and a hard besiege is made before the sudden pounce. This strategy is more like the soft besiege, just this time, the hawks attempt to reduce the distance of their average location with the escaping prey. Therefore, the final rule for this case is the same as the previous strategy as eq where Y and Z are produced using new rules as eq 35 and eq 36and eq 36

$$Y = X_{\text{rabbit}}(t) - E|JX_{\text{rabbit}}(t) - X_m(t)| \tag{35}$$

$$Z = Y + S \times LF(D) \tag{36}$$

**Jaguar Algorithm (JA)**

Taking into account the Jaguars' special ability to move in the direction of their prey shortly, Chen et al. [55] (2015) proposed the Jaguar Algorithm (JA) having strong abilities both in exploitation and exploration as these are executed separately, unlike the traditional procedures. In addition, when a jaguar discoveries prey in its territory, it not only moves to the prey in a short time but also attacks carefully after nearing the prey. Adopting these behaviours (hunting, learning and territoriality) of jaguars, JA has the strong ability to reach the global optimum solution.

The algorithm starts with initialising the unique adaptive parameter '$d$' (step) in the domain. In order to find the best moving direction from the current position, jaguars checks two directions as follows:

$x_c \pm d * 2^i,\ \ i = 0, 1, 2, \ldots$

where $x_c$ denotes the current position. The potential direction for the jaguar is selected by comparing the fitness values in both directions.

Learning is another prey-searching way of jaguars where four sides are considered when jaguars move two dimensions at a time. Four sides { ( + , + ), ( - , - ), ( + , - ),( - , + )} are tested pairwise ($\{(+,+),\ (-,-)\}$ and $\{(+,-),(-,+)\}$) to obtain the potential direction. After executing the learning procedure by testing these two classes respectively, '$d$' (step) is updated by $d_{\text{new}} = \frac{d_{\text{old}}}{2}$. The next round of hunting or learning is continued exploiting the new '$d$' until '$d$' is too small to differentiate.

Finally, the territory is restructured, and jaguars reborn in the decision space excluding other parts.

**Particle Swarm Optimisation (PSO)**

PSO is a population-based optimisation procedure invented by Eberhart and Kennedy [47] (1995) reviewing the social behaviour of several natural swarms like fish schooling and folks of birds. In this procedure, each possible solution is presented by a particle and the set of all solutions or populations is called a swarm. The execution of the algorithm is commenced by initiating random people from particles. Each possible solution ($j-$th) is determined with the help of the corresponding particle's position vector.

$\left(y_j^k\right)$ and velocity vector $\left(v_j^k\right)$. For all iterations, the acceleration direction of every particle is adjusted by the particle's personal best $\left(\text{pbest}_j\right)$ position and global best $\left(\text{gbest}_j\right)$ position obtained up to this point in the population. The position and velocity of each solution is upgraded by the following equations ( eq 37 and 38):

$$y_j^{k+1} = y_j^k + v_j^{k+1} \tag{37}$$

and

$$v_j^{k+1} = wv_j^{k+1} + c_1 r_1(\text{pbest}_j - y_j^k) + c_2 r_2(\text{gbest}_j - y_j^k) \tag{38}$$

where $w$ is inertia weight, $c_1, c_2 > 0$ are coefficients of acceleration and $r_1, r_2 \sim U(0,1)$ uniformly distributed.

**Sine Cosine Algorithm (SCA)**

SCA is another novel population-based optimisation procedure designed by Mirjalili [70] (2016) utilising the characteristics of sine and cosine trigonometric functions. The SCA starts with generating several initial solutions randomly and then, leads to the best solution by adopting two key search policies, namely, global exploration and local exploitation. In SCA, the position of each the solution is upgraded with the help of the following mathematical equations for both strategies:

$$y_j^{k+1} = y_j^k + c_1 \times \sin(c_2) \times \left|c_3 p_j^k - y_j^k\right|, \qquad c_4 < 0.5 \tag{39}$$

and

$$y_j^{k+1} = y_j^k + c_1 \times \cos(c_2) \times \left|c_3 p_j^k - y_j^k\right|, \qquad c_4 \geq 0.5 \tag{40}$$

where $y_j^k$ is the current position of the solution, $p_j^k$ denotes the position of the best-found solution, $c_1, c_2, c_3, c_4$ are random numbers. The parameter $c_1$ indicates the moving direction while $c_2$ indicates the moving distance to the next updated position. The parameter $c_3$ is a random weight to balance in calculating the distance in every iteration and, finally, $c_4$ measures equal switches between the sine and cosine element.

**Salp Swarm Algorithm (SSA)**

Inspiring from the swarming behaviour of salps during navigating and searching in oceans, Mirjalili et al. [71](2017) proposed a novel optimization technique called Salp Swarm Algorithm (SSA). During foraging in oceans, all slaps form a swarm called a salp chain. The front member in the salp chain is the leader and all the remaining salps are the followers. All the followers update their position following the leader while the leader updates the position as follows:

$$y_j^k = W_j + c_1 \{l_j + c_2 (u_j - l_j)\}, \qquad c_3 \geq 0 \tag{41}$$

and

$$y_j^k = W_j - c_1 \{l_j + c_2 (u_j - l_j)\}, \qquad c_3 < 0 \tag{42}$$

where $y_j^k$ denotes the leader position in the $j-$th dimension under $k-$th iteration, $W_j$ denotes the food source's position in the $j-$th dimension, $u_j$ indicates the upper bound in the $j-$th dimension, $l_j$ indicates the lower bound in the $j-$th dimension and $c_1, c_2, c_3$ are random numbers. All the followers update their positions by using eq 43

$$y_j^i = \frac{1}{2}\left(y_j^i + y_j^{i-1}\right) \tag{43}$$

where $i \in \{2, 3, \ldots\}$ and $y_j^i$ denotes the position of the $i-$th followers in $j-$th dimension.

**Whale Optimisation Algorithm (WHO)**

Mirjalili et al. [56] proposed a whale optimisation algorithm in 2016 which mimics the hunting technique of the humpback species of whale. Humpback whale eats krill and small fishes. After discovering the prey, they dive up to 12 meters deep into the sea and create bubble nets to catch their prey. This is called the bubble net feeding method. There are three different phases of this hunting process of humpback whales.

**Searching for prey:** Coefficient vector, A is used to search for prey where A is assigned a random value>1. If A>1, then the new individual is far from the prey.

$$C = 2 * r, \tag{44}$$

$$a = \frac{2 - 2 * t}{\text{MaxT}}, \tag{45}$$

$$A = 2 * a * r - a, \tag{46}$$

$$D = \left|C * X_{\text{rand}}(t) - X(t)\right|, \tag{47}$$

$$X(t+1) = X_{\text{rand}}(t) - A * D \tag{48}$$

Once the best search agent is defined, other search agents will update their positions towards the best search agent.

**Encircling prey:** Once humpback whales recognise their prey and its The location they encircle the prey. Target prey is the best candidate solution. This algorithm assumes the current optimal solution as the prey position.

$$D = \left|C * X^*(t) - X(t)\right| \tag{49}$$

$$X(t+1) = X^*(t) - A * D \tag{50}$$

**Attacking prey:** The attacking method is done by shrinking, encircling mechanism. The value of A is decreased. When A<1, this means the agent is approaching the current optimal solution. For updating the position first, the distance between the whale and the prey is calculated. The process can be expressed as follows.

$$D^{'} = \left|X^*(t) - X(t)\right|, \tag{51}$$

$$X(t+1) = D^{'}.e^{\text{bl}}.\cos(2\pi l) + X^*(t), \tag{52}$$

$$X(t+1) = \begin{cases} X^*(t) - A.D & , p < 0.5 \\ D^{'}.e^{\text{bl}}.\cos(2\pi l) + X^*(t) & , p \geq 0.5 \end{cases} \tag{53}$$

### C. MACHINE LEARNING ALGORITHMS

After selecting the optimum feature set by employing 13 different NI algorithms, we distinctly applied 11 ML classifiers with each NI algorithm. Amongst these classifiers, Linear Regression (LR), SVM, RF, K-NN and DT have been widely used in many studies as a classifier or a group of classifiers. Along with these commonly used classifiers, we implemented six newly used classifiers, namely Adaptive Boosting Algorithm (AdaB), Gradient Boosting Algorithm (GradB), Extreme Boosting Algorithm (XGB), Stochastic Gradient Descent Algorithm (SGD), Gaussian Naive Bayes (GNB), Bernoulli Naive Bayes (BNB) in this research area. Among these algorithms, the boosting algorithms performed remarkably on the assessment criteria. In this section, all these 11 ML classifiers are described shortly with their contribution to classification with respect to different types of data.

One of the simplest ML algorithms is linear regression (LR) [72]. A linear relationship between one or more independent variables (predictor variable) with one dependent variable (target variable) is shown in LR, thus the name. Because LR reveals a linear relationship, it determines variation in the dependent variable concerning the variation in the independent variable. Practically, other algorithms are more suitable than LR for classification problems as LR works with continuous value, whereas classification problem requires a discrete value. DT [73] is a supervised learning algorithm that is most commonly employed to solve classification problems, although it can also solve regression problems. In this tree-structured classifier, internal nodes represent dataset attributes, decision rules are represented by the branches, and each leaf node provides the output. Decision trees need less data preparation work than other techniques. SVM [74] is one of the most well-liked supervised learning algorithms for solving classification problems. The purpose of the SVM algorithm is to generate the finest line or decision boundary (also called a hyperplane) for categorising n-dimensional space into distinctly classified data points. SVMs are especially popular because of their ability to operate multiple categorical and continuous variables. For solving classification problems, RF [75] is one of the best algorithms that generate decision trees from data samples, obtain predictions from each of them and then select the best solution. It is based on the idea of ensemble learning. Since it averages the results to reduce over-fitting, it performs better than a single decision tree. By using KNN [76], the model's performance can be improved, and complex problems can be solved by merging multiple classifiers. KNN algorithm saves the training dataset, and based on the similarity with the existing data; it classifies a new data point. This implies that incoming data can be quickly classified into a suitable category using the KNN method. This makes the KNN method significantly faster than other training-based algorithms like SVM and LR.

**Boosting Algorithm**

The AdaB algorithm was originally named AdaBoost.M1 by the algorithm's creators, Freund and Schapire [77]. It's been termed discrete AdaBoost in recent years as it is utilised for classification rather than regression. AdaB is a machine learning algorithm that can be applied to boost the act of any other ML algorithm. The best use of AdaB is to improve decision tree performance on binary classification problems. Decision trees with one level are the most suitable and commonly used method with AdaB. These trees are known as decision stumps because they are so short and only have one classification decision. Each instance is weighted in the training dataset. For each instance, the initial weight is set as eq 54

$$\text{weight}\,(x_i) = \frac{1}{n} \tag{54}$$

$x_i$ is the $i-th$ instance and n is the total number of training instances.

GradB algorithm belongs to the supervised branch of machine learning that is a tree-based algorithm [78]. The errors in ML algorithms are mainly categorised into two types, i.e. Variance error and Bias error. GradB is one of the boosting techniques that are used to minimizing the model's bias error. The base estimator cannot be specified in the GraB technique, unlike the AdaB algorithm. Decision Stump is the base estimator of the GradB algorithm, which is fixed. In this algorithm, the n_estimator can be tuned like the AdaB algorithm. However, the default value of n_estimator for GradB is 100 if the value of n_estimator is not specified. The GradB approach can be used to predict continuous target variables and categorical target variables.

Tianqi Chen first developed the extreme Gradient Boosting Algorithm in 2014, and Chen and Carlos Guestrin detailed it in their article published in 2016 [79]. It also belongs to the class of ensemble ML algorithms like AdaB and GradB algorithm. Like Gradient Boosting, XGB uses decision trees as its base estimators. DT models are used to construct ensembles. To address the prediction errors resulting from prior models, trees are inserted one at a time and fitted. Models are fitted using a gradient descent optimisation approach and any arbitrary differentiable loss function. XGB is designed to be highly effective and computationally efficient. Execution speed and model performance are the two important reasons to employ this algorithm. Unlike conventional GradB, XGB builds trees in its way, with the finest node splits determined by the Similarity Score(SS), gain and Previous Probability(PP) in equation 55

$$SS = \frac{\left(\sum_{i=1}^{n} \text{Residuals}_i\right)^2}{\sum_{i=1}^{n} \left[PP_i \times (1 - PP_i)\right] + \lambda} \tag{55}$$

**Stochastic Gradient Descent Algorithm**

The SGD is one of the simplest yet most effective optimisation techniques that is used to learn discriminative linear classifiers using convex loss functions like LR and SVM. Though it is easy to implement SGD, its sensitivity to feature scaling and multiple hyperparameters requirements could not seek this field's researchers' attention. Nevertheless, it's been a long time since SGD has been included in the ML Community; it has taken considerable attention in recent years regarding large-scale problems. This algorithm updates the coefficients for each instance instead of at the end of the instances. That is why it's been successfully applied to large-scale datasets.

**Naive Bayes classifiers**

Naive Bayes classifiers are a set of classification algorithms based on Bayes' Theorem with the naive conditional independence assumption. Each feature is assumed to be independent and equal. Bayes' theorem can be mathematically expressed as eq. 56 where $x_1$ to $x_n$ are dependent feature vectors and y is class variable [80].

$$P(y|x_{1,\ldots,}x_n) = \frac{P(y)P(x_{1,\ldots,}x_n|y)}{P(x_{1,\ldots,}x_n)} \tag{56}$$

A variant of Naive Bayes classifiers is Gaussian Naive Bayes which allows continuous data. It is named as follows the Gaussian normal distribution. The features' likelihood is considered to be Gaussian (eq. 57) [80].

$$P\left(x_i|y\right) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left\{-\frac{\left(x_i - \mu_y\right)^2}{2\sigma_y^2}\right\} \tag{57}$$

In contrast with the GNB method, Bernoulli Naive Bayes algorithm allows discrete data distributed as per Bernoulli distribution. The features are assumed to be binary-valued, e.g. 0 or 1, yes or no, etc. To handle any other kind of data, a BernoulliNB instance may binarize its input [80]. The basis for the BNB decision rule is:

$$P(x_i|y) = P(i|y)\,x_i + (1 - P(i|y))\,(1 - x_i) \tag{58}$$

## IV. EXPERIMENTAL ANALYSIS

The experimental analysis is split into several parts. At first, a discussion of the findings of a comparative examination of 11 classification algorithms trained and tested on all 753 characteristics of PD data is provided. The next part discusses the findings of a statistical comparison of 11 ML classifiers trained on the optimum set of attributes from 13 NI algorithms. Then, the efficiency of 13 NI algorithms in reducing the PD data set's total size is explored compared to machine learning algorithms. Lastly, a comparison of the best-performing pairs for NI algorithms feature selection and machine learning classifiers are also provided.

All experiments were conducted on a system equipped with an Intel Core i5 CPU and an Nvidia Geforce 920Mx GPU. Python was chosen as the implementation language for this experiment and the anaconda distribution. Scikit learn was used to implement machine learning algorithms. Eighty per cent of the data in the datasets were used for training purposes, while the remaining twenty per cent were used for testing purposes.

### A. PERFORMANCE METRICS

Numerous assessment criteria are used to assess feature selection and machine learning classification performance. Confusion matrices with True positive (TP), True Negative (TN), False positive (FP), and False Negative (FN) values were examined first for the final classification. Following that, several additional standard measures were computed. The following is a quick description of the assessment metrics utilised in this work:

**Accuracy** is the metric of how many often guesses are correct, both TN and TP divided by the total number of predictions.
**Recall** is the measurement of positive prediction among retrieved which retribution by the false negative.
**Precision** is the ratio of correct prediction with total positive prediction
**F1-Score** is the harmonic mean of both precision and recall. Thus, it takes both FP and FN into the computation.
**Fitness Function** measures a specified classification or selection approach to attaining the defined goals. In this article, the result of a KNN prediction is utilised, where K=3.
**Cross-Validation Mean Score** is the average of the results from each K-fold training and testing. For this experiment, 5-Fold cross-validation with F1-Score as scoring metrics is implemented.

### B. PERFORMANCE OF ML CLASSIFIER WITHOUT FEATURE SELECTION

A comparison is needed for several feature selection and classification problems in this study. Since it is a binary classification, accuracy and F1-score are selected as evolution measures for comparisons. Furthermore, for comparison, only testing results are considered since training does not reflect real-world performance. At first, the evolution measures of the eleven ML algorithms are computed with both training and testing sets of the data, which is shown in figure 2. No feature selection algorithm is applied in this step. Among these classifiers, ensemble algorithms work better than others in testing and treating scenarios. Random Forest, AdaB, GradB, and XGB algorithms achieved a testing accuracy of 83.60%, 88.360%, 86.77% and 85.71%, respectively. For KNN classifiers, the value of K is set at 2 to 10, and K=3 achieved the best results among those. Hence in this article, the KNN is referred to as the K=3 classification. After the ones mentioned above, LR, DT, BNB and GNB results are the most prominent. Again, the AdaB classifier achieved the highest testing F1 score of 88% with a cross-validation mean of 80.86%, whereas XGB achieved the highest cross-validation mean of 82.02%. NI algorithm has been shown to offer substantial answers to various real-world situations, including the one selected for this study. Thirteen nature-inspired algorithms were considered feature selection algorithms for the dimensionality reduction of data.

### C. PERFORMANCE OF ML CLASSIFIER WITH NI ALGORITHM FEATURE SELECTION

To test the influence of feature selection for the PD dataset, each NI algorithm was trained with the ML classifiers previously described. The number of features selected by each NI algorithm is shown in table 1. Each feature selection algorithm is trained for 100 iterations to achieve the best fitness outcomes. For the experiments, the number of components was set between 10 and 25. Figure 3 depicts the relationship between fitness and iteration. Because many of the training accuracy and F1 Score is 100 per cent, one key component of the analysis is testing results, which are mentioned for measuring performance.

#### 1) BA

Several values have been tried with the trial and error method for BA to find out the best combination for optimum performance. In this study, the final value for maximum frequency is set at 5, and the minimum frequency is set at 0 with maximum loudness of 3 and a maximum pulse rate of 1. The number of selected features is 256. The highest training accuracy and F1-score have been achieved by DT, AdaB, GradB, and XGB is 100% though the result of SGD is noticeably poor, with an accuracy of 25% and an F1-score of 10% only. The boosting algorithms have also obtained higher values of F1-score 86%, 88%, and 90% for AdaB, GradB, and XGB respectively.

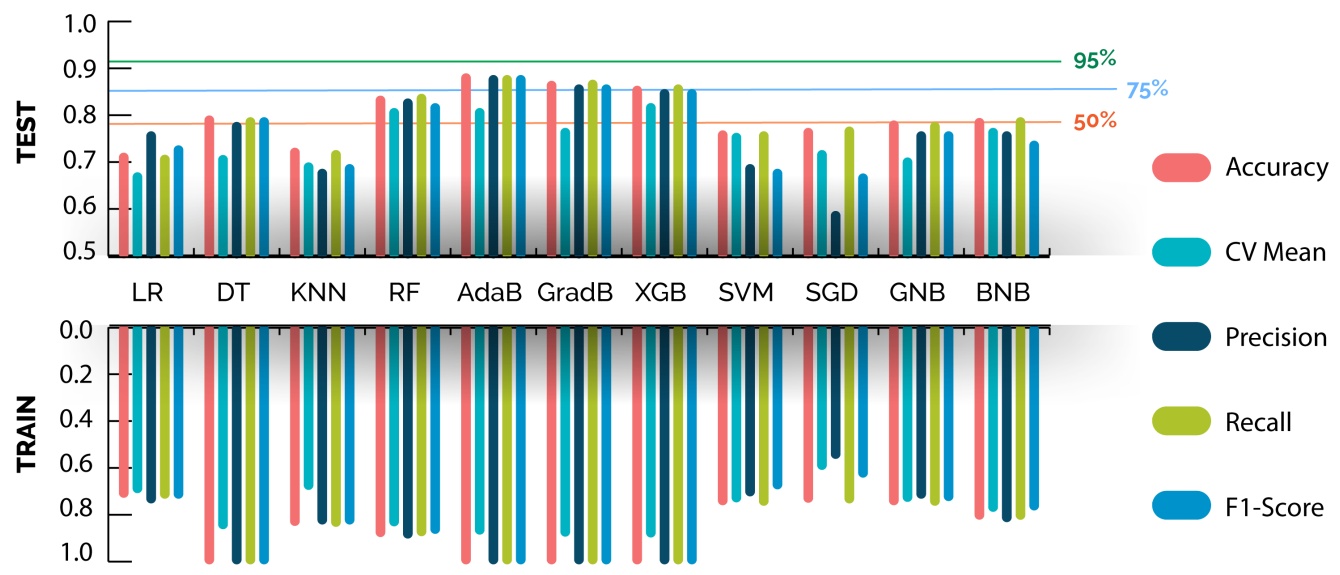


**FIGURE 2.** Training and testing performance of the classifiers without feature selection described in Section IV B. The upper and lower portions reflect the testing and training results, respectively. The horizontal axis shows the eleven ML classifiers, and the vertical axis shows the acquired scores of the classifiers for different performance metrics. No classifier has achieved a testing accuracy of more than 90%. Legends: LR — Linear Regression; SVM — Support Vector Machine; RF — Random Forest; K-NN — K-Nearest Neighbors; DT — Decision Tree; AdaB — Adaptive Boosting Algorithm; GradB — Gradient Boosting Algorithm; XGB — Extreme Boosting Algorithm; SGD — Stochastic Gradient Descent Algorithm; GNB — Gaussian Naive Bayes; BNB — Bernoulli Naive Bayes.

**TABLE 1.** A summary of the thirteen NI Algorithms' parameter settings for feature selection

| NI Algorithm | Parameter settings |
|---|---|
| BA | maximum frequency=5, maximum loudness=3, maximum pulse rate=1 |
| CS | optimum discovery rate=0.25, Levy component=1.5, threshold value=0.65 |
| DE | crossover rate=0.9, default factor=0.5 |
| FA | light amplitude=2, absorption coefficient=1, control alpha=0.97 |
| FPA | levy component=1.5, switch probability=0.8 |
| GA | crossover rate=0.8, mutation rate=0.05 |
| GWO | threshold=0.5 |
| HHO | threshold=0.5,levy component=1.5 |
| PSO | inertia weight=0.9, acceleration factor=2,3 |
| SCA | relative importance of error (alpha)= 2 |
| SSA | threshold=0.5 |
| WOA | threshold=0.5, logarithmic spiral=1.3 |

Legends: NI algorithm — Nature-inspired algorithm.

The average testing accuracy is 75.75% among all the ML classifiers, with an average testing F1-score of 72.63% shown in figure 4. The maximum accuracy for the AdaB, GradB and XGB classifiers are 89.95%, 89.42% and 90.48%, respectively. XGB has also achieved the highest F1-score of 90%.

### 2) CS

For CS feature selection, the optimum discovery rate is found at 0.25, the Levy component is 1.5, and the threshold value is 0.65. The number of selected features is 328. Like the BA feature selection, DT, AdaB, GradB, and XGB have achieved the highest training accuracy and F1-score of 100% and SGD showed a very poor result with an accuracy of 34% and F1-score 30% only.

The method has an average testing accuracy of 74.7% and an F1-score of 72.63% with all the ML classifiers.

The AdaB, GradB and XGB classifiers have maximum accuracy of 83%, 84% and 86%, respectively. XGB also has the highest F1 score of 86%. The maximum training and testing cross-validation mean are achieved by XGB with a value of 89% and 82% respectively can be found in figure 5.

### 3) DE

In the case of DE feature selection, the results are better than most other algorithms. The crossover rate of 0.9 and the default factor of 0.5 has been selected for feature selection. The number of selected features is 289. Like the BA and CS, this algorithm has attained maximum training accuracy and F1-score of 100% for DT, AdaB, GradB, and XGB. But unlike BA and CS, this approach has performed not that poor accompanied by SGD with an accuracy of 75% and F1-score 64%. The highest cross-validation mean is brought out with XGB by 87% and AdaB and GradB have scored the same, 86%.

With a mean testing accuracy of 79.81%, this approach has the highest accuracy with GradB and XGB classifiers, 91% and the average F1-score is 78%. XGB achieves the highest cross-validation mean of 84% which is depicted in figure 6.

### 4) FA

FA is a kind of mixed bag in terms of performance. For the parameters of this algorithm, the light amplitude of 2, the absorption coefficient of 1 and the controlling alpha of 0.97

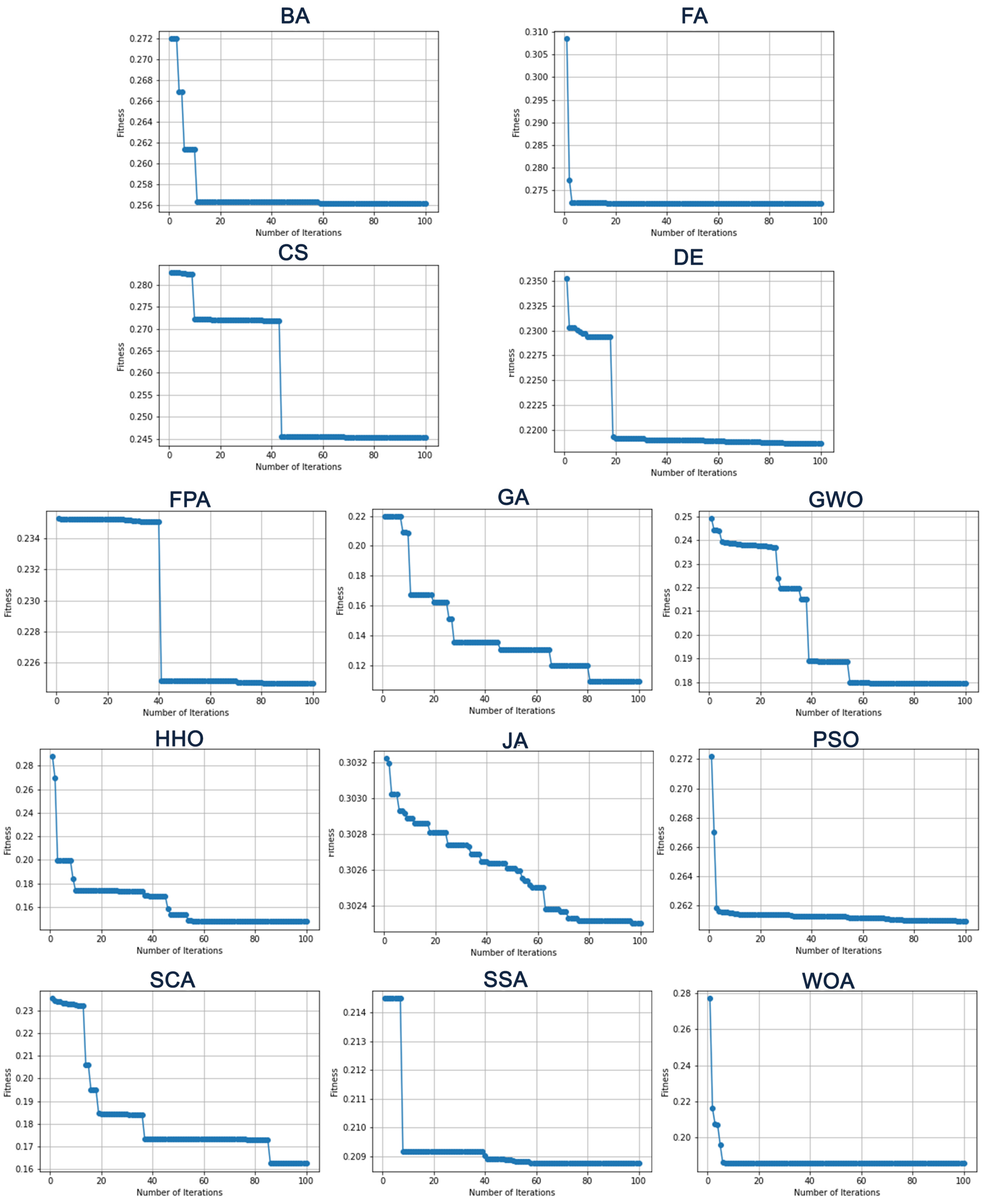


FIGURE 3. Fitness for 100 Epoch of each NI feature selection algorithm. The horizontal axis shows the number of iterations, and the vertical axis shows the fitness of the respective NI algorithm.

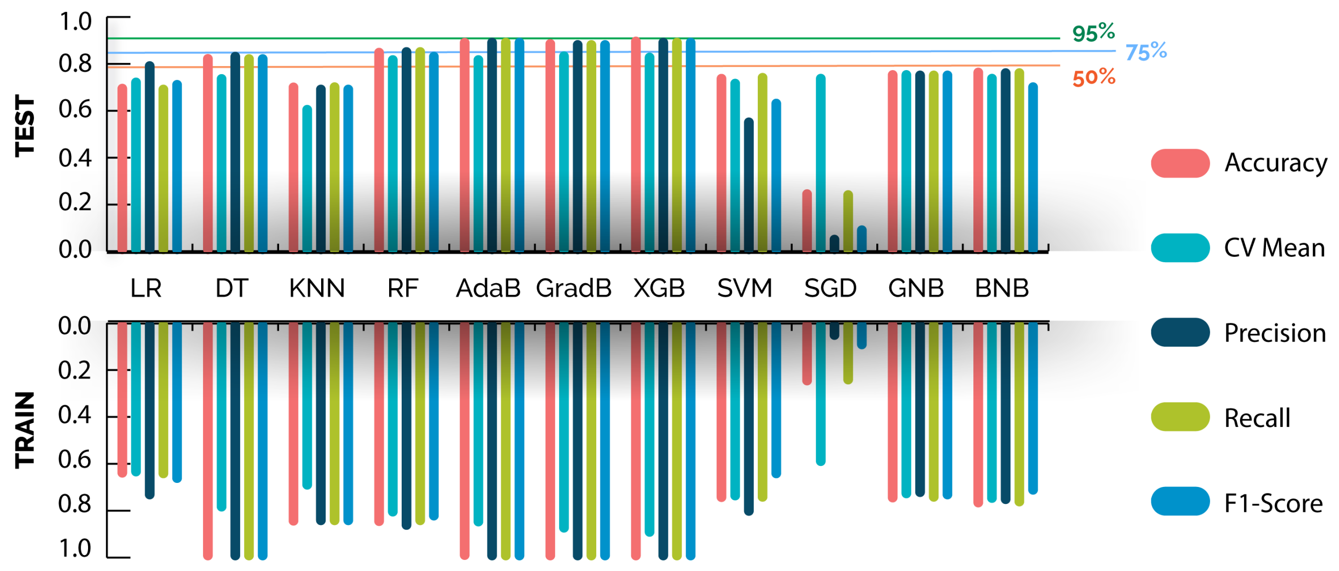


FIGURE 4. Training and testing performance of the classifiers after feature selection with BA described in Section IV C 1) BA. The highest training accuracy and F1-score are 100% which has been achieved by DT, AdaB, GradB, and XGB. The maximum testing accuracy is achieved by XGB.

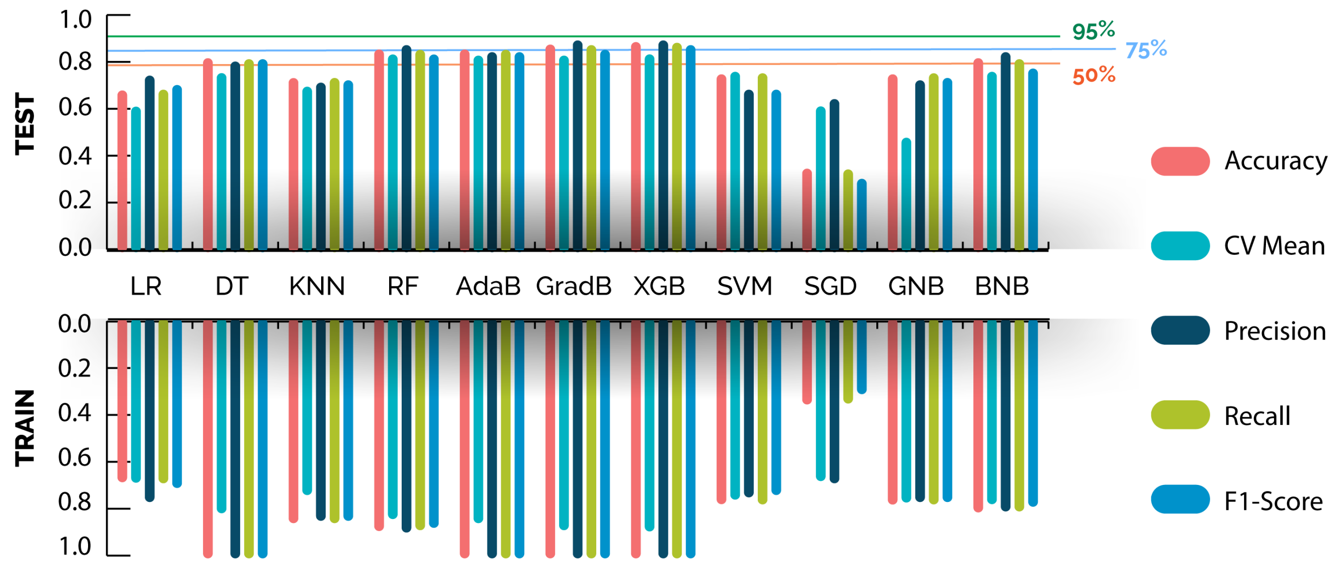


FIGURE 5. Training and testing performance of the classifiers after feature selection with CS described in Section IV C 2) CS. XGB has the maximum testing accuracy and SGD has the minimum.

have been selected. Alike the previous three algorithms, FA has carried out the highest training accuracy and F1-score with DT, AdaB, GradB, and XGB of 100%. Additionally, the FA-SGD method has yielded the same results as the DE-SGD.

The average testing accuracy for all ML classifiers is 77.36%, and the average F1-Score is 74.81%. Here, BNB and GNB classifiers achieved 74% and 75% cross-validation scores that can be observed from figure 7. The number of features selected was 363.

#### 5) FPA

In the instance of FPA feature selection, the testing accuracy outperforms other techniques that are delineated in figure 8. For feature selection, a levy component of 1.5 and a switch probability of 0.8 were chosen. 348 features have been selected. The technique has the highest testing accuracy for GradB and XGB classifiers, 91%, with a mean testing accuracy of 79.81%. The typical F1-score is 78%. XGB has the most significant cross-validation mean of 84%.

DT, AdaB, GradB, and XGB have also obtained 100% training accuracy and F1-score for this approach. GradB has scored the highest cross-validation means of 87% and AdaB,

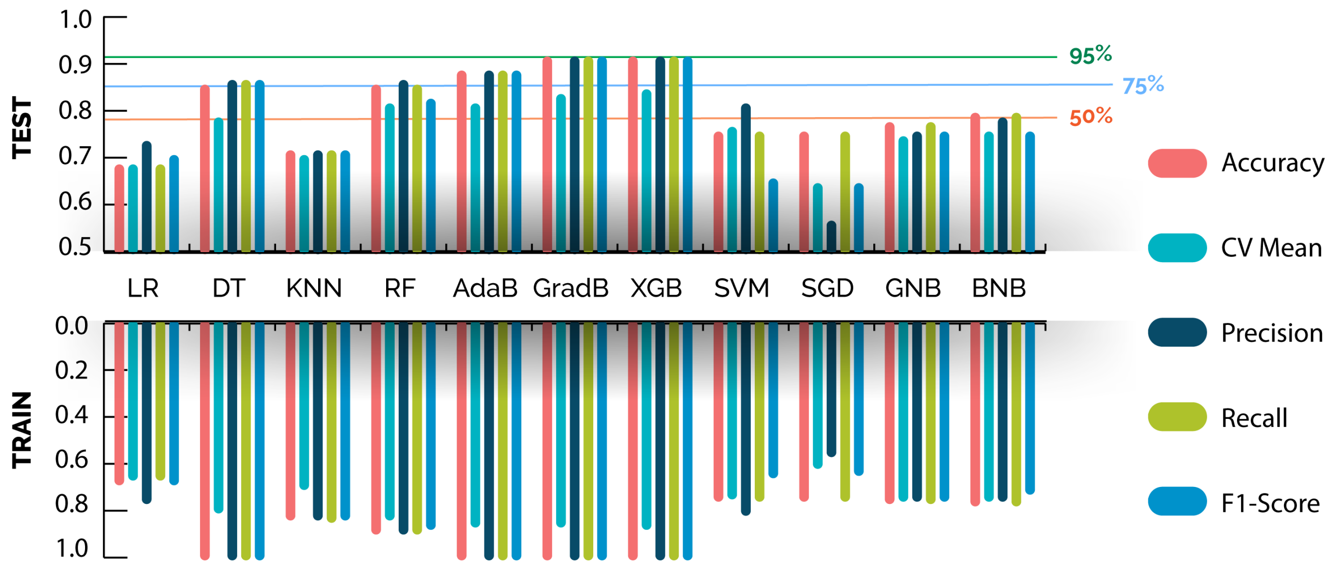


FIGURE 6. Training and testing performance of the classifiers after feature selection with DE described in Section IV C 3) DE. From an accuracy perspective, SGD performed better than BA and CS with this algorithm.

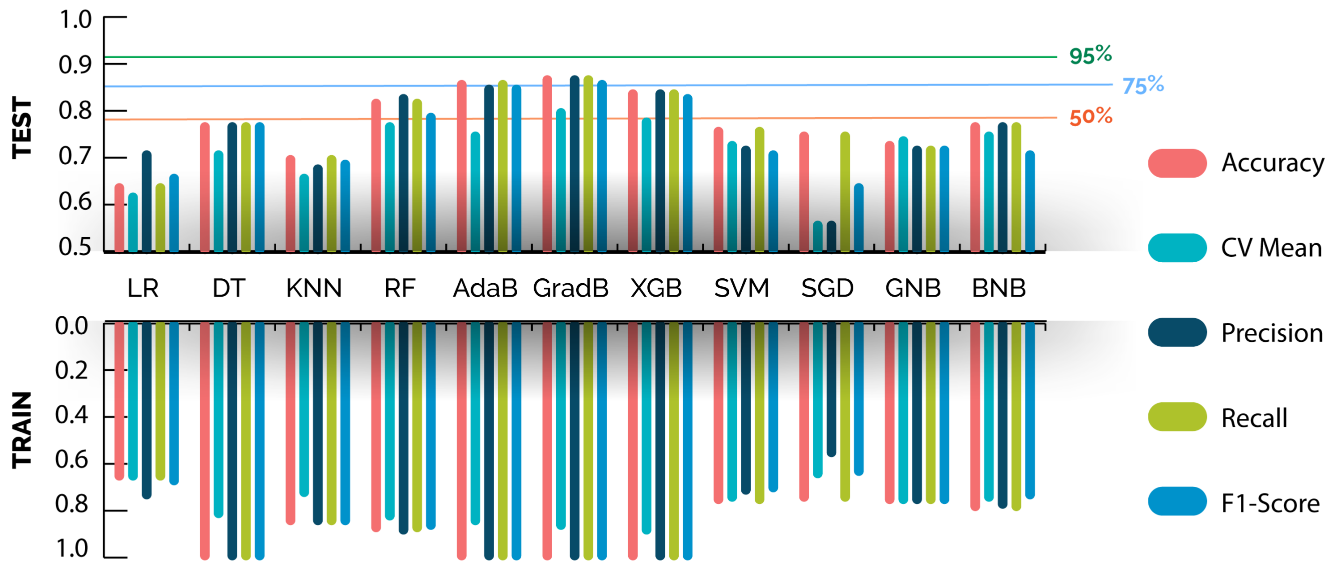


FIGURE 7. Training and testing performance of the classifiers after feature selection with FA described in Section IV C 4) FA. No classifier could obtain accuracy and F1-score to 90% with this algorithm.

XGB has scored the same, 86%. The number of selected features is 348.

### 6) GA

For GA, a crossover rate of 0.8 and a mutation rate of 0.05 are selected as the parameter values. 335 feature has been selected through the GA. The maximum value of accuracy, F1-score and cross-validation mean for GA with ML classifiers is akin to DE. Among all other NI algorithms, GA has performed better with SGD with an accuracy of 81% and F1-score of 80%.

GA has achieved average testing accuracy of 82.92% and an average F1-Score of 81.18% (figure 9). XGB achieved the highest F1-Score of 92% compared to all other NI algorithms and ML classifiers. Though in terms of cross-validation accuracy, AdaB performs better in the experiment. The AdaB classifier has achieved the maximum testing cross-validation mean score of 86.28%.

### 7) GWO

The ideal threshold for GWO feature selection is 0.5 for the experiments. The number of features chosen is 55. The average training accuracy of all ML classifiers with this algorithm is 86% which is the second-highest score. But like

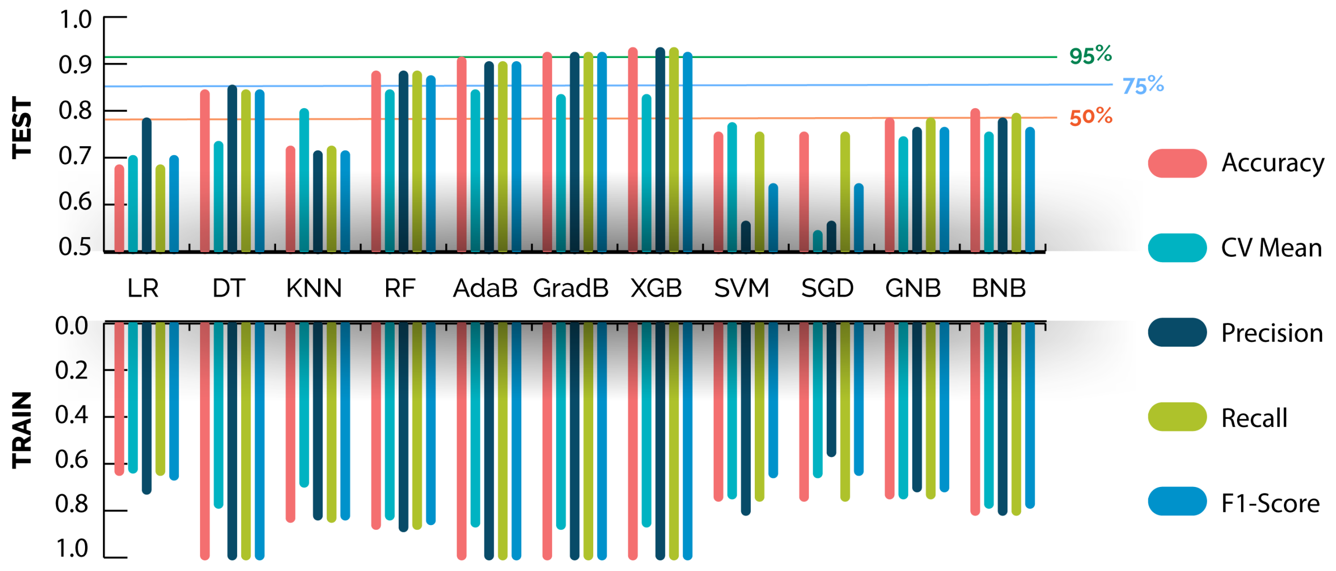


FIGURE 8. Training and testing performance of the classifiers after feature selection with FPA described in Section IV C 5) FPA. The testing accuracy value of AdaB reached the 95th percentile line and the testing accuracy values of GradB and XGB were above the 95th percentile line.

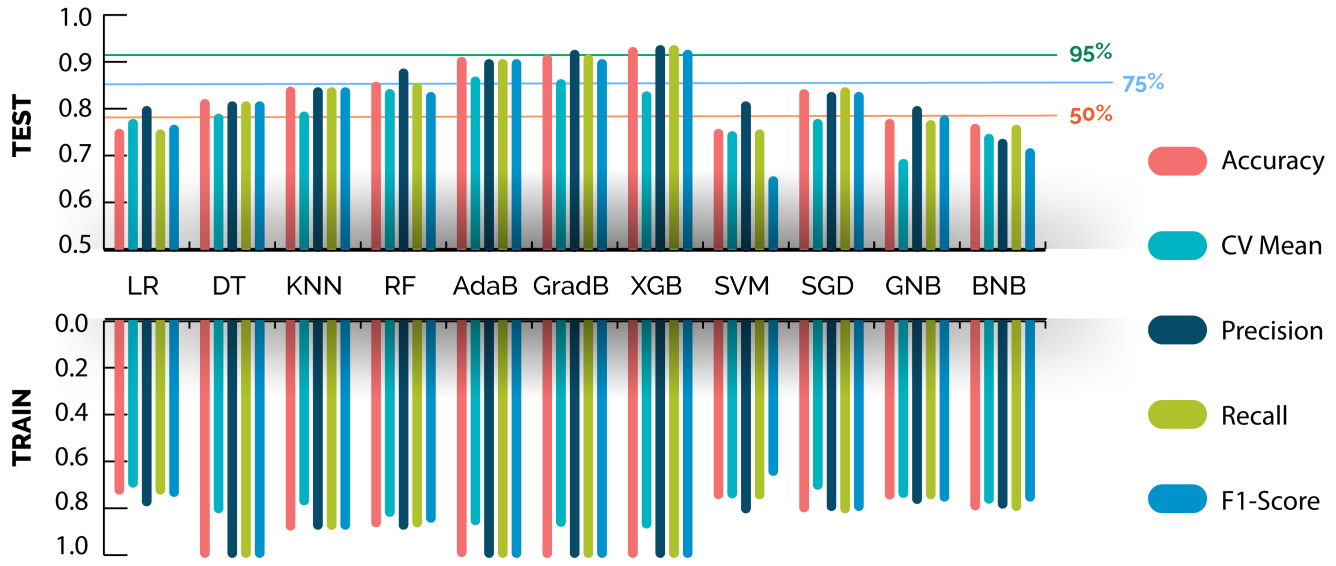


FIGURE 9. Training and testing performance of the classifiers after feature selection with GA described in Section IV C 6) GA. Maximum testing accuracy has been gained by XGB and maximum cross-validation means by AdaB.

most other algorithms AdaB could not achieve 100% training accuracy and F1-score after selecting features by GWO. The accuracy values of LR, SVM, SGD, and GNB are very close to this technique.

The approach has the average testing accuracy with all ML classifiers, at 79.63% and F1-Score at 77.90%. From figure 10, it can be noticed that GradB classifiers have achieved the maximum accuracy of 89%. GradB also has the highest testing F1-score of 88%. The maximum cross-validation means has been achieved by AdaB with a value of 86%.

### 8) HHO

The HHO algorithm is one of the few algorithms that was never used in the case of PD feature selection. For tuning, the levy component of 1.5 has been fixed as the optimum value for all the classifiers. HHO selected 78 features from the PD dataset. After applying the eleven classifiers, HHO came up with an average training accuracy of 85.1%. The maximum value of accuracy, F1-score and cross-validation mean for HHO with ML classifiers is matched with GA and GWO.

HHO achieved an average testing accuracy of 79.79% and an F1-Score of 78.27%. Amid all ML classifiers, GNB

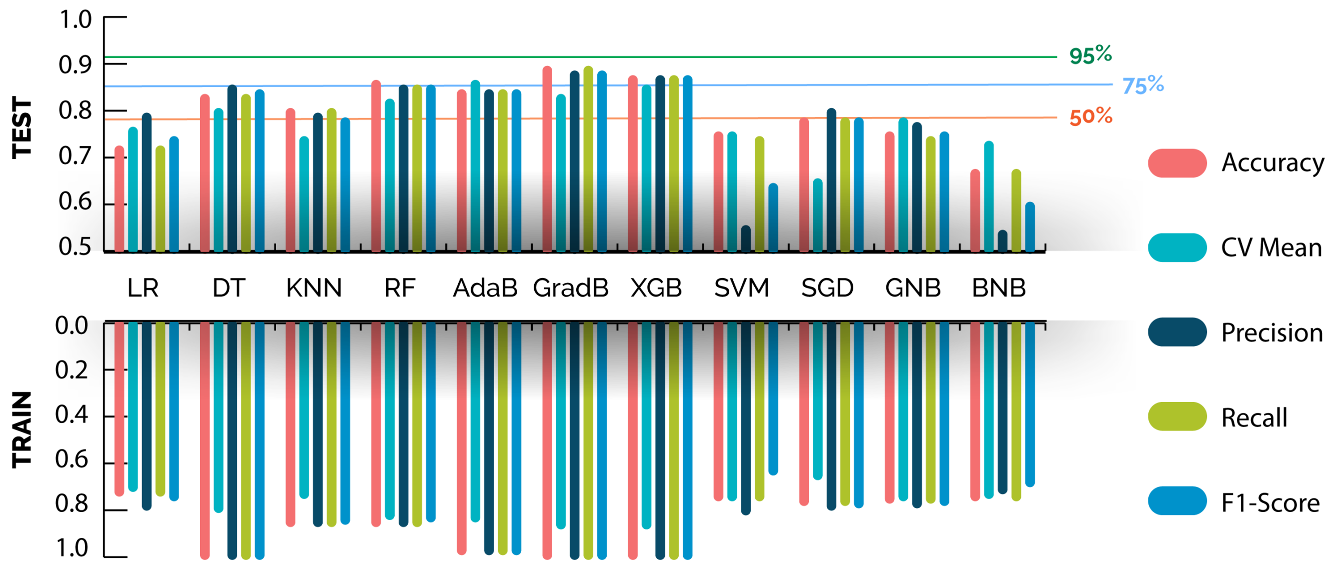


FIGURE 10. Training and testing performance of the classifiers after feature selection with GWO described in Section IV C 7) GWO. Unlike the others, the testing accuracy of AdaB is below the 75th percentile. The performance of BNB is the lowest with GWO.

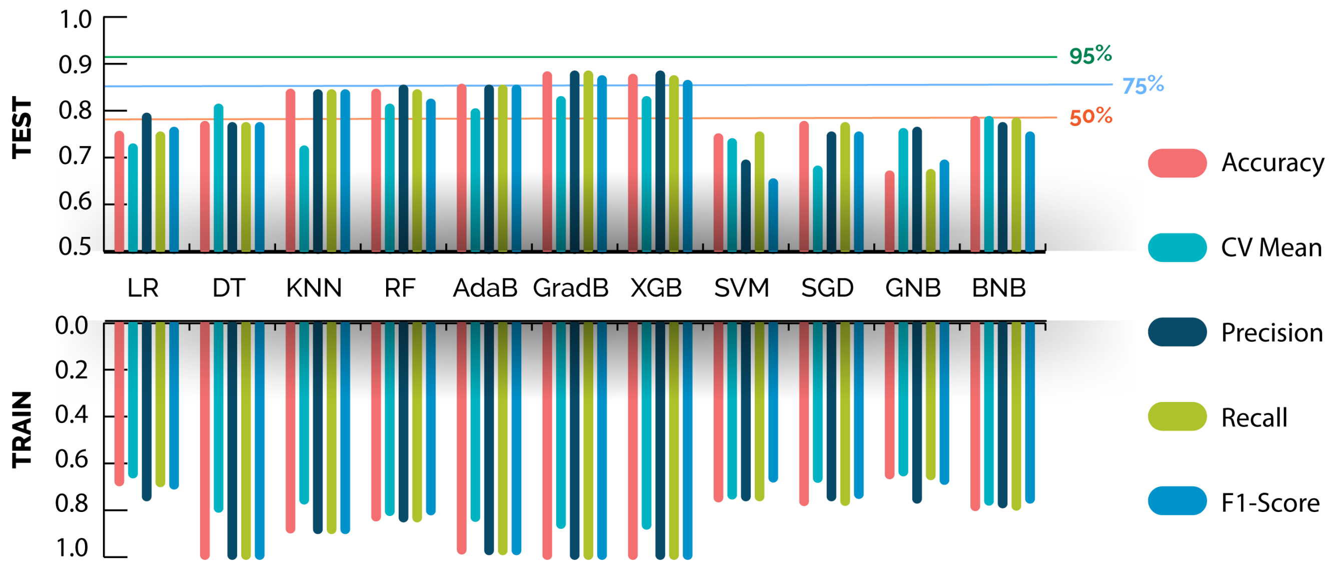


FIGURE 11. Training and testing performance of the classifiers after feature selection with GA described in Section IV C 8) HHO. The average performance of the classifiers is good for this algorithm.

performed worst having an accuracy of 65%. Maximum testing accuracy is achieved by GradB at 87.83%, maximum F1-Score of 87%, and the maximum cross-validation mean of 82.54% is achieved by both GradB and XGB are depicted in figure 11

### 9) JA

For the experiments, the best threshold for JA feature selection was 0.5. A total of 281 characteristics were selected. JA arouse with an average training accuracy of 85.64%, near GWO. The maximum value of training accuracy, F1-score for JA with ML classifiers is 100% similar to GWO and HHO. Other than this, JA-AdaB, JA-KNN, and JA-RF methods have produced an accuracy of 99%, 87%, 85% and F1-score of 99%, 86%, and 84%, respectively.

Figure 12 outlined that with an average F1-Score of 78.45%, the technique has an average testing accuracy of 79.54% with all ML classifiers. GradB classifier has reached a maximum accuracy of 91%. GradB also has the highest F1-Score of 90% in testing. RF has achieved the highest cross-validation mean with a value of 84%.

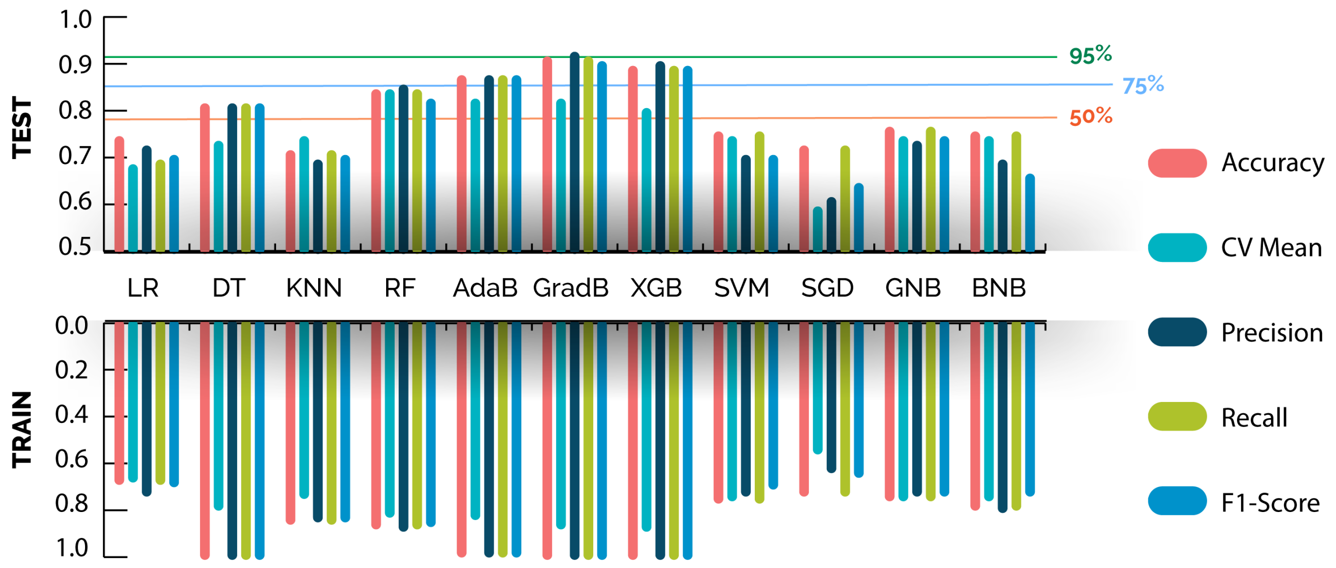


**FIGURE 12.** Training and testing performance of the classifiers after feature selection with JA described in Section IV C 9) JA. JA boosted the performance of GradB the most which enabled the accuracy of GraB to reach the 95th percentile.

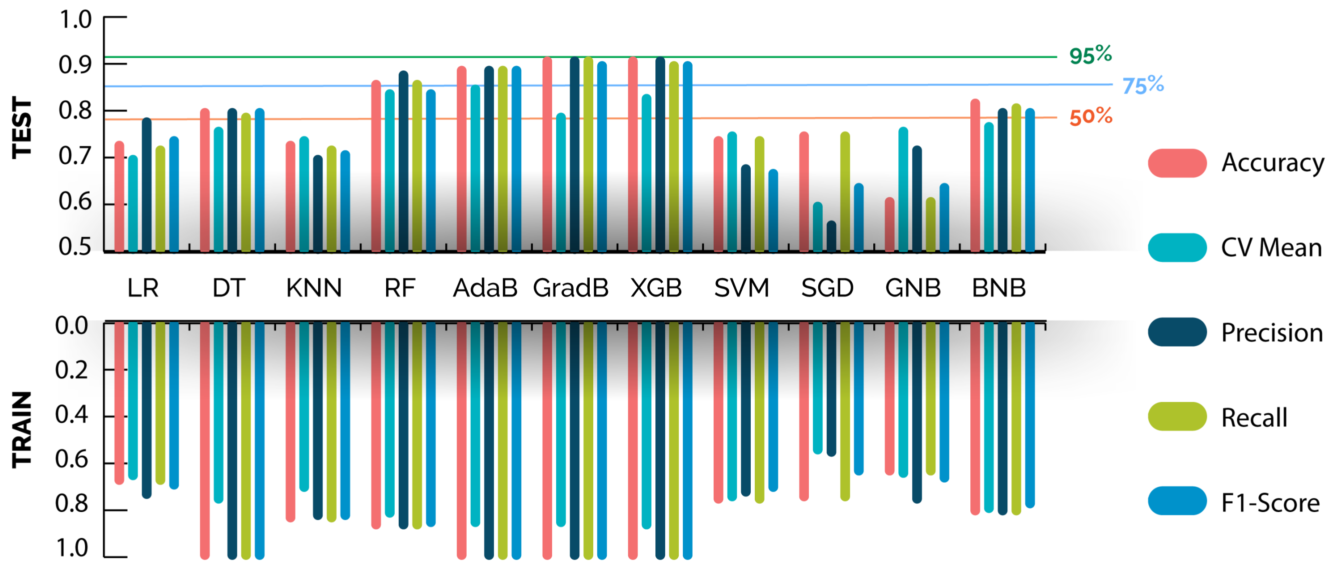


**FIGURE 13.** Training and testing performance of the classifiers after feature selection with PSO described in Section IV C 10) PSO. GradB and XGB reached the 95th percentile line.

### 10) PSO

PSO is one of the well-established NI algorithms for feature selection. For this algorithm, an inertia weight of 0.9 and an acceleration factor of 2 and 3 were selected. This algorithm carried out an average training accuracy of 85% with all classifiers. PSO attained the highest training accuracy and F1-score with DT, AdaB, GradB, and XGB valued at 100% and the highest cross-validation mean of 87% with XGB.

The mean testing accuracy is 79.54%, and the F1-Score is 77.54%. GradB and XGB have achieved the maximum accuracy at 91%. Although, AdaB has the maximum cross-validation mean of 82% (figure 13). The number of selected features is 321.

### 11) SCA

The number of selected features for SCA was very low through multiple experiments. Only 14 features were selected using SCA. Surprisingly the average testing accuracy of SCA is higher than its training accuracy valued at 84.27%. Unlike most other NI algorithms, SCA does not have a training accuracy of 100% with AdaB and GradB. SCA-DT and SCA-XGB methods achieved the highest accuracy of 100%. The maximum cross-validation means has been achieved by GradB and XGB, 82%. But SVM had the lowest F1-score

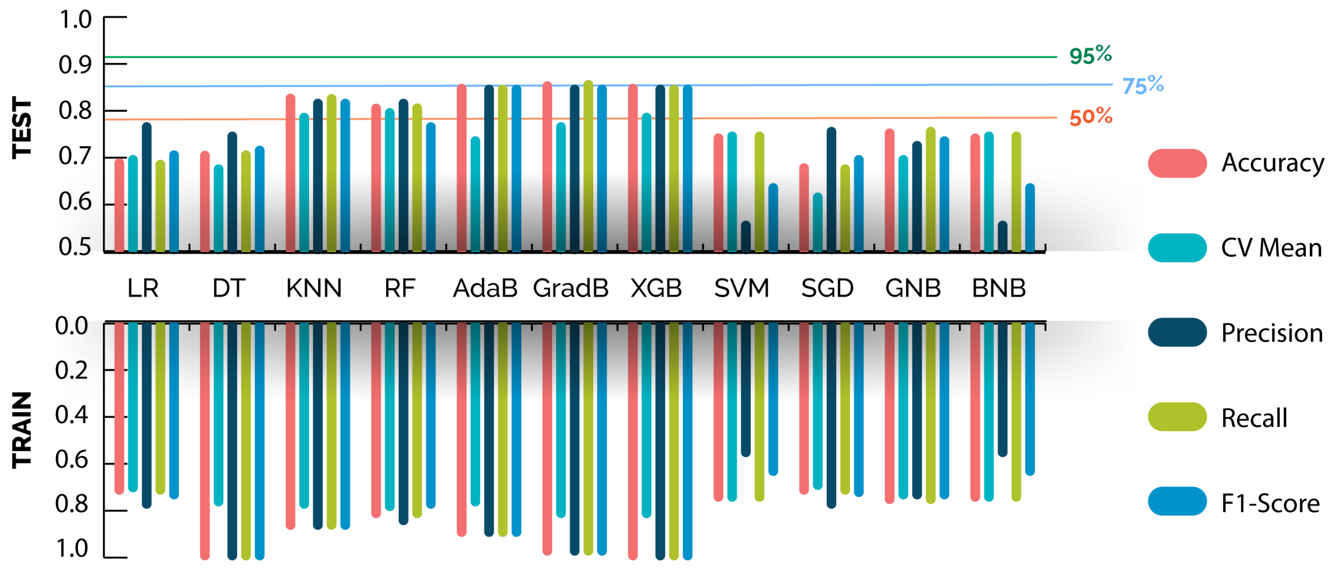


**FIGURE 14.** Training and testing performance of the classifiers after feature selection with SCA described in Section IV C 11) SCA. The average performance of the classifiers is worst in accordance with this algorithm. No classifier has accuracy values in and above the 75th percentile.

value of 64% with this approach.

SCA achieved maximum testing accuracy of 85.71% and 85.19% for the GrarB and AdaB classifiers, respectively. The method has achieved average testing accuracy of 77.58% and an F1-Score of 75% after applying the eleven classifiers (figure 14). The maximum cross-validation mean is attained by KNN at 79% (figure 14).

#### 12) SSA

The optimal threshold for SSA feature selection in the studies was 0.5. There were a total of 337 features chosen. The maximum training accuracy of SSA is 100% which is indistinguishable from BA, CS, DE, FA, FPA and PSO regarding ML classifiers. Among all the NI algorithms, the classification accuracy and F1-score of GNB are the least with SSA valued at 52% and 54% respectively.

The approach has an average testing accuracy of 78.36% and an average F1-Score of 76.181%. GradB classifier has achieved a maximum accuracy of 92%. GradB and XGB also have the highest F1-score in testing, with 91%. With a value of 86%, RF has the highest cross-validation mean. These are illustrated in figure 15.

#### 13) WOA

The best WOA feature selection threshold was 0.5 in the experiments and the constant beta equals 1.3. There were a total of 181 traits chosen. The average training accuracy of WOA is the highest of all NI algorithms valued at 86.62%. KNN showed the best accuracy and F1-score of 90% after selecting features with this algorithm among all NI algorithms.

The approach has a mean testing accuracy of 78.88% among all ML classifiers, with an average F1-Score of 75.91%. GradB classifier has an 87.83% accuracy. GradB has the highest F1-Score in testing, with a score of 87% (figure 16). With a rating of 97.8%, RF has the best cross-validation mean.

### D. PERFORMANCE COMPARISON AND DISCUSSION

F1-score and accuracy of different ML classifiers were compared to assess the effectiveness of NI feature selection algorithms. Because each ML algorithm was configured the same throughout the experiments, only a few features were responsible for different performance measures. Figure 18 represent accuracy and figure 19 represent F1-Score of pairwise NI-ML approaches. The x-axis is for the feature selection algorithms, and the y-axis is for the ML classifiers. N/A in the x-axis is the classification without any feature selection. Most dimensionality reduction algorithms increase the accuracy and F1-score compared to the plain ML approaches. Each algorithm reduces the number of features, reducing the train-test time. Comparing from the NI algorithms' perspective (column-wise in fig 18 and fig 19), FPA, GA and SSA algorithms' accuracy and f1-score are on par with or better than other NI algorithms as well as without feature selection. Again FPA and SSA algorithms are comparatively new inventions, and their contribution towards feature selection in recent literature is yet to be found. From the ML classification perspective (row-wise), classifiers play the most important roles in the difference in the result.

All three boosting algorithms (AdaB, GradB and XGB) with RF perform better than all other classifiers. As shown in figure 18 and figure 19, XGB outperformed most other classifiers in the testing and training phases. As a result, in Figure 17, testing performance for various NI algorithms' results was compared. Another comparison can be found in the 17, which depicts the RF classifier's performance. As delineated in figure 17, FPA, SSA, GA, DE, and GWO were

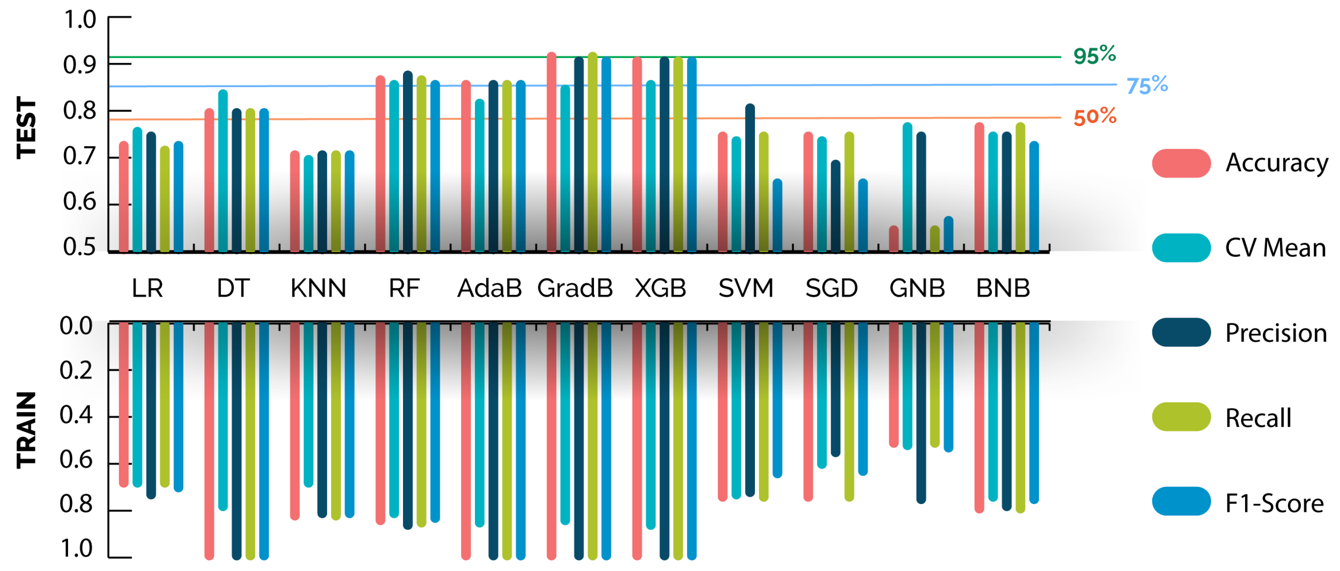


**FIGURE 15.** Training and testing performance of the classifiers after feature selection with SSA described in Section IV C 12) SSA. The accuracy of GNB is the least with PSO among all NI algorithms but GraB crossed the 95th percentile line, i.e., has an accuracy above 91%.

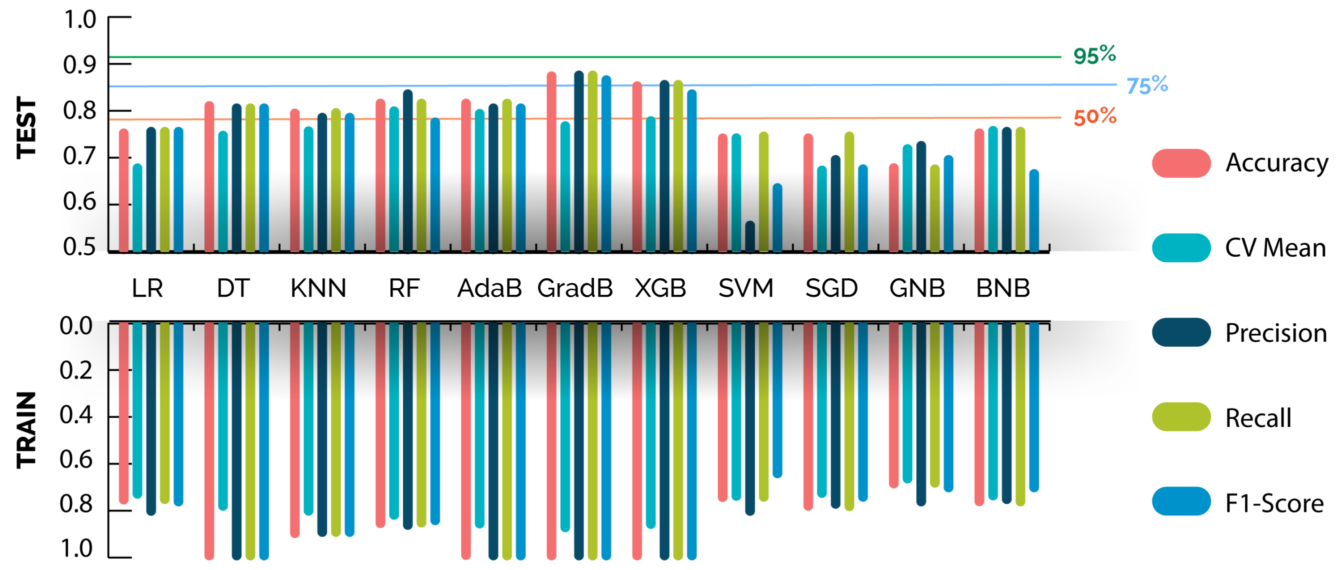


**FIGURE 16.** Training and testing performance of the classifiers after feature selection with WOA described in Section IV C 13) WOA. The performance of WOA is average with the classifiers.

most prominent in terms of feature selection.

However, the performance of different algorithm pairs needs to be addressed by some assessment criteria. Here, we have ranked the performance of different methods based on percentile. The methods are ranked as excellent, good and moderate based on the percentile score shown in figure 20. The 50th, 75th and 95th percentile have been appraised to express the value at a particular rank. The median of the data is the 50th percentile. After calculating the percentile function, the value of the 50th percentile was found at 0.785, i.e. 50% of the accuracy values of different methods are above 78.5%. Similarly, the accuracy values above the 75th percentile were calculated at 0.867. Hence, NI-ML pairs' testing accuracy values which are greater than 0.867 are in the domain of the 75th percentile. We have assessed the 95th percentile for ranking our highest data points which are above 0.91 and 12 pairs of algorithms are found in this range. FPA-AdaB, DE-GradB, GA-GradB, JA-GradB, PSO-GradB, DE-XGB, PSO-XGB, SSA-XGB, FPA-GradB, SSA-GradB, GA-XGB, FPA-XGB are the 12 pairs that are evaluated as best and ranked as excellent. 25 NI-ML methods were found between 75th and 95th percentile, and 34 pairs between the 50th to 75th were rated as excellent and moderate performance.

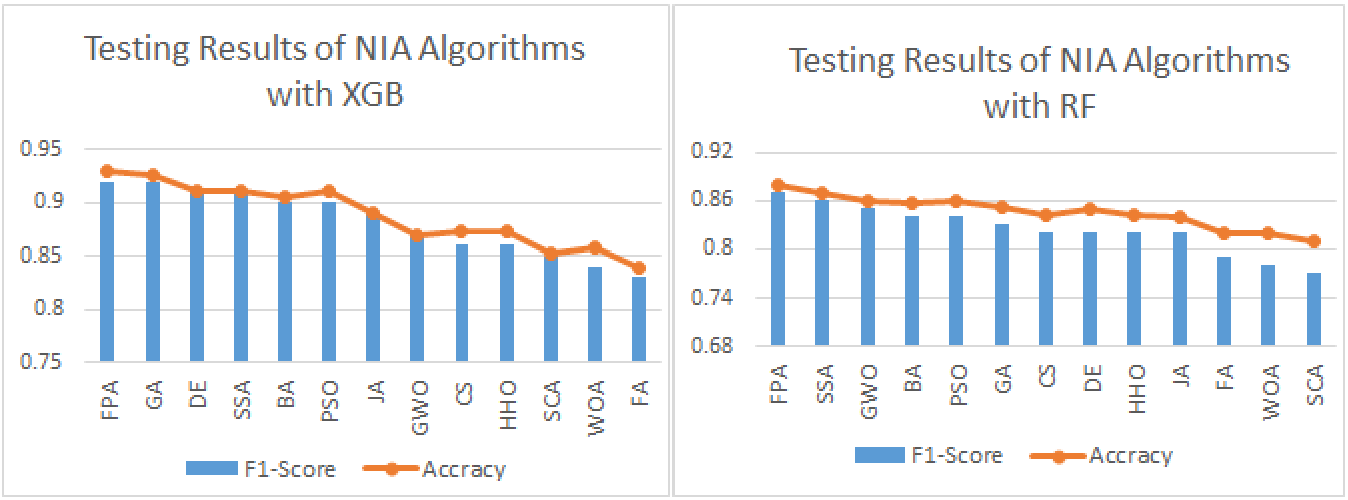
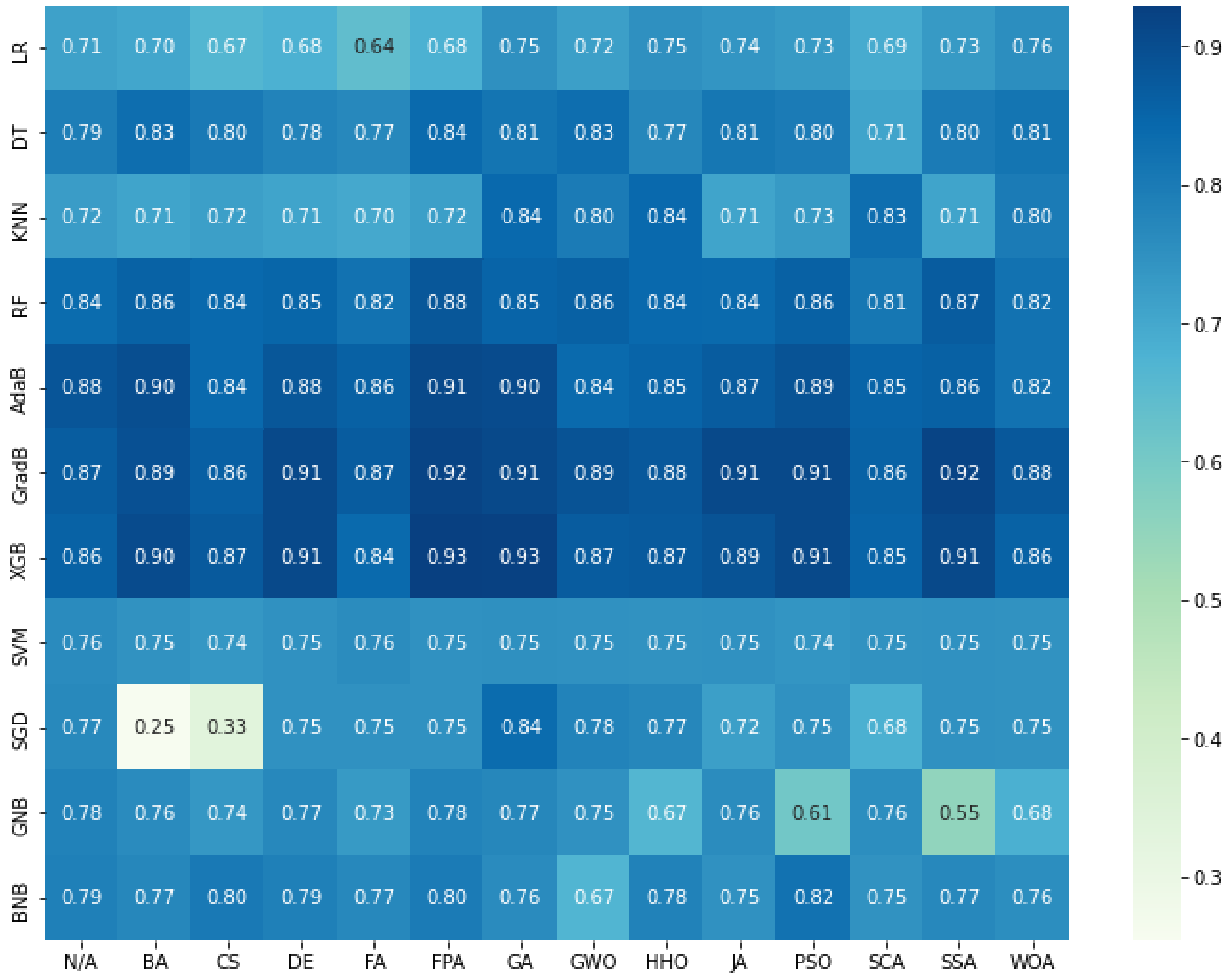


**FIGURE 17.** Performance comparison of the 13 NI algorithms for feature selection each pairing with XGB and RF classifiers.



| | N/A | BA | CS | DE | FA | FPA | GA | GWO | HHO | JA | PSO | SCA | SSA | WOA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LR | 0.71 | 0.70 | 0.67 | 0.68 | 0.64 | 0.68 | 0.75 | 0.72 | 0.75 | 0.74 | 0.73 | 0.69 | 0.73 | 0.76 |
| DT | 0.79 | 0.83 | 0.80 | 0.78 | 0.77 | 0.84 | 0.81 | 0.83 | 0.77 | 0.81 | 0.80 | 0.71 | 0.80 | 0.81 |
| KNN | 0.72 | 0.71 | 0.72 | 0.71 | 0.70 | 0.72 | 0.84 | 0.80 | 0.84 | 0.71 | 0.73 | 0.83 | 0.71 | 0.80 |
| RF | 0.84 | 0.86 | 0.84 | 0.85 | 0.82 | 0.88 | 0.85 | 0.86 | 0.84 | 0.84 | 0.86 | 0.81 | 0.87 | 0.82 |
| AdaB | 0.88 | 0.90 | 0.84 | 0.88 | 0.86 | 0.91 | 0.90 | 0.84 | 0.85 | 0.87 | 0.89 | 0.85 | 0.86 | 0.82 |
| GradB | 0.87 | 0.89 | 0.86 | 0.91 | 0.87 | 0.92 | 0.91 | 0.89 | 0.88 | 0.91 | 0.91 | 0.86 | 0.92 | 0.88 |
| XGB | 0.86 | 0.90 | 0.87 | 0.91 | 0.84 | 0.93 | 0.93 | 0.87 | 0.87 | 0.89 | 0.91 | 0.85 | 0.91 | 0.86 |
| SVM | 0.76 | 0.75 | 0.74 | 0.75 | 0.76 | 0.75 | 0.75 | 0.75 | 0.75 | 0.75 | 0.74 | 0.75 | 0.75 | 0.75 |
| SGD | 0.77 | 0.25 | 0.33 | 0.75 | 0.75 | 0.75 | 0.84 | 0.78 | 0.77 | 0.72 | 0.75 | 0.68 | 0.75 | 0.75 |
| GNB | 0.78 | 0.76 | 0.74 | 0.77 | 0.73 | 0.78 | 0.77 | 0.75 | 0.67 | 0.76 | 0.61 | 0.76 | 0.55 | 0.68 |
| BNB | 0.79 | 0.77 | 0.80 | 0.79 | 0.77 | 0.80 | 0.76 | 0.67 | 0.78 | 0.75 | 0.82 | 0.75 | 0.77 | 0.76 |

**FIGURE 18.** Accuracy comparison for thirteen NI algorithms and eleven ML classifiers. The horizontal axis presents the feature selection algorithms without feature selection (N/A), and the vertical axis presents the classifiers used in this study. The colour bar represents the accuracy values with respect to colour intensities.

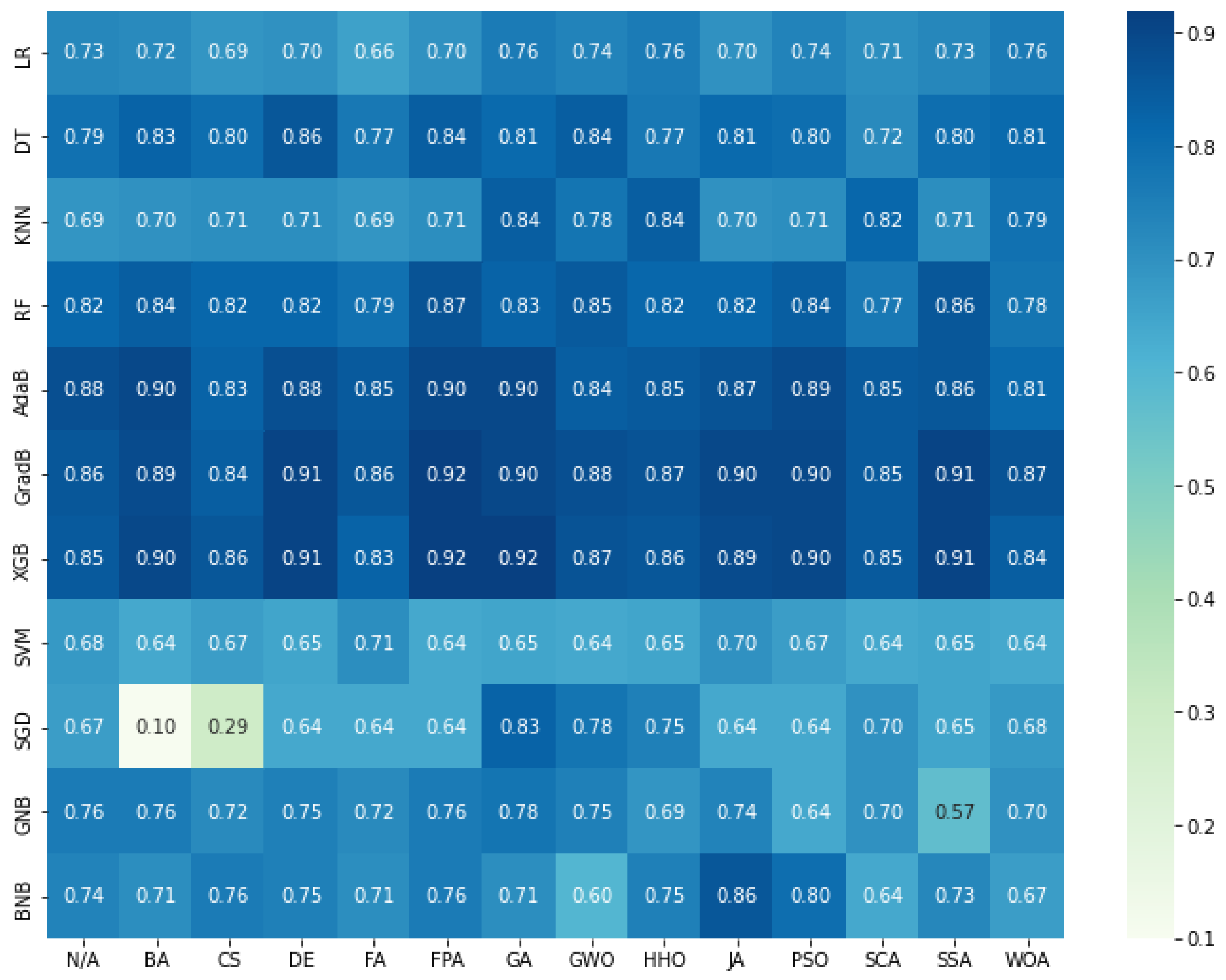


**FIGURE 19.** F1-Score Comparison for thirteen NI algorithms and eleven ML classifiers.

In this study, a rather comprehensive comparison is done with various nature-inspired feature reduction algorithms with ML classifiers. Originally, the PD dataset contains a huge number of features (754) which has been collected from speech. Since the applied speech feature extraction is independent of the classification problem, it is important to select the necessary features and remove the unnecessary ones which may cause false classification. From this perspective, the learning-based nature-inspired algorithm has shown tremendous success in recent years for feature reduction/selection [41], [43]. Moreover, fewer features also account for less training time for the ML classification. Although there are many feature selection algorithms presented in the literature for PD classification, a thorough comparison to select the best ones, in another perspective best feature selection and classification duo for PD is indeed a research gap. This article addresses this issue by implementing and experimenting with 13 feature selection algorithms ( BA, CS, DE, FA, FPA, GA, GWO, HHO, JA, PSO, SCA, SSA, WOA) with 11 ML classification algorithms (LR, DT, KNN, RF, AdaB, GradB, XGB, SVM, SGD, GNB, BNB). These algorithms are mostly chosen based on the recent literature with some exceptions for some recent ones such as JA, FPA, and SSA. From the experiments, the ensemble and boosting styles algorithms perform significantly better than other ones because of the heterogeneity of datasets features. These algorithms allocate a subsection of features to individual smaller ML models, make predictions and correct each of them from previous errors. Such techniques benefit from the large feature 18, the effect of feature selection is also prominent. BA, DE, FPA, GA, PSO, and SSA algorithms work mostly better than other feature selection algorithms. Although there is not much common among them compared to the ML classifiers, particular evolution heuristics might be the case for ideal feature selection in PD classifications.

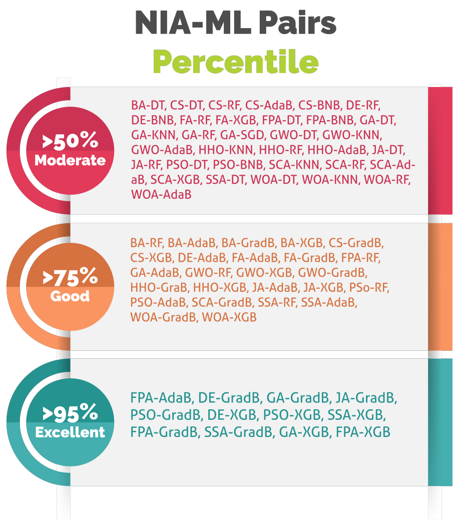

NIA-ML Pairs
Percentile
>50% Moderate
BA-DT, CS-DT, CS-RF, CS-AdaB, CS-BNB, DE-RF, DE-BNB, FA-RF, FA-XGB, FPA-DT, FPA-BNB, GA-DT, GA-KNN, GA-RF, GA-SGD, GWO-DT, GWO-KNN, GWO-AdaB, HHO-KNN, HHO-RF, HHO-AdaB, JA-DT, JA-RF, PSO-DT, PSO-BNB, SCA-KNN, SCA-RF, SCA-AdaB, SCA-XGB, SSA-DT, WOA-DT, WOA-KNN, WOA-RF, WOA-AdaB
>75% Good
BA-RF, BA-AdaB, BA-GradB, BA-XGB, CS-GradB, CS-XGB, DE-AdaB, FA-AdaB, FA-GradB, FPA-RF, GA-AdaB, GWO-RF, GWO-XGB, GWO-GradB, HHO-GraB, HHO-XGB, JA-AdaB, JA-XGB, PSo-RF, PSO-AdaB, SCA-GradB, SSA-RF, SSA-AdaB, WOA-GradB, WOA-XGB
>95% Excellent
FPA-AdaB, DE-GradB, GA-GradB, JA-GradB, PSO-GradB, DE-XGB, PSO-XGB, SSA-XGB, FPA-GradB, SSA-GradB, GA-XGB, FPA-XGB

**FIGURE 20.** Performance ranking among the 143 pairs of NI-ML method based on percentile. Scores above the 50th, 75th and 95th percentile are considered to have moderate, good and excellent performance respectively.

## V. CONCLUSION

Datasets play an important role in solving classification problems, where reducing the high dimensionality of data is a considerable challenge. This problem is addressed and solved in a systematic comparison manner in the current study. The dimensionality reduction of the PD dataset is carried out effectively by selecting features employing 13 NI algorithms. It is worth mentioning that five of the NI algorithms employed in this study have not been studied previously in PD detection, among which the reduction performance of FPA, SSA and DE is off the mark. 11 ML classifiers, including boosting algorithms, are used for classification purposes. Each NI algorithm is compared with each of the 11 ML classifiers, distinctly keeping the same parameters. The performance of these classifiers is analysed before and after the feature selection approach with NI algorithms. Numerous assessment criteria, namely confusion matrices (TP, TN, FP, FN), accuracy, recall, precision, F1-score, fitness function, and cross-validation mean score, are used to assess feature selection and ML classification performance. The performance of different methods and the 50th, 75th and 95th percentile are used to rank. 12 pairs of NI-ML methods performed excellently, which are assessed from the 95th percentile for ranking the highest data points scored testing accuracy of 91% and above. Similarly, 34 and 25 pairs performed as good and moderate defined from the 75th and 50th percentile, which resulted in 86% and 79% accuracy, respectively. No previous study has carried out this number of NI algorithms and ML classifiers regarding PD classification. Moreover, 12 pairs of NI-ML are conveyed, with an accuracy of over 91%. In conclusion, this study brings up some newly applied NI algorithms for dimensionality reduction in PD detection and also reveals the significant effect of boosting algorithms for classification. Therefore, this will help interested researchers to understand the significant role of NI algorithms in dimensionality reduction, the performance of the five newly used NI algorithms and boosting algorithms. Thus, this study will play a pivotal role in the early diagnosis of PD.

## REFERENCES

[1] "What is parkinson's? | parkinson's foundation," https://www.parkinson.org/understanding-parkinsons/what-is-parkinsons, (Accessed on 01/27/2022).
[2] M. B. T. Noor, N. Z. Zenia, M. S. Kaiser, M. Mahmud, and S. A. Mamun, "Detecting neurodegenerative disease from mri: a brief review on a deep learning perspective," in International conference on brain informatics. Springer, 2019, pp. 115–125.
[3] M. B. T. Noor, N. Z. Zenia, M. S. Kaiser, S. A. Mamun, and M. Mahmud, "Application of deep learning in detecting neurological disorders from magnetic resonance images: a survey on the detection of alzheimer's disease, parkinson's disease and schizophrenia," Brain Inform., vol. 7, no. 1, pp. 1–21, 2020.
[4] "Modeling movement disorders in parkinson's disease using computational intelligence - white rose etheses online," https://etheses.whiterose.ac.uk/12964/, (Accessed on 01/28/2022).
[5] "Statistics | parkinson's foundation," https://www.parkinson.org/Understanding-Parkinsons/Statistics, (Accessed on 01/28/2022).
[6] "Economic burden associated with parkinson disease," https://www.patientcareonline.com/view/economic-burden-associated-parkinson-disease, (Accessed on 01/28/2022).
[7] "Parkinson's disease: health-related quality of life, economic cost, and implications of early treatment - pubmed," https://pubmed.ncbi.nlm.nih.gov/20297871/, (Accessed on 01/28/2022).
[8] "Parkinson's disease - diagnosis and treatment - mayo clinic," https://www.mayoclinic.org/diseases-conditions/parkinsons-disease/diagnosis-treatment/drc-20376062, (Accessed on 01/28/2022).
[9] M. A. Rahman, D. J. Brown, N. Shopland, A. Burton, and M. Mahmud, "Explainable multimodal machine learning for engagement analysis by continuous performance test," in Proc. HCII, 2022, pp. 386–399.
[10] M. A. Rahman, D. J. Brown, N. Shopland, M. C. Harris, Z. B. Turabee, N. Heym, A. Sumich, B. Standen, D. Downes, Y. Xing et al., "Towards machine learning driven self-guided virtual reality exposure therapy based on arousal state detection from multimodal data," in Proc. Brain Inform., 2022, pp. 195–209.
[11] G. Rabby et al., "A flexible keyphrase extraction technique for academic literature," Procedia Comput. Sci., vol. 135, pp. 553–563, 2018.
[12] G. Rabby, S. Azad, M. Mahmud, K. Z. Zamli, and M. M. Rahman, "Teket: a tree-based unsupervised keyphrase extraction technique," Cognitive Computation, vol. 12, no. 4, pp. 811–833, 2020.
[13] F. I. Adiba et al., "Effect of corpora on classification of fake news using naive bayes classifier," Int. J. Autom. Artif. Intell. Mach. Learn., vol. 1, no. 1, pp. 80–92, 2020.
[14] A. Nawar et al., "Cross-content recommendation between movie and book using machine learning," in Proc. AICT, 2021, pp. 1–6.
[15] N. Islam et al., "Towards machine learning based intrusion detection in iot networks," Comput. Mater. Contin, vol. 69, no. 2, pp. 1801–1821, 2021.
[16] F. Farhin, M. S. Kaiser, and M. Mahmud, "Secured smart healthcare system: blockchain and bayesian inference based approach," in Proc. TCCE, 2021, pp. 455–465.
[17] S. Ahmed et al., "Artificial intelligence and machine learning for ensuring security in smart cities," in Data-driven mining, learning and analytics for secured smart cities, 2021, pp. 23–47.
[18] S. Zaman et al., "Security threats and artificial intelligence based countermeasures for internet of things networks: a comprehensive survey," Ieee Access, vol. 9, pp. 94 668–94 690, 2021.
[19] T. Ghosh, M. H. Al Banna, M. S. Rahman, M. S. Kaiser, M. Mahmud, A. S. Hosen, and G. H. Cho, "Artificial intelligence and internet of things in screening and management of autism spectrum disorder," Sustain. Cities Soc., vol. 74, p. 103189, 2021.
[20] M. Biswas, M. S. Kaiser, M. Mahmud, S. Al Mamun, M. Hossain, M. A. Rahman et al., "An xai based autism detection: The context behind the detection," in Proc. Brain Informatics, 2021, pp. 448–459.
[21] A. I. Sumi et al., "fassert: A fuzzy assistive system for children with autism using internet of things," in Proc. Brain Inform., 2018, pp. 403–412.
[22] N. U. Akhund et al., "Adeptness: Alzheimer's disease patient management system using pervasive sensors-early prototype and preliminary results," in Proc. Brain Inform., 2018, pp. 413–422.

[23] M. Al Banna, T. Ghosh, K. A. Taher, M. S. Kaiser, M. Mahmud et al., "A monitoring system for patients of autism spectrum disorder using artificial intelligence," in Proc. Brain Informatics, 2020, pp. 251–262.
[24] S. Jesmin, M. S. Kaiser, and M. Mahmud, "Artificial and internet of healthcare things based alzheimer care during covid 19," in Proc. Brain Inform., 2020, pp. 263–274.
[25] S. Ahmed, M. Hossain, S. B. Nur, M. Shamim Kaiser, M. Mahmud et al., "Toward machine learning-based psychological assessment of autism spectrum disorders in school and community," in Proc. TEHI, 2022, pp. 139–149.
[26] M. Mahmud et al., "Towards explainable and privacy-preserving artificial intelligence for personalisation in autism spectrum disorder," in Proc. HCII, 2022, pp. 356–370.
[27] M. Nahiduzzaman et al., "Machine learning based early fall detection for elderly people with neurological disorder using multimodal data fusion," in Proc. Brain Inform., 2020, pp. 204–214.
[28] M. Biswas et al., "Indoor navigation support system for patients with neurodegenerative diseases," in Proc. Brain Inform., 2021, pp. 411–422.
[29] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli, "Applications of deep learning and reinforcement learning to biological data," IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 6, pp. 2063–2079, 2018.
[30] M. Mahmud, M. S. Kaiser, T. M. McGinnity, and A. Hussain, "Deep learning in mining biological data," Cogn. Comput., vol. 13, no. 1, pp. 1–33, 2021.
[31] M. Mahmud and M. S. Kaiser, "Machine learning in fighting pandemics: a covid-19 case study," in COVID-19: prediction, decision-making, and its impacts, 2021, pp. 77–81.
[32] S. Kumar et al., "Forecasting major impacts of covid-19 pandemic on country-driven sectors: challenges, lessons, and future roadmap," Pers. Ubiquitous Comput., pp. 1–24, 2021.
[33] H. R. Bhapkar et al., "Rough sets in covid-19 to predict symptomatic cases," in COVID-19: Prediction, Decision-Making, and its Impacts, 2021, pp. 57–68.
[34] M. S. Satu et al., "Short-term prediction of covid-19 cases using machine learning models," Appl. Sci., vol. 11, no. 9, p. 4266, 2021.
[35] N. Prakash et al., "Deep transfer learning for covid-19 detection and infection localization with superpixel based segmentation," Sustain. Cities Soc., vol. 75, p. 103252, 2021.
[36] A. AlArjani et al., "Application of mathematical modeling in prediction of covid-19 transmission dynamics," Arab. J. Sci. Eng., pp. 1–24, 2022.
[37] A. Paul et al., "Inverted bell-curve-based ensemble of deep learning models for detection of covid-19 from chest x-rays," Neural Comput. Appl., pp. 1–15, 2022.
[38] F. Farhin, M. S. Kaiser, and M. Mahmud, "Towards secured service provisioning for the internet of healthcare things," in Proc. AICT, 2020, pp. 1–6.
[39] M. S. Kaiser et al., "6g access network for intelligent internet of healthcare things: opportunity, challenges, and research directions," in Proc. TCCE, 2021, pp. 317–328.
[40] M. Biswas et al., "Accu3rate: A mobile health application rating scale based on user reviews," PloS one, vol. 16, no. 12, p. e0258050, 2021.
[41] Z. Soumaya, B. D. Taoufiq, N. Benayad, K. Yunus, and A. Abdelkrim, "The detection of parkinson disease using the genetic algorithm and svm classifier," Applied Acoustics, vol. 171, p. 107528, 2021.
[42] A. Ul Haq, J. Li, M. H. Memon, Z. Ali, S. Z. Abbas, S. Nazir et al., "Recognition of the parkinson's disease using a hybrid feature selection approach," Journal of Intelligent & Fuzzy Systems, vol. 39, no. 1, pp. 1319–1339, 2020.
[43] A. Pasha and P. H. Latha, "Bio-inspired dimensionality reduction for parkinson's disease (pd) classification," Health information science and systems, vol. 8, no. 1, pp. 1–22, 2020.
[44] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," Neurocomputing, vol. 172, pp. 371–381, 2016.
[45] R. Rajalaxmi and S. Kaavya, "Feature selection for identifying parkinson's disease using binary grey wolf optimization," in Proceedings of the International Conference on Intelligent Computing Systems (ICICS 2017–Dec 15th-16th 2017) organized by Sona College of Technology, Salem, Tamilnadu, India, 2017.
[46] M. Mitchell, An introduction to genetic algorithms. MIT press, 1998.
[47] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95-international conference on neural networks, vol. 4. IEEE, 1995, pp. 1942–1948.
[48] S. Rahman, T. Sharma, S. Reza, M. Rahman, M. Kaiser et al., "Pso-nf based vertical handoff decision for ubiquitous heterogeneous wireless network (uhwn)," in 2016 International Workshop on Computational Intelligence (IWCI). IEEE, 2016, pp. 153–158.
[49] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, 2010, pp. 65–74.
[50] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in 2009 World congress on nature & biologically inspired computing (NaBIC). Ieee, 2009, pp. 210–214.
[51] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization, vol. 11, no. 4, pp. 341–359, 1997.
[52] X.-S. Yang, Nature-inspired metaheuristic algorithms. Luniver press, 2010.
[53] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," Future generation computer systems, vol. 97, pp. 849–872, 2019.
[54] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," Advances in engineering software, vol. 69, pp. 46–61, 2014.
[55] C.-C. Chen, Y.-C. Tsai, I. Liu, C.-C. Lai, Y.-T. Yeh, S.-Y. Kuo, and Y.-H. Chou, "A novel metaheuristic: Jaguar algorithm with learning behavior," in 2015 IEEE international conference on systems, man, and cybernetics. IEEE, 2015, pp. 1595–1600.
[56] S. Mirjalili and A. Lewis, "The whale optimization algorithm," Advances in engineering software, vol. 95, pp. 51–67, 2016.
[57] J. Goyal, P. Khandnor, and T. C. Aseri, "Analysis of parkinson's disease diagnosis using a combination of genetic algorithm and recursive feature elimination," in 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4). IEEE, 2020, pp. 268–272.
[58] D. Gupta, S. Sundaram, A. Khanna, A. E. Hassanien, and V. H. C. De Albuquerque, "Improved diagnosis of parkinson's disease using optimized crow search algorithm," Computers & Electrical Engineering, vol. 68, pp. 412–424, 2018.
[59] D. Gupta, A. Julka, S. Jain, T. Aggarwal, A. Khanna, N. Arunkumar, and V. H. C. de Albuquerque, "Optimized cuttlefish algorithm for diagnosis of parkinson's disease," Cognitive systems research, vol. 52, pp. 36–48, 2018.
[60] P. Sharma, S. Sundaram, M. Sharma, A. Sharma, and D. Gupta, "Diagnosis of parkinson's disease using modified grey wolf optimization," Cognitive Systems Research, vol. 54, pp. 100–115, 2019.
[61] P. Sharma, R. Jain, M. Sharma, and D. Gupta, "Parkinson's diagnosis using ant-lion optimisation algorithm," International Journal of Innovative Computing and Applications, vol. 10, no. 3-4, pp. 138–146, 2019.
[62] S. Sehgal, M. Agarwal, D. Gupta, S. Sundaram, and A. Bashambu, "Optimized grass hopper algorithm for diagnosis of parkinson's disease," SN Applied Sciences, vol. 2, no. 6, pp. 1–18, 2020.
[63] R. Durgut, Y. Y. Baydilli, and M. E. Aydin, "Feature selection with artificial bee colony algorithms for classifying parkinson's diseases," in International Conference on Engineering Applications of Neural Networks. Springer, 2020, pp. 338–351.
[64] S. Dash, A. Abraham, A. K. Luhach, J. Mizera-Pietraszko, and J. J. Rodrigues, "Hybrid chaotic firefly decision making model for parkinson's disease diagnosis," International Journal of Distributed Sensor Networks, vol. 16, no. 1, p. 1550147719895210, 2020.
[65] "Uci machine learning repository," https://archive.ics.uci.edu/ml/index.php, (Accessed on 03/13/2022).
[66] "Genetic algorithms on jstor," https://www.jstor.org/stable/24939139, (Accessed on 01/29/2022).
[67] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," 1988.
[68] R. Storn, "On the usage of differential evolution for function optimization," in Proceedings of North American Fuzzy Information Processing. IEEE, 1996, pp. 519–523.
[69] X.-S. Yang, "Flower pollination algorithm for global optimization," in International conference on unconventional computing and natural computation. Springer, 2012, pp. 240–249.
[70] S. Mirjalili, "Sca: a sine cosine algorithm for solving optimization problems," Knowledge-based systems, vol. 96, pp. 120–133, 2016.
[71] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," Advances in Engineering Software, vol. 114, pp. 163–191, 2017.
[72] X. Yan and X. Su, Linear regression analysis: theory and computing. World Scientific, 2009.

[73] J. R. Quinlan, "Simplifying decision trees," International journal of man-machine studies, vol. 27, no. 3, pp. 221–234, 1987.
[74] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, no. 3, pp. 273–297, 1995.
[75] T. K. Ho, "Random decision forests," in Proceedings of 3rd international conference on document analysis and recognition, vol. 1. IEEE, 1995, pp. 278–282.
[76] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," International Statistical Review/Revue Internationale de Statistique, vol. 57, no. 3, pp. 238–247, 1989.
[77] Y. Freund, R. E. Schapire et al., "Experiments with a new boosting algorithm," in icml, vol. 96. Citeseer, 1996, pp. 148–156.
[78] T. Hastie, R. Tibshirani, and J. Friedman, "Boosting and additive trees," in The elements of statistical learning. Springer, 2009, pp. 337–387.
[79] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.
[80] "1.9. naive bayes — scikit-learn 1.0.2 documentation," https://scikit-learn.org/stable/modules/naive_bayes.html, (Accessed on 03/11/2022).