Complex & Intelligent Systems
https://doi.org/10.1007/s40747-023-01118-z

ORIGINAL ARTICLE

Addressing feature selection and extreme learning machine tuning by diversity-oriented social network search: an application for phishing websites detection

Nebojsa Bacanin[1] · Miodrag Zivkovic[1] · Milos Antonijevic[1] · K. Venkatachalam[2] · Jinseok Lee[3] · Yunyoung Nam[4] · Marina Marjanovic[1] · Ivana Strumberger[1] · Mohamed Abouhawwash[5,6]

Received: 16 August 2022 / Accepted: 16 May 2023

Abstract
Feature selection and hyper-parameters optimization (tuning) are two of the most important and challenging tasks in machine learning. To achieve satisfying performance, every machine learning model has to be adjusted for a specific problem, as the efficient universal approach does not exist. In addition, most of the data sets contain irrelevant and redundant features that can even have a negative influence on the model's performance. Machine learning can be applied almost everywhere; however, due to the high risks involved with the growing number of malicious, phishing websites on the world wide web, feature selection and tuning are in this research addressed for this particular problem. Notwithstanding that many metaheuristics have been devised for both feature selection and machine learning tuning challenges, there is still much space for improvements. Therefore, the research exhibited in this manuscript tries to improve phishing website detection by tuning extreme learning model that utilizes the most relevant subset of phishing websites data sets features. To accomplish this goal, a novel diversity-oriented social network search algorithm has been developed and incorporated into a two-level cooperative framework. The proposed algorithm has been compared to six other cutting-edge metaheuristics algorithms, that were also implemented in the framework and tested under the same experimental conditions. All metaheuristics have been employed in level 1 of the devised framework to perform the feature selection task. The best-obtained subset of features has then been used as the input to the framework level 2, where all algorithms perform tuning of extreme learning machine. Tuning is referring to the number of neurons in the hidden layers and weights and biases initialization. For evaluation purposes, three phishing websites data sets of different sizes and the number of classes, retrieved from UCI and Kaggle repositories, were employed and all methods are compared in terms of classification error, separately for layers 1 and 2 over several independent runs, and detailed metrics of the final outcomes (output of layer 2), including precision, recall, f1 score, receiver operating characteristics and precision–recall area under the curves. Furthermore, an additional experiment is also conducted, where only layer 2 of the proposed framework is used, to establish metaheuristics performance for extreme machine learning tuning with all features, which represents a large-scale NP-hard global optimization challenge. Finally, according to the results of statistical tests, final research findings suggest that the proposed diversity-oriented social network search metaheuristics on average obtains better achievements than competitors for both challenges and all data sets. Finally, the SHapley Additive exPlanations analysis of the best-performing model was applied to determine the most influential features.

Keywords Extreme learning machine · Feature selection · Metaheuristics optimization · Social network search · Hyper-parameters optimization · Diversification

✉ Nebojsa Bacanin
nbacanin@singidunum.ac.rs

✉ Yunyoung Nam
ynam@sch.ac.kr

Extended author information available on the last page of the article

Introduction

The sphere of network research and engineering that has, in previous decades, led to the development of the world wide web (WWW) has seen a constant stream of innovations, development, and improvements. The web has moved from a niche technology to a staple of everyday life. With these developments came user convenience and many services migrated to hybrid and even entirely online models [1–3]. Shopping, trading, baking, business meetings, and many other operations that deal with sensitive data on an everyday basis now take place online [4, 5]. With all this in mind, it is worth noting that malicious actors exist, and they prioritize their interests over the privacy and security of others [6, 7]. Malicious actors make use of many techniques and tools during their operations.

Depending on the current goal, the methods used can vary from intrusion tools to scanning and probing tools used to evaluate systems for vulnerabilities. For example, by posing as a trustworthy entity, an attacker can use social engineering techniques which involve tricking users into disclosing sensitive information, such as login credentials or personal information. The wide range of publicly available phishing kits makes the job fairly easy even for the less technically proficient actors. Other well know and applied methods for obtaining sensitive information include using spoofing techniques to create fake websites or emails that mimic legitimate ones, or sending malicious software (malware) that can infect a user's computer and give the attacker unlimited access [8, 9]. It is also important to note that these techniques evolve along with the development of new methods used for detection and counteraction. Researchers need to constantly remain several steps ahead of malicious actors to maintain a platform secure enough to support the convinces that make the Web essential to modern life.

The field of artificial intelligence (AI) has seen many developments in recent years due to the wide adaption of computation across multiple fields, for example, business [10], finance [11], various medical [12], and many other fields [13], rely on the AI in daily operations. Accordingly, AI presents many approaches to addressing real-world problems. Various methods tackle tasks differently, and with varying degrees of success. The ability of AI to learn and adapt to a changing landscape makes it a promising candidate for addressing problems in the ever-developing field of web and network security in general. While traditional methods, such as firewalls, security certificates, blacklists, and others, exist [14–17], they require constant monitoring and maintenance to maintain an acceptable level of efficacy. By applying AI to these problems, researchers have attempted to improve existing models and provide an overall improvement in network security. Many such applications exist that tackle intrusion detection [18–20], detect attempts on exploiting users through phishing attacks [21–23], uncover embedded botnets in IoT networks [24–26], as well as tackle vectors of spread for malicious software such as spam [27–29].

The AI field in general can be roughly divided into two categories: machine learning (ML) and metaheuristics optimization algorithms. Machine learning and metaheuristics differ in terms of their objective and methodology. The ML aims to recognize patterns and correlations in data to facilitate forecasting or decision-making, whereas metaheuristics are employed to discover effective solutions for optimization challenges. The ML algorithms are typically trained on vast data sets to enhance their precision, whereas metaheuristics do not necessitate big data sets and are usually more computationally economical. In general, ML and metaheuristics are both important tools in the field of AI, but they serve different purposes and employ different approaches to problem-solving.

A popular group of ML techniques, that models principles observed in human brains are artificial neural networks (ANN) [30]. They present a simplified model of the internal mechanisms observed in various nerve clusters. Neural networks remain popular due to their ability to tackle nonlinear approximations by processing input data. In addition, their versatility enables them to tackle a wide range of problems, that are otherwise challenging to address using traditional methods [31, 32]. However, despite many advantages, neural networks are computationally demanding, making them slower to train and evaluate. This led to many researchers developing various methods for improving their performance while preserving the positive traits that make them appealing [33–35].

One of the most efficient ANN types is the extreme learning machine (ELM) model, because it does not require traditional training, which is time and computationally consuming [36]. The ELM was originally proposed as a method for training single-hidden layer feed-forward networks (SFFN) [36]. It makes use of the Moore–Penrose (MP) generalized inverse to calculate output weights, while the input and hidden layer biases are generated randomly. The approach attempts to avoid problems present in traditional gradient-descent algorithms, e.g., getting stuck in local minima, vanishing and exploding gradients, and improve on the much slower convergence rate, providing better general performance.

However, ML like every other field suffers from some challenges and two of the most important ones are feature selection and tuning (hyper-parameters optimization). To achieve satisfying performance, every machine learning model has to be adjusted for a specific problem, as according to the no free lunch (NFL) theorem [37], the efficient universal approach for all practical challenges does not exist. In addition, most of the data sets contain irrelevant and redundant features that can even have a negative influence on the model's performance. In most cases, both above-mentioned ML challenges are NP-hard instances and since metaheuristics have proven as successful NP-hard problem solvers [38, 39], this is where these two groups of AI methods can be hybridized.

The ELMs can be tuned and adjusted for specific problems in two ways. First, the number of hidden neurons needs to be high enough to ensure good generalization, conversely, network structures with only a few neurons in the hidden layer may exhibit degraded performance. Second, the model's performance, in terms of classification/regression metrics, depends to a large extent on the values of randomly initialized weights and biases. Every problem is specific and since the ELMs do not undergo traditional training, e.g., using stochastic gradient descent (SGD) based algorithms, the final output depends on initial randomly generated weights and biases values. Therefore, determining adequate hidden input weights and biases for each specific practical application presents an additional and important optimization challenge in this area.

Population-based metaheuristic algorithms have even been successfully utilized to address NP difficult problems, considered impossible to solve with traditional computational methods [40]. By mathematically modeling behaviors of individual agents, that obey simple rule sets, and with the use of an objective evaluation function, metaheuristics enable complex behaviors to manifest on a larger scale. A promising novel population metaheuristics for addressing optimization problems is the social network search (SNS) metaheuristics [41]. It simulates user interactions on social networks in a simplified manner by modeling user moods. In turn, replicating the flow of information and propagation of ideas and views. The way by which users generate popularity on social networks forms the basis of the functionality of this approach. These mechanisms make the novel algorithm an attractive option for researchers tackling various optimization problems as it shows promising results under test conditions [41].

The motivation, as the goal, behind the research proposed in this manuscript is to try to further improve phishing website detection by tuning the ELM, which utilizes the most relevant subset of features in the available phishing data sets. Such motivation stems from two facts: notwithstanding that many approaches have been devised for both, feature selection and ML tuning challenges, there is still much space for improvements, because the method that obtains satisfying accomplishments for every practical task cannot be created [37]; one of the most important challenges on the web is phishing and developing intelligent ML classifier for detecting such malicious websites is among top priorities in this domain [42–44].

A novel diversity-oriented SNS algorithm has been developed to accomplish above mentioned objective and integrated into a two-level cooperative framework for feature selection and ELM tuning. Layer 1 of devised framework performs feature selection, while the best-obtained subset of features has then been used as the input to the framework's level 2, where metaheuristics perform tuning of ELM, respecting to the number of neurons in the hidden layer and weights and biases initialization. Notwithstanding that wrapper-based feature selection can be computationally demanding [45], the level 1 of proposed framework implements wrapper feature selection approach, because compared to other approaches for this challenge it obtains better performance in terms of classification accuracy and smaller chosen subset of relevant features.

In addition, to establish metaheuristics performance for a larger scale NP-hard challenge, a second experiment, that uses only the layer 2 part of the framework, where the ELM was tuned with all available features, is also conducted. The proposed boosted SNS algorithm has been compared to six other cutting-edge metaheuristics algorithms, that were also implemented in the framework and tested under the same experimental conditions. Finally, the SHapley Additive exPlanations (SHAP) analysis was applied to the proposed model, aiming to interpret the best performing model and to discover the most influential features of two considered data sets.

Based on everything stated so far, the main contributions of this work may be summarized as the following:

1. proposal of a novel SNS algorithm based on the solution diversity adapted for feature selection and ELM tuning
2. implementation of the cooperative two-layer framework that performs feature selection (layer 1), and ELM optimization (layer 2)
3. integration of developed enhanced SNS in the framework to tackle a practical issue of phishing website detection.

The remainder of this work is structured as follows: Sect. 2 covers works related to the subject matter followed by a summary of the concepts behind ELM. Section 3 presents methods proposed in this research, followed by an overview of the experimental setup and discussion, as well as validation, of the attained results in Sect. 4. Finally, the conclusion and possible future works in the field are given in Sect. 5.

Literature review and preliminaries

This section provides basic background information related to proposed research. First, a concept of feature selection is introduced, followed by ELM mathematical formulation and details of metaherustics optimization methods, along with relevant literature review.

Feature selection

The ML models frequently try to find useful patterns and connections in large data sets, packed with redundant and inessential data, that have a profound influence concerning models' accuracy and computational complexity. Frequently, the said data sets are high-dimensional, which also impedes ML model performance. This occurrence refers to the curse of dimensionality [46].

Hence, identifying essential information is crucial to tackling this issue. For this reason, the technique of dimensionality reduction [47], where the classification problem is simplified, is a main pre-processing task for machine learning. It is an action, where the data are transformed from a high-dimensional domain to a lower dimension by reducing the number of classification variables while still keeping enough meaningful attributes of the original data set. There are two approaches to dimensionality reduction: feature extraction and feature selection. While feature extraction [48] generates new variables derived from the primary set of data, feature selection chooses a subset of relevant informative variables for the desired objective.

The purpose of feature selection is to determine the relevant subset from high-dimensional data sets eliminating the insignificant features, thus enhancing the classification accuracy for ML. There are three feature selection methods: filter, wrapper, and embedded methods, as per [45].

Wrapper methods utilize learning algorithms, like ML classifiers, to evaluate feature subsets to find relevant features, i.e., the feature selection challenge is treated as search problem. Wrapper methods use a specific ML algorithm as a black-box model to evaluate the usefulness of each feature subset. This method involves evaluating each feature subset by training a machine learning model on the subset and measuring its performance. The performance of the model is then used as a criterion for selecting the best feature subset. Wrapper methods tend to provide better results than filter methods in terms of classification accuracy and smaller feature subsets, but they can be computationally expensive as they involve training and evaluating a model for each subset of chosen features [45].

Filter methods do not use a training process and instead designate a score to feature subsets using statistical or mathematical metrics that evaluate the relationship between each feature and the target variable. Filter methods consider the relevance and redundancy of features, and those with the highest scores are selected for further analysis. Due to this property, this method is not as computationally demanding as the wrapper and can be applied to a broad range of data sets, making them a popular choice for feature selection in many applications [45, 49].

Finally, the embedded method employs feature selection as a segment of the model creation process, i.e., methods perform feature selection during the model training. These methods are more accurate than filters, with the same execution speed. With computational complexity in mind, embedded methods are in between the methods mentioned above.

Extreme learning machine (ELM)

A relatively novel learning algorithm, ELM is applied to training single-layer feed-forward network (SLFN) [50]. In the initial stage, the algorithm randomly initializes weights and biases for hidden layer neurons. This is followed by computational steps to determine output weights by applying the MP generalized inverse. Hidden neuron layer randomization presents an interesting and demanding challenge for researchers. This hidden layer transforms input values into higher dimensional ELM feature space, using nonlinear transforms. With this approach, the process of attaining a solution is simplified, since the probability of linear separability of inputs across feature space increases.

With a training set $\aleph = (x_i, t_i) | x_i \in R^d, t_i \in R^m, i = 1, ..., N$, the output with L hidden neurons, using $g(x)$ as the activation function, can be determined according to Eq. (1):

$$\sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = y_j,\ j = 1, \ldots, N \tag{1}$$

where b bias and $w_i = [w_{iq}m, \ldots, w_{id}]^T$ are the weights of a hidden neuron. Output weights are represented by $\beta_i = [\beta_{i1}, \ldots, \beta_{im}]^T$, and $w_i \cdot x_j$ is the inner product of w_i and x_j.

Standard SLNF parameters $\beta_i, i = 1, \ldots L$ can be approximated to Eq. (2):

$$\sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = t_j,\ j = 1, \ldots, N \tag{2}$$

In addition, the parameter T can be computed according to Eq. (3):

$$T = H\beta \tag{3}$$

where H is the hidden layer output matrix seen as in Eq. (4):

$$h = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \ldots & g(w_L \cdot x_1 + b_L) \\ \vdots & \ldots & \vdots \\ g(w_1 \cdot x_N + b_1) & \ldots & g(w_L \cdot x_N + b_L) \end{bmatrix}_{N \times L} \tag{4}$$

Furthermore, β and T are shown in Eq. (5):

$$\beta = \begin{bmatrix} \beta_i^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad \text{and } T = \begin{bmatrix} t_i^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m} \tag{5}$$

Output weights, represented β are calculated using Eq. (6)

The pseudocode for the ELM method applied to SLFN can be seen in Algorithm 1.

Algorithm 1 Pseudocode for the ELM

```
With training data $\aleph = (x_i, t_i) | x_i \in R^d, t_i \in R^m, i = 1, ...., N, g(x)$ as the activation function, and $\tilde{N}$ the as the number of hidden nodes
Step 1: Assign $w_i$ and $b_i$ randomly
Step 2: Compute hidden layer output matrix $H$
Step 3: Compute output weight $\beta$
```

where β is calcuated according to Eq. (6):

$$\beta = H^{\dagger} T \tag{6}$$

with $H^{\dagger}$ representing the Moore–Penrose generalized inverse of H.

It is worth noting that many ELM applications and improved ELM approaches are presented in modern literature. Some of the more notable applications and variations include: pattern recognition, where neurons deemed irrelevant are eliminated via pruning (pruned ELM–P-ELM) [51]; a proposal of an improved method called the optimally pruned ELM (OP-ELM) [52], which constructs larger networks by applying the standard ELM algorithm, has been applied to regression and classification problems; implementation of evolutionary ELM (E-ELM) that takes advantage of differential evolution (DE) algorithm to make adequate adjustments to weights and biases within a network [53].

Metaheuristics optimization

When dealing with optimization problems, metaheuristic algorithms are a popular choice among researchers due to their ability to address complex problems as well as higher dimensional data sets, within acceptable time periods and realistic resources. A subgroup among these is population-based algorithms, with the most prominent representative, swarm intelligence [54], that mimics a group of organisms from nature. Often nature-inspired, metaheuristics model the actions of individuals with simple rules. By following these rule sets, separate, independent agents act as a cooperative group, working towards a common goal, often aided by external objective (fitness) functions that are used to assess its performance. This mechanism allows for complex behaviors to manifest on a global scale. The exact mechanisms by which optimization is achieved depend on the details of the selected algorithm.

However, most metaheuristics rely on exploration and exploitation as primary internal mechanisms [55]. In the exploration stage, the algorithms focus on broadly covering a large area of the search space, looking for promising regions. When certain criteria are met, the algorithm transitions toward exploitation. In this stage, the primary goal is to increase the resolution of results around promising areas to gain more favorable outcomes. Due to the stochastic nature of these stages, an optimal solution is not guaranteed; however, a satisfying solution can be generated within a reasonable time span [56, 57].

Metaheuristic algorithms have been used to successfully tackle problems considered NP-hard which are often impossible to resolve using traditional methods [58–60]. As such, they continue to be a popular choice for researchers owing in part to their problem-solving abilities, relative simplicity, and low computational demands. Many algorithms have been developed to model various observed phenomena.

Some notable nature-inspired algorithms, that fall into the category of swarm intelligence, include the artificial bee colony (ABC) [61] algorithm, often used to improve performance through many practical applications [62–64]. Other examples include the whale optimization algorithm (WOA) [65], which models a hunting strategy unique to a species of whale. This approach is popular among researchers for its interesting search patterns and has successfully been applied to many real-world problems [39, 66, 67]. In addition, the gray wolf optimization (GWO) [68] also relies on modeling hunting techniques and has likewise proven effective when applied to real-world challenges [69–71]. Firefly algorithm (FA), introduced by [72], mimics the social behavior of fireflies, and is considered to be a very powerful optimizer, with a wide range of recent applications that include credit card fraud detection [73], intrusion detection classifier optimization [74], medical diagnostics [75], neural networks optimization [35, 76, 77], plant classification [78], and many others.

Other population-based metaheuristics have been inspired by basic mathematical principles. For example, the sine cosine algorithm (SCA) [79], relies on trigonometric formulas as its primary mechanisms for exploration and exploitation. Researchers have successfully applied the SCA in various fields with favorable results [80–82]. Another example includes the arithmetic optimization algorithm (AOA) [83], which makes use of arithmetic operations as an inspiration. This method has also been applied to various tasks with promising outcomes [84–86].

Metaheuristic algorithms inspired by social interactions form the basis of social-based metaheuristics [87]. These algorithms make use of social interaction models as the basis for their function. They sometimes draw inspiration from purely social adaptations or tactical approaches found in sports and competitions. Some notable examples include the teaching-learning-based optimization (TLBO) [88], which simulates the effect teachers have on their pupils in a school environment that is a representation of the algorithms population. When a teacher shares information with a student the quality of teaching affects the student's grade represented in the fitness value. This approach has given promising results when applied to resolving complex problems [89–91]. One more metaheuristics example that models social interactions is league championship algorithm [92], and it also gives promising results when applied to real-world problems [93–95].

One of the most popular research fields currently is endeavors in hybridization between population-based metaheuristics and ML. Numerous recent publications show very successful hybridization of the ANN with swarm intelligence algorithms. Some of the most representative applications include COVID-19 cases prediction [96], classification and severity estimation [97–100], brain tumor classification [31, 101], cryptocurrencies trends estimation [102], intrusion detection [103–105], and many others.

It is also worth noting that ELM has also recently been subjected to swarm intelligence optimization, as several recent research suggests [106–108]. Moreover, in the modern literature, many population-based approaches that perform feature selection using various machine learning models can be identified [109, 110].

Proposed method and simulation framework

This section first presents the original implementation of the SNS algorithm, followed by the observed drawbacks of the basic version. Finally, the novel improved version of the SNS algorithm is provided along with details of two-level framework used for simulation and solution's encoding strategy.

Social network search

Social behaviors are part of human nature, and in the modern communication age, these behaviors have adapted to informational technologies. Social media networks have become a part of everyday life. Just as technology adapted to human nature, humans adjusted social behaviors to emerging technologies, developing methods for interaction on social networks. The SNS algorithm models the methods users on social networks implement to gain popularity. This is done through the implementation of the user's moods in the algorithm. These moods guide the behavior of simulated users and accordingly govern the behavior of the algorithm [41].

In the algorithms model, simulated user actions are affected by the mood of those around them. These moods form simplified representations of those seen in real life and include imitation, conversation, disputation, and innovation.

Imitation models are one of the main characteristics present in social media. Users follow friends and family as well as people they like. When a user shares a new post, users following them get informed of this and are given the opportunity of perpetuating this information by sharing. If the opinions expressed in this post pose challenging concepts, users will strive to imitate their views and post similar topics. The mathematics behind this model is expressed in Eq. (7):

$$\begin{aligned} &X_{i\ new} = X_j + rand(-1,1) \times R \\ &R = rand(0,1) \times r \\ &r = X_j - X_i, \end{aligned} \tag{7}$$

with X_j being the jth users view vector selected at random where $i \neq j$. Accordingly, X_i represents the view vector of the ith user. Random value selection is denoted with $rand(-1, 1)$ and $rand(0, 1)$, representing the section or a random value with intervals of $[-1, 1]$ and $[0, 1]$ respectfully. While in the imitation mood, new solutions will be created in the imitation space, using a mixture of radii of shock and popularity. The radius of shock R is proportional to the influence of the jth user, while its magnitude is based on a multiple of r. Likewise, the value of r is dependent on the popularity of the jth user and is computed according to differences in opinions of these users. The final influence of the shock is determined by multiplying the random vector value $[-1, 1]$, with positive component values if the shared opinions match, and negative values if they do not.

Social networks encourage communication between individual users about different issues. This is mirrored by the algorithm in the conversation mood. In this mood, simulated users learn from one another exchanging information privately. Through conversations, users gain insight into differences in their opinions. This allows them to draft a new perspective on issues. This mechanism is modeled by Eq. (8):

$$\begin{aligned} &X_{i\ new} = X_k + R \\ &R = rand(0,1) \times D \\ &D = sign(f_i - f_j) \times (X_j - X_i), \end{aligned} \tag{8}$$

in which X_k is the randomly selected issues vector, R is the chats effect based on the difference in opinions and represents the opinion change towards X_k. The difference in opinions is represented by D. A random value selection between $[0, 1]$ is represented by $rand(0, 1)$. In addition, chat vectors of randomly selected users i and j are represented by X_i and X_j, respectively. It should be noted that when selecting users $j \neq i \neq k$. The $sign(f_i - f_j)$ determines the direction of X_k via comparison. Should a user's opinion change through conversation it is considered a new view and is accordingly shared.

While in the disputation mood users elaborate and defend opinions among themselves. On social networks, this is usually done through a debate in comment or group chat sections, where users with differing views may be influenced by the reasoning of others. In addition, users may form friendly relations between themselves, forming additional discussion groups. The algorithms model this mood by considering a random number of users as cementers or group members. The way opinions are shaped during this process is depicted in Eq. (9):

$$X_{i\ new} = X_i + rand(0,1) \times (M - AF \times X_i)$$
$$M = \frac{\sum_t^{N_r} X_t}{Nr} \tag{9}$$
$$AF = 1 + round(rand),$$

with X_i representing the ith users view vector, $rand(0,1)$ a random vector with the interval [0, 1], M being the mean value of views. The admission factor AF is a random value of 1 or 2 and represents the users having their own opinions discussed among peers. The input is rounded to the nearest integer using the $round()$ function, while $rand$ represents a random value in the interval [0, 1]. The number of comments or members of a discussion group is depicted in N_r and is a random value between 1 and N_{user}, where N_{user} is the total number of users of the social network.

When considering an issue, occasionally users share original concepts when they can understand the nature of the problem differently. This process is modeled through the innovation mood. Sometimes a specific topic may pose different features, with each one affecting the issue as a whole. By questioning ideas behind the established norm, the fundamental way of understanding might change, resulting in a novel view. This approach is modeled by the algorithm in the innovation mood and can be mathematically expressed according to Eq. (10):

$$X_{i\ new}^d = t \times x_j^d + (1-t) \times n_{new}^d$$
$$n_{new}^d = lb_d + rand_1 \times (ub_d - lb_d) \tag{10}$$
$$t = rand_2,$$

where d stands for the dth randomly chosen variable withing the $[1, D]$ interval, and D representing the number of variables available for the current problem. Additional random values in in the [0, 1] are represented by $rand_1$ and $rand_2$. Minimum and maximum values for the dth value of n_{new}^d are represented by ub_d and lb_d respectfully. The current idea for the dth idea is represented by X_j^d, with the j user selected randomly, so that $j \neq i$. Should the ith user change their opinion, a new idea is formed and becomes n_{new}^d. Finally, a new view x_{new}^d is formed on the dth dimension as a new interpolation of the current idea.

The dimension shift in x_{new}^d introduces an overall switch in concept and may be considered a new view to share. This mechanism is modeled according to Eq. (11):

$$X_{i\ new} = [x_1, x_2, x_3, \ldots, x_{i\ new}^d, \ldots, x_D], \tag{11}$$

where, as in previous equations, $x_{i\ new}^d$ represents a new insight on an issue for the dth point of view, replaced with the current one x_i^d.

To construct the initial network, parameters for the number of users, maximum iterations as well as limits are set. Each initial view is created according to Eq. (12):

$$X_0 = lb + rand(0,1) \times (ub - lb) \tag{12}$$

in which X_0 stands for each users initial view vector, a random value in the interval [0, 1] is represented by $rand(0,1)$, and upper and lower bounds of each parameter are represented as ub and lb, respectively.

When addressing maximization problems, Eq. (13), is used by the algorithm to limit views:

$$X_i = \begin{cases} X_i, & f(X_i) < f(X_{i\ new}) \\ X_{i\ new}, & f(X_{i\ new} \geq f(X_i)) \end{cases} \tag{13}$$

Deficiencies and complexity of the social network search Algorithm

The original implementation of SNS has been established as an exceptional metaheuristics with respectable performance, yet, certain drawbacks have been noticed. Due to the lack of exploration power in the early rounds of the execution, the basic SNS tends to linger in the sub-optimal areas of the search domain, as a result of premature convergence. As a consequence, the resulting solutions' quality is not satisfactory.

According to the previous research with the SNS algorithm and additional simulations against large-scale optimization challenges for the purpose of this research (benchmark CEC bound-constrained testing suite and practical ELM tuning task), the most significant cons of the basic SNS algorithm are weak exploration mechanism and the inappropriate trade-off between exploration and exploitation [73, 111, 112]. Conversely, the SNS local search process is guided by relatively strong exploitation and population diversity condenses rapidly throughout the algorithm run.

In runs, when the algorithm can not find a proper search space part in early iterations, the whole population convergences towards sub-optimum domains, rendering final results which are too far from optimum. These scenarios are facilitated by above mentioned relatively strong SNS exploitation. However, when the algorithm 'is lucky' and the initial random population is generated around the optimum region, guided by strong exploitation, fine-tuned search around the promising solutions is conducted even in early iterations and final rendered results are of high quality.

Concerning the complexity of the basic SNS algorithm, the original publication [41] examines it on two levels, during initialization and within the main loop of the algorithm. At the start of the algorithm execution, during initialization, a new pseudo-random population of individuals is produced, followed by the evaluation of the solutions. The complexity of producing the random individuals is $O(NP \cdot D)$, NP denoting the count of users, while D represents the problem's dimensionality. The fitness function evaluation complexity is obtained by $O(NP) \cdot O(F(x))$, where $F(x)$ marks the objective function being optimized.

The level of popularity is obtained through the main loop that iterates T times (T denotes the maximum number of iterations), producing a new solution for every user as the new view and evaluating it in every iteration. Therefore, the complexity of the main loop can be defined as $O(T \cdot NP \cdot D)$, and the complexity of fitness functions evaluations over the rounds can be determined as $O(T \cdot NP) \cdot O(F(x))$.

According to the common practice of establishing computational exhaustiveness of metaheuristics, complexity is often calculated with respect to fitness function evaluations ($FFEs$), as it is the most complex computing operation [113]. Accordingly, in terms of $FFEs$, the complexity of SNS metaheuristics can be observed as $O(SNS) = O(NP) + O(T \cdot NP)$.

It can be concluded that, despite observed drawbacks and taking into account relatively low computational complexity, the SNS is a promising algorithm for solving NP-hard optimization tasks; however, there is space for its enhancements.

Improved SNS algorithm

As it was already pointed out, the fact that SNS exhibits strong exploitation combined with weak exploration, leads the to low population diversity and premature convergence. In other words, the final results at large extent depend on the solutions' positions in the initial random population.

Improved SNS algorithm proposed in this research attempts to tame the lack of exploration through the establishment of the adequate population diversity during the initialization procedure and through the entire algorithm's run. With this in mind, two modifications are introduced in the original SNS metaheuristics: novel initialization scheme and an instrument to maintain the satisfactory solutions diversity throughout the whole run of the algorithm.

Novel initialization scheme

The algorithm suggested in this research utilizes a traditional initialization equation for creating the solutions in the initial population:

$$X_{i,j} = lb_j + \psi \cdot (ub_j - lb_j), \tag{14}$$

where $X_{i,j}$ denotes the jth component of ith individual, lb_j and ub_j define the lower and upper boundaries of parameter j, while ψ represents a pseudo-random value derived from the normal distribution in range [0, 1].

Nevertheless, as demonstrated in [114], it is possible to cover wider area of the search domain if the quasi-reflection-based learning (QRL) procedure is added to the individuals created by Eq. (14). Consequently, for each individual's parameter j (X_j), a quasi-reflexive-opposite element (X_j^{qr}) is obtained by

$$X_j^{qr} = \text{rnd}\left(\frac{lb_j + ub_j}{2}, x_j\right), \tag{15}$$

where the function rnd is utilized to select the pseudo-random value within $[\frac{lb_j + ub_j}{2}, x_j]$ range.

According to the QRL, the proposed initialization scheme is not introducing the additional overhead to the algorithm in respect of $FFEs$, as it starts by initializing only $NP/2$ individuals. Utilized initialization scheme is presented in Algorithm 2.

Algorithm 2 Initialization scheme based on QRL procedure pseudo-code

```
Step 1: Generate population $P^{init}$ of $NP/2$ individuals by applying Eq. (14)
Step 2: Generate QRL population $P^{qr}$ from the $P^{init}$ using Eq. (15)
Step 3: Produce starting population $P$ by merging $P^{init}$ and $P^{qr}$ $(P \cup P^{qr})$
Step 4: Obtain fitness for each individual in $P$
Step 5: Sort all solutions in $P$ in respect of fitness
```

According to the results presented in the experimental section, the introduced initialization scheme contributions are twofold. It enhances the diversification at the start of the algorithm's run, and it also provides an initial boost for the search procedure, because with the same number of individuals larger search space is covered.

Mechanism for maintaining population diversity

One way to monitor the converging/diverging extent of the algorithm's search procedure is diversification of the population, as described by [115]. The approach taken in this paper utilizes one recent definition of the population diversity metrics, the $L1$ norm [115], which includes diversities obtained over two components—individuals in the population and the dimensionality of the problem.

The research presented in [115] also indicates the importance of data obtained from the dimensionwise component of the $L1$ norm, that can be used to evaluate the search process of the tested algorithm.

Assuming that m represents the number of individuals in the population, and n defines the dimensionality of the problem, the $L1$ norm is formulated as shown in Eq. (16 - 18):

$$\overline{x_j} = \frac{1}{m}\sum_{i=1}^{m} x_{ij} \tag{16}$$

$$D_j^p = \frac{1}{m}\sum_{i=1}^{m}\left|x_{ij} - \overline{x}_j\right| \quad (17)$$

$$D^p = \frac{1}{n}\sum_{i=1}^{n} D_j^p, \quad (18)$$

where $\overline{x}$ denotes the vector with mean positions of the solutions in each dimension, D_j^p represents the solutions' position diversity vector as $L1$ norm, and D^p marks the diversity value, as a scalar value, for the entire populace.

During the early rounds of the algorithm's execution, where the individuals in the population are produced by utilizing the common initialization equation (14), the diversity of the entire population should have a high value. Still, in later rounds, when the algorithm is converging to optimum (sub-optimum), this value should be decreased dynamically. Introduced improved SNS method uses the $L1$ norm to manage the population's diversity during the execution of the algorithm by applying the dynamic diversity threshold (D_t) parameter.

The mechanism for maintaining population diversity takes into account an auxiliary control parameter, nrs that denotes the number of replaced solutions. This mechanism is being executed as follows: at the start of the algorithm's run, the initial value of D_t (D_{t0}) is obtained; in every iteration, the condition $D^P < D_t$ is examined, where D^P represents current population diversity; if the condition is fulfilled (diversity of the population is not satisfactory), nrs of the worst individuals are substituted by the random solutions, produced analogously as in the population initialization expression.

With respect to the empirical simulations and theoretical analysis, the expression that is used to obtain D_{t0} can be formulated in the following way:

$$D_{t0} = \sum_{j=1}^{NP} \frac{(ub_j - lb_j)}{2 \cdot NP} \quad (19)$$

Equation (19) follows the assumption that most of the individuals' components will be produced in the proximity of the mean of the lower and upper boundaries of the parameters, as given by Eq. (14) and Algorithm 2. Nevertheless, as the algorithm progresses, assuming that the population will move in the direction of the optimal domain of the search region, the D_t should decrease from the starting value $D_t = D_{t0}$ in the following way:

$$D_{t+1} = D_t - D_t \cdot \frac{t}{T}, \quad (20)$$

where t and $t+1$ denote current and subsequent iterations, respectively, while T represents the maximum number of rounds in a single run. Consequently, as the algorithm progresses, the D_t is dynamically decreasing, and at the end, above described method will not be triggered, regardless of the D^P.

Inner workings and complexity of the proposed algorithm

With respect to the introduced modifications, novel proposed SNS metaheuristic has been named diversity oriented SNS (DOSNS). The computational complexity of the suggested DOSNS is not greater than the basic SNS in terms of $FFEs$. First of all, novel initialization scheme does not impose additional $FFEs$ overhead. Similarly, mechanism for maintaining population diversity replaces nrs worst solutions regardless newly generated random individuals have better or worse fitness; therefore, its eligibility is not validated. Finally, the complexity of the DOSNS in terms of the $FFEs$ can be expressed as: $O(DOSNS) = O(NP) + O(T \cdot NP)$.,

The DOSNS inner-working procedures are provided in Algorithm 3. As it can be seen from the proposed pseudocode introduced modification are additions to the original SNS algorithm [41].

Feature selection and tuning framework

As described earlier, the proposed DOSNS algorithm was adapted to address the feature selection and ELM's hyperparameters tuning tasks, as part of the hybrid framework between metaheuristics and ML, that has been developed. Tuning of the ELM takes into account both, determining the number of neurons in the hidden layer (nn), and weights and biases values initialization, that link the input features with the hidden layer.

Devised framework consists of two levels—level 1 (L1), that deals with feature selection and level 2 (L2), that performs ELM tuning. Levels in two-level framework can be used independently, i.e., performing only feature selection or ELM tuning. In addition, the L1 can execute in a cooperative or individual mode. When set to cooperative mode, all metaheuristics included in the framework perform feature selection independently; however, at the end of execution (after predetermined number of runs), selected feature subset generated by best performing metaheuristics is used as the input to L2 and then all metaheuristics perform ELM tuning using the same set of selected features. Conversely, if the L1 is set to individual mode, than all metaheuristics use their own best set of selected features from L1, as an input to L2, regardless of the classifier performance with the chosen set of features.

As already emphasized in Sects. 1 and 2.1, notwithstanding that the wrapper-based feature selection can be computationally demanding [45], the L1 of proposed framework implements wrapper feature selection approach, because

Algorithm 3 Pseudocode for the DOSNS algorithm

```
Set number of solutions (users), $T$, $lb$, $ub$, $t = 0$
Initialize starting population $P$ of $NP$ solutions according to Algorithm 2
Determine values of $D_{t0}$ and $D_t$
Evaluate each user (solution) according to objective function
$i = 0$
do
    if ($i \geq N$) then
        $t = t + 1$
        $i = 0$
    end if
    $i = i + 1$
    $Mood = rand(1, 4)$
    if ($Mood == 1$) then
        Create new views based on Eq. (7)
    else if ($Mood == 2$) then
        Create new views based on Eq. (8)
    else if ($Mood == 3$) then
        Create new views based on Eq. (9)
    else if ($Mood == 4$) then
        Create new views based on Eq. (10)
    end if
    Limit new views According to Eq. (13)
    Evaluate new view based on the objective function
    if (New view better than current view) then
        Replace old view with new view and share it
    else
        Keep old view, don't share new view
    end if
    Calculate $D^P$
    if ($D^P < D_t$) then
        Replace worst $nrs$ with solutions created as in (14)
    end if
    Asses the population
    Find the current best
    Update $D_t$ by expression (20)
while ($t \leq T$)
Return Optimal Solution
Return Overall and detailed optimization statistics and visualize
```

compared to other approaches for this challenge it obtains better performance in terms of classification accuracy and smaller chosen subset of relevant features [45]. In this particular case, the calculation of objective function for each metaheuristics individual involves training and testing ML classifier based on the selected features subset.

The framework is developed in Python using machine data science and ML libraries: numpy, scipy, pandas, matplotlib, seaborn and scikit-learn. The ELM was implemented from the scratch as custom library, since it is not available in scikit-learn.

Metaheuristics incorporated into the framework are encoded using flat encoding scheme (vector), and the length of the solution depends of the level of the framework. In L1 of the framework every individual in the population is represented as a vector with length $L = nf$, where nf represents the total number of features in the data set.

For framework L2, one individual represents the ELM's hyper-parameter nn and weights and biases values between the input features and neurons in the hidden layer. The length of the solutions depends on the tuned nn and number of selected features (nsf) from L1. Therefore, the L2 solutions' length is given as: $L = 1 + nn \cdot nsf + nn$. The first parameter represents the number of neurons, which is simple scalar integer value, $nn \cdot nsf$ are continuous parameters that encode weights values, while the nn components, which are also of a real data type, denote hidden layer biases. It is noted that all metaheuristics solutions in the L2 are of variable lengths (its length changes over iterations), because they dependent on the determined nn.

In the L1, the search is performed in the binary search space, while in the L2, the solutions' vectors consist of integer (nn) and real parameters (weights and biases). Therefore, as the result, binary and continuous variants of DOSNS and other metaheuristics incorporated into the framework have been utilized. For feature selection task, in L1 of the framework, to map the continuous search region to binary, it is necessary to use a transfer function, that has been selected empirically, through numerous simulations with sigmoid, threshold and hyperbolic tangent function transfer functions that are common choice in recent literature [81, 116–119]. After executing experiments with a variety of transfer functions, tanh yielded the best result, and it was chosen and used in this research.

Discrete variable, that encode the nn, have been rounded up to the nearest integer value, because of the large search domain. The search equation has not been modified to match the discrete domain, as the empirical simulations indicated that there is no notable enhancement, and the simple rounding up is preserving the most of the resources.

Therefore, in the L1 of the framework, a binary tanh-based binary versions of metaheuristics are employed, e.g., bDOSNS, while in the L2 standard metaheuristics versions for real-parameters optimization are used, e.g., DOSNS.

For the L1 part of the framework, k-nearest neighbor (KNN) is chosen as the classifier with its default parameters from scikit-learn library. According to previous research [118, 120], KNN is the most common choice for feature selection tasks, because it is computationally efficient. Likewise, the most commonly used objective function for feature selection challenges [117, 118], that takes into account the KNN's classification error rate (err) and the number of chosen features (nsf), is used in this research:

$$F = \alpha \cdot err + \beta \cdot \frac{nsf}{nf}, \tag{21}$$

where α and β are weight coefficients for determining the relative importance of err to nsf, and $\beta = 1 - \alpha$.

The objective function for the L2 framework is simply the classification error rate. Since some of the data sets are only slightly imbalanced, the error rate is viable fitness function.

Block diagram of proposed framework in cooperative mode is shown in Fig. 1. Any number of metaheuristics can be integrated into the framework; however, on the presented figure flow-chart of proposed DOSNS is shown, where other metaheuristics are represented with the general flow-chart.

Experimental setup and results

This section first describes the data sets utilized in the experiments together with the overview of experimental setup. Later on, the outcomes of the executed experiments along with the immense comparative analysis, results discussion and statistical tests validation, are delivered.

Data sets

The experiments were conducted over three different publicly available benchmark phishing data sets. Phishing is a type of cyber-attack where malicious user masquerades as known and trusted entity, and tries to obtain sensitive data typically through a false website that tricks the end user to enter private information such as credit card number or similar. After obtaining this data, the attacker can utilize it to access the bank accounts, steal money, data or identity, and place malware to the target's computer.

First data set is named Phising websites Kaggle[1] [121], and it is publicly available from Kaggle repository. It is comprised of 48 features, derived from 5000 phishing and 5000 legit websites, collected in years 2015 and 2017. This data set is balanced, and represents a binary classification task. The class distribution and boxplot of this data set are shown in Fig. 2, while the the heat map is provided in Fig. 3.

Second data set is named phishing websites UCI data set,[2] proposed by [122–124], and it is publicly available on UCI Machine Learning Repository [125]. Although it is stated on the UCI repository that this data set is comprised of 2456 instances, it actually contains 11,056 instances, with 30 features. This data set is slightly imbalanced, and represents a binary classification task. The phishing websites UCI visual representation in a form of bar charts and box and whiskers diagrams are shown in Fig. 4, while the the heat map is provided in Fig. 5.

Finally, third data set is named phishing websites UCI small data set,[3]. introduced by [126], and it is also publicly available on UCI Machine Learning Repository [125]. It contains total of 1,353 instances, and although it is stated that it has 10 features, it actually contains 9 features with one target variable. It represents a multi-class classification task, as it contains three classes (0—normal, 1—suspicious, 2—phishing). This data set is also slightly imbalanced. The class distribution and boxplot of this data set are shown in Fig. 6, while the the heat map is provided in Fig. 7.

Experimental setup

The performance of introduced DOSNS algorithm with respect to the converging speed and overall capabilities has been tested for feature selection and ELM optimization tasks on the above mentioned data sets. The experimental outcomes have been validated and compared to the performances of several state-of-the-art reference metaheuristics, that have been tested under the same experimental setup and simulation conditions. All metaheuristics are integrated into the devised two-level framework described in Sect. 3.3.

The set of cutting-edge metaheuristics included in comparative analysis encompasses: basic SNS [41], firefly algorithm (FA) [72], bat algorithm (BA) [127], artificial bee colony (ABC) [61], harris hawks optimization (HHO) [128] and sine cosine algorithm (SCA) [79]. This particular set of algorithms is used for comparison, because they all exhibit different exploitation and exploration abilities and in this way it was feasible to perform a robust performance validation of proposed DOSNS approach. All reference algorithms have been implemented independently in Python for the sake of this research, with the proposed control parameters values as described in their respective publications. In the sections with the experimental results, for easier understanding, for all metaheuristics the prefix FS was used for feature selection results, e.g., FS–DOSNS, FS–SNS, FS–BA, etc., while prefix ELM was utilized for ELM tuning simulations, e.g., ELM–DOSNS, ELM–SNS, ELM–BA, etc.

Prior to execution, all data sets were divided into the train and test subsets using 80–20% split rule, as it is shown in Fig. 1. Since the ELM does not employ traditional training, validation data is not used.

All experiments, for both L1 and L2 parts of the framework, have been executed by utilizing the population size of 6 ($NP = 6$), maximum number of iterations was set to 10 ($T = 10$) in one run, with the total number of 30 independent runs ($runtime = 30$).

The DOSNS specific control parameter nrs is set to 1 because of realtively small population size, while the D_{t0} and D_t are set and updated according to Eqs. (19) and (20), respectively

The KNN used as the classifier in the L1 is set with five neighbor solutions $k = 5$, while α in objective function (Eq. 21) is set to 0.99. The lower and upper bounds for both

[1] https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning.

[2] http://archive.ics.uci.edu/ml/datasets/phishing+websites.

[3] https://archive.ics.uci.edu/ml/datasets/Website+Phishing.

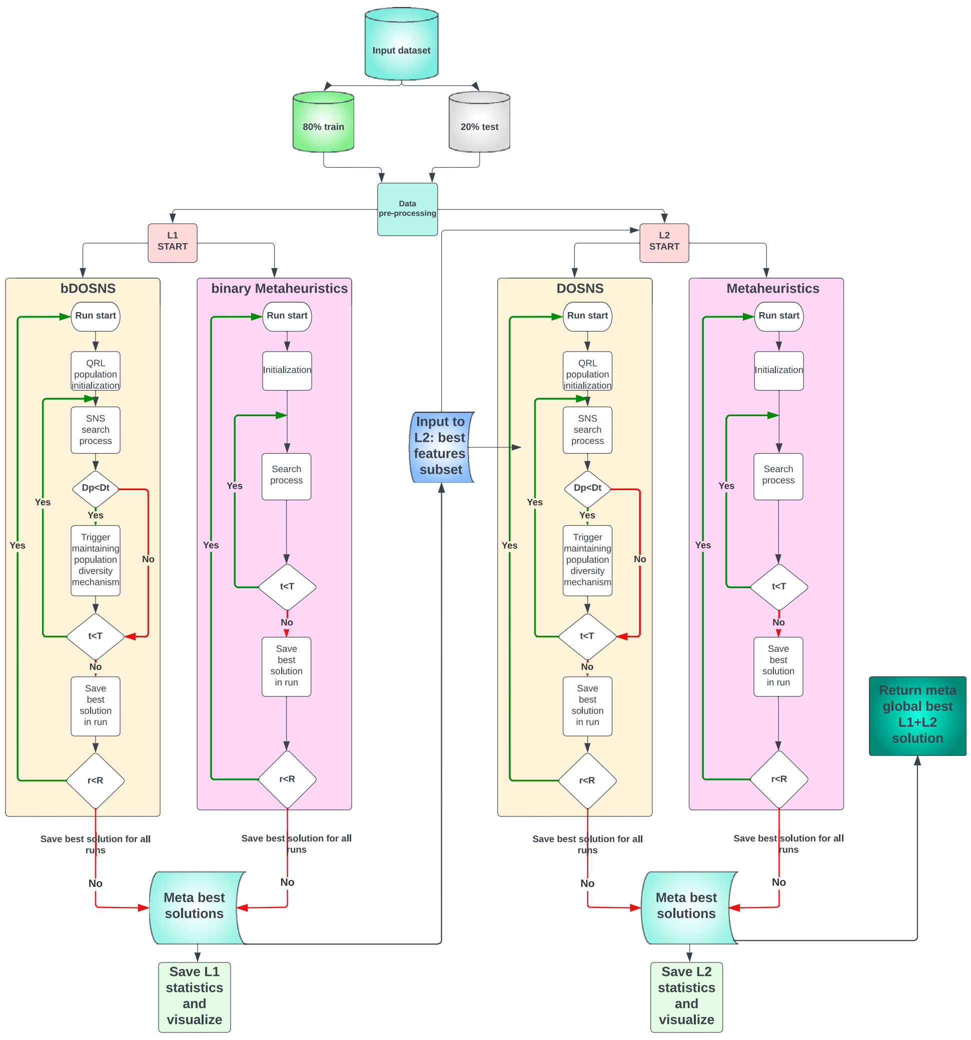

Input dataset
80% train
20% test
Data pre-processing
L1 START
L2 START
bDOSNS
binary Metaheuristics
DOSNS
Metaheuristics
Run start
QRL population initialization
Initialization
SNS search process
Search process
Dp<Dt
Trigger maintaining population diversity mechanism
t<T
Save best solution in run
r<R
Yes
No
Save best solution for all runs
Input to L2: best features subset
Meta best solutions
Save L1 statistics and visualize
Save L2 statistics and visualize
Return meta global best L1+L2 solution

Fig. 1 Block diagram of devised two-layer feature selection and ELM tuning framework in cooperative mode with flow-chart of proposed DOSNS metaheuristics

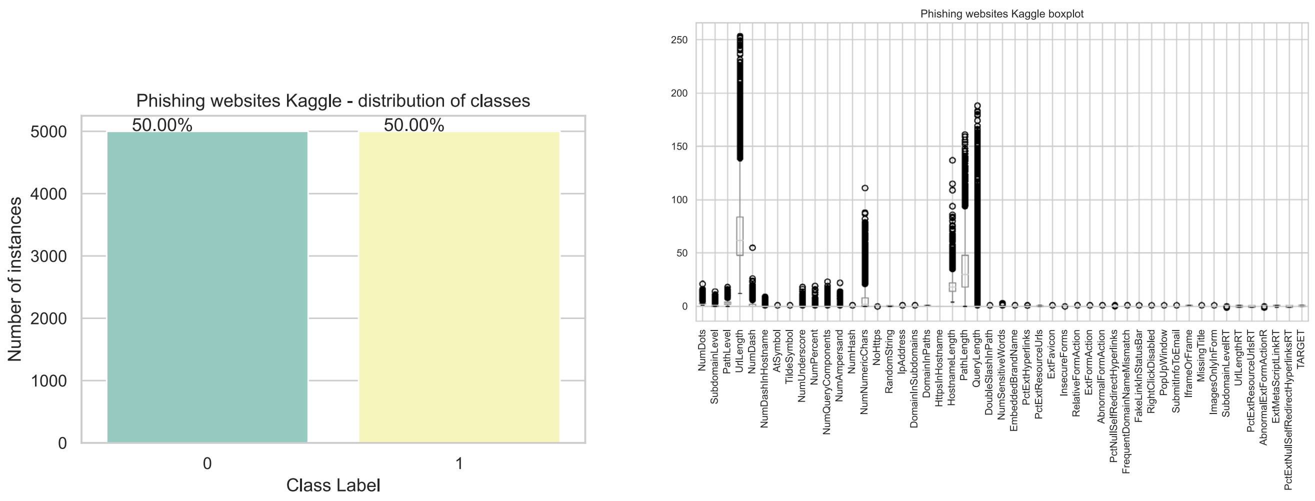


Fig. 2 Phishing websites Kaggle data set class distribution and boxplot

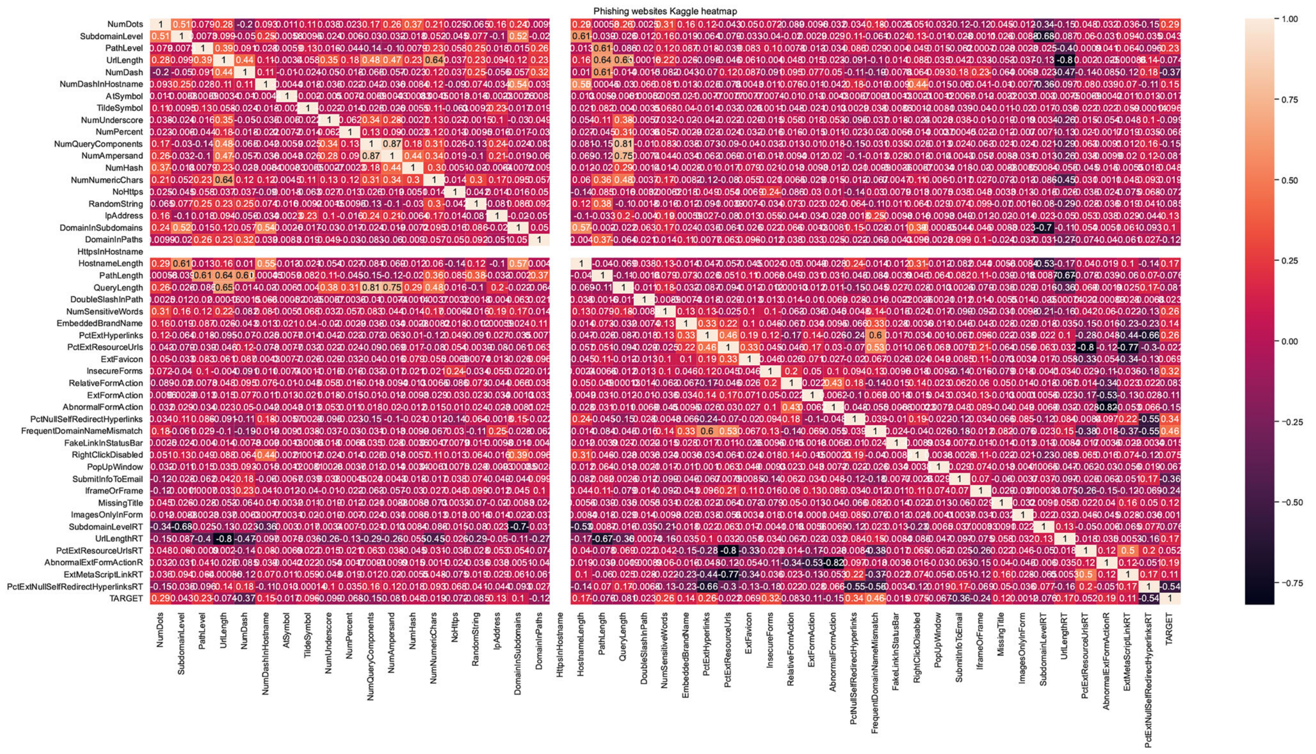
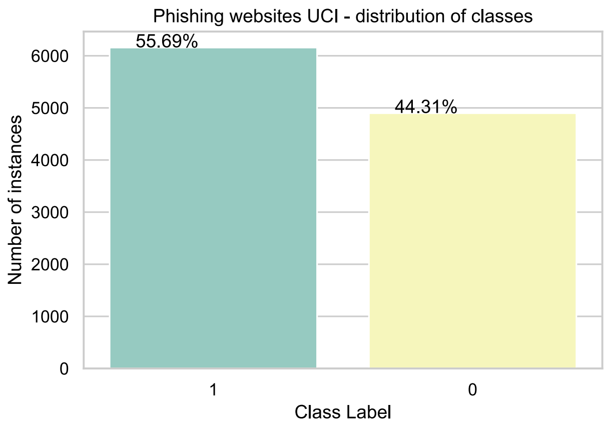


Fig. 3 Phishing websites Kaggle data set heat map



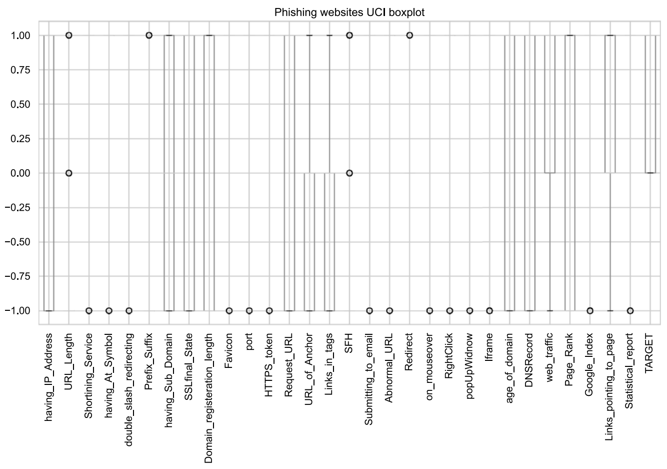


Fig. 4 Phishing websites UCI data set class distribution and boxplot

weight and bias variables in the L2 for ELM are defined by the interval $[-1, 1]$. The search space limits for the *nn* variable are specified in the range $[nsf \cdot 3, nsf \cdot 15]$. All these values are determined empirically. It is noted that prior to rendering final decision regarding the L1 classifier, experiments with support vector machine (SVM) were also executed and similar results are obtained. However, due to the faster execution, the KNN is utilized instead of SVM.

Experimental findings

Two sets of experiments were executed in this research. First experiment utilizes both L1 and L2 of the proposed optimization framework. The L1 is set to cooperative mode and in this way, the best obtained set of features for all metaheuristics is used as the input to L2, i.e., in the L2 part of the framework, all methods tune ELM with the same set of input features.

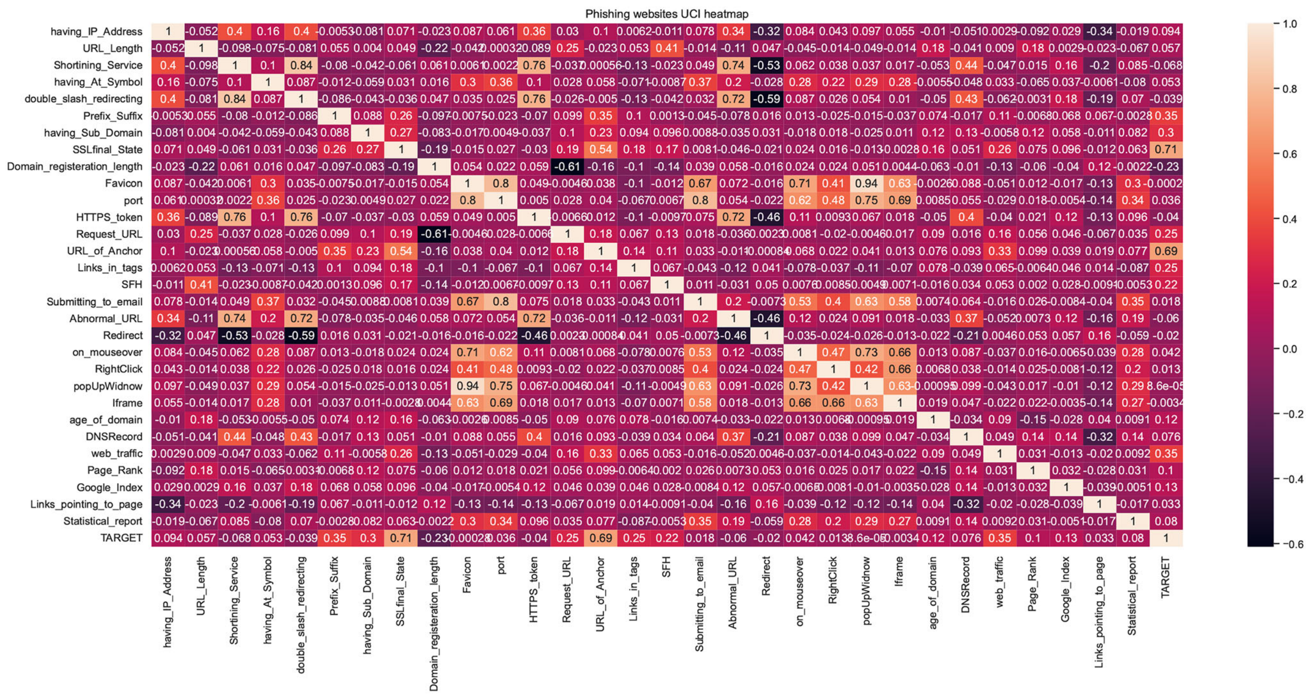
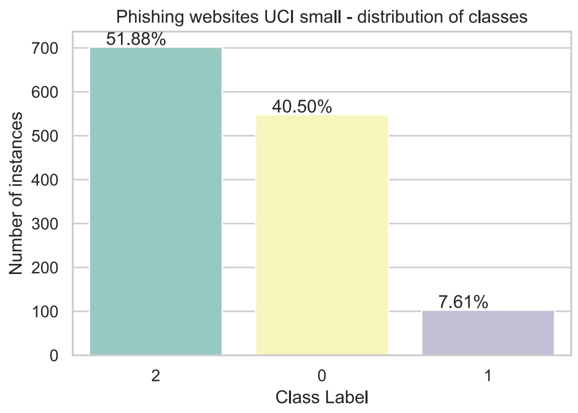


Fig. 5 Phishing websites UCI data set heat map



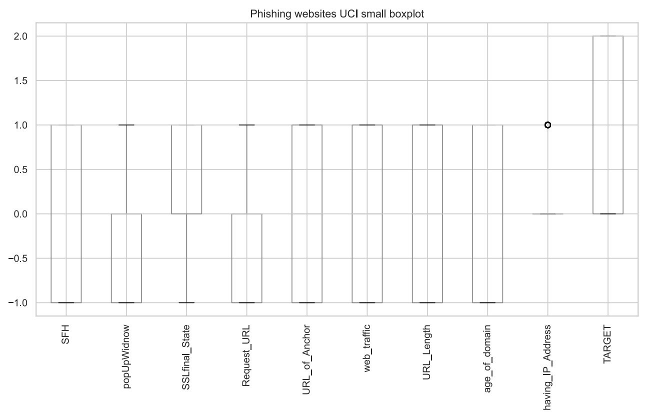


Fig. 6 Phishing websites UCI small data set class distribution and boxplot

Second simulation is focused only on tuning the ELM with all employed features (without the feature selection), using only L2 part of the framework. The goal of this simulation is to test performance of suggested DOSNS for large-scale global optimization tasks with many parameters (components).

For both experiments, two different group of metrics were presented. First, overall metrics, that summarize average values obtained over 30 runs, include best, worst, mean, median, standard deviation for objective in L1 and classification error in L2. In addition, number of selected features for best obtained objective in L1, and the number of neurons for the best performing ELM in L2 are also shown.

Detailed performance indicators were also provided for simulations with L2 part of the framework and this set include the following metrics for the best generated solution of each metaheuristics: accuracy, precision, recall and f1 score per each class and micro-averaged over all classes. Metrics for receiver operating characteristic (ROC) and precision recall (PR) area under the curve (AUC) are shown visually.

Findings for both experiments are shown in Sects. 4.3.1 and 4.3.2 and best obtained results for each category of metrics is marked with bold style in the tables with reported results.

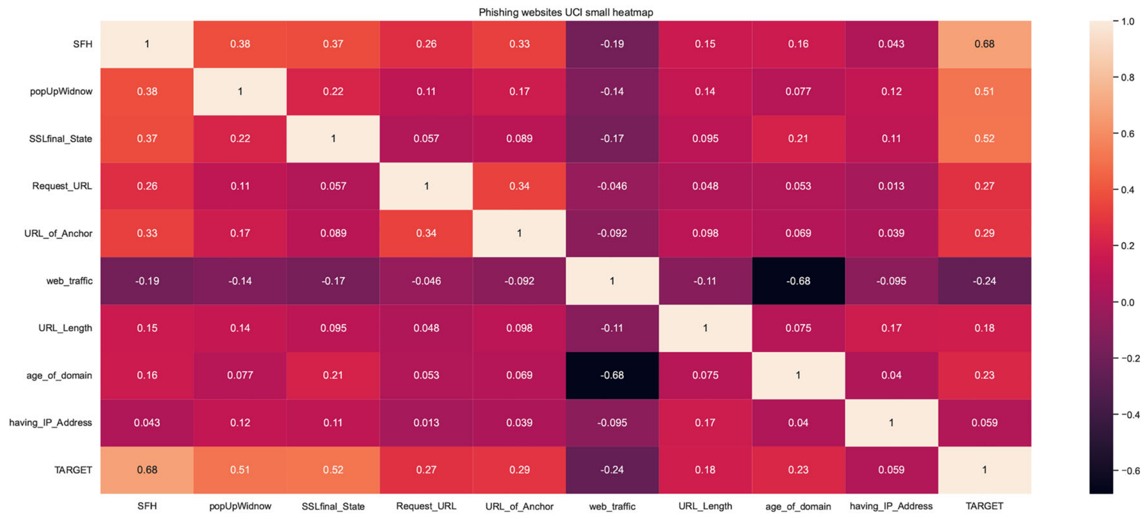


Fig. 7 Phishing websites UCI small data set heat map

Table 1 Experiment I—overall metrics for L1 results (feature selection) with respect to objective function

Method	FS–DOSNS	FS–SNS	FS–FA	FS–BA	FS–ABC	FS–HHO	FS–SCA
Kaggle dataset							
Best	**0.038530**	0.040980	0.045330	0.051478	0.050175	0.050045	0.040302
Worst	**0.048977**	0.066615	0.061092	0.073180	0.064868	0.078885	0.074822
Mean	**0.045316**	0.050902	0.051420	0.062082	0.055203	0.061156	0.051322
Median	**0.046358**	0.049473	0.050593	0.061300	0.054722	0.058655	0.049616
Std	**0.003067**	0.007689	0.004276	0.006315	0.004063	0.010100	0.010862
Var	**0.000009**	0.000059	0.000018	0.000040	0.000017	0.000102	0.000118
Best error	**0.034500**	0.039500	0.042000	0.048000	0.045000	0.045500	0.036500
Best no. feat	21	9	18	19	27	24	20
UCI dataset							
Best	**0.056274**	0.057836	0.057388	0.057617	0.056597	0.068020	0.062104
Worst	**0.067458**	0.080109	0.075413	0.108204	0.076537	0.109547	0.072622
Mean	**0.062618**	0.071968	0.067185	0.076603	0.068676	0.081738	0.067400
Median	**0.063261**	0.071721	0.067348	0.074012	0.068015	0.079142	0.067239
Std	0.003676	0.006546	0.005327	0.016053	0.006475	0.012600	**0.002857**
Var	0.000014	0.000043	0.000028	0.000258	0.000042	0.000159	**0.000008**
Best error	0.052465	0.053370	0.052917	0.053822	**0.051108**	0.063320	0.059701
Best no. feat	13	15	15	13	18	16	9
UCI small dataset							
Best	0.099106	0.116261	0.099106	0.119914	0.117372	0.135638	0.114830
Worst	**0.127220**	0.192185	0.163752	0.287167	0.190754	0.212673	0.191866
Mean	**0.112568**	0.148941	0.137782	0.179993	0.154480	0.176994	0.147670
Median	**0.116261**	0.154063	0.146597	0.165023	0.164467	0.172329	0.144930
Std	**0.011258**	0.027886	0.021520	0.047198	0.027404	0.029065	0.029909
Var	**0.000127**	0.000778	0.000463	0.002228	0.000751	0.000845	0.000895
Best error	0.092251	0.110701	0.092251	0.114391	0.110701	0.129151	0.107011
Best no. feat	7	6	7	6	7	7	8

Simulation I—L1 and L2 (feature selection and ELM tuning)

This section shows the result of the first experiment, that utilized both L1 and L2 of the proposed cooperative framework. The best obtained subset of features at L1 that performs the feature selection is used as the input to the L2 that conducts the ELM tuning process.

Table 1 presents the overall metrics for L1 results (feature selection task) in terms of objective function, defined in Eq. (21). In addition, best rendered classification error for the best objective is also presented.

From Table 1, it is clear that the proposed FS–DOSNS method obtains superior results on Kaggle and UCI data sets, in terms of best, worst, mean and median values. In the case of UCI small data set, FS–DOSNS shares the first place with FS–FA for the best result, while other metrics are clearly in favor of the proposed FS–DOSNS.

As the conclusion for L1 simulations, the FS–DOSNS establishes the best balance between the number of features (complexity) and classification performance (error).

Table 2 presents the overall metrics for ELM tuning (L2 of the framework), where the best obtained subset of features (from all algorithms in L1) was used as the input. The proposed FS–DOSNS achieved the best accuracy on the Kaggle data set, and shares the first place on UCI data set (with FS–SCA) and UCI small data set (again with FS–SCA). Concerning other metrics, the best worst and mean results on Kaggle data set were obtained by FS–SCA, while FS–HHO obtained the best median and standard deviation. In case of UCI data set, the best worst and mean results were achieved by the proposed FS–DOSNS approach, FS–SCA obtained the best median result, while FS–ABC obtained the best standard deviation. Finally, on the UCI small data set, the best worst value was achieved with the proposed FS–DOSNS and FS–SCA approaches that shared the first place. The proposed FS–DOSNS also obtained the best mean and median values, while the FS–ABC obtained the best standard deviation.

Table 3 shows the detailed metrics on the Kaggle, UCI, and UCI small data sets, for the best solutions after the execution of the complete framework (L1 + L2). In case of Kaggle data set, the proposed ELM–DOSNS approach obtained the best accuracy of 95.65%, while the ELM–FA approach finished second with 95.50%. For the UCI data set, the proposed ELM–DOSNS method obtained the best accuracy of 93.94%, together with the ELM–SCA approach that obtained the same accuracy. Finally, the detailed metrics on the UCI small data set, show that the proposed ELM–DOSNS approach share the first place with ELM–SCA, where both methods accomplished an accuracy of 90.78%. All other performance metrics included in Table 3, precision, recall and f1 score, on average prove superiority of ELM–DOSNS.

Convergence graphs of the objective function for the L1, and error for the L2 experiments, for all observed metaheuristics and all three utilized data sets, are shown in Fig. 8. Box and whiskers diagrams, that display the dispersion of the objective function for L1, and error for L2 framework simulations, over 30 independent runs for all observed methods are depicted in Fig. 9.

Confusion matrices for output of framework L1 with feature selection, for all observed algorithms and all three data sets, are shown in Fig. 10, while Fig. 11 shows the ROC AUC curves with micro and macro averages for the best solutions (output) of the framework L2, for all observed algorithms over all three utilized data sets.

Simulation II—L1 (ELM tuning)

In the second set of the experiments, all algorithms were tested for the task of ELM tuning, without feature selection (meaning that all features were used by the models), and in this case only L2 of the framework was utilized. Table 4 shows the overall metrics for this scenario, and it can be noticed that the proposed ELM–DOSNS obtained superior results on all three used data sets, by achieving the first place in terms of best, worst, mean and median results. It is also noticed that the ELM–DOSNS managed to establish good results' quality with relatively simple ELM (small number of neurons in the hidden layer).

Table 5 shows the detailed metrics on the Kaggle, UCI, and UCI small data sets, for the best solutions without feature selection. In case of Kaggle data set, the proposed ELM–DOSNS approach obtained the best accuracy of 97.25%, in front of the ELM–FA and ELM–BA approaches that obtained the accuracy of 97.00%. Similarly, in case of UCI data set, the proposed ELM–DOSNS approach again obtained the best accuracy of 94.75%, in front of the ELM–SCA that obtained the accuracy of 94.57%. Finally, also for the UCI small data set, proposed ELM–DOSNS obtained the best accuracy of 90.04 %, followed by the ELM–SCA and ELM–FA that managed to establish an accuracy of 89.93%. Similarly as in the previous experiment, all other metrics are on average in favor to ELM–DOSNS.

Convergence graphs and box plots of the error (as this experiment is executed with just L2 framework without feature selection), for all observed methods and all three data sets are given in Fig. 12.

The PR AUC curves for the best generated solution of all metaheuristics, where only ELM hyper-parameters tuning was performed without feature selection, are shown in Fig. 13. In addition, to visualize performance of the classifier with more details, one vs rest (OvR) ROC curves are visualized in Fig. 14, for all observed methods and all three data sets, in case of ELM tuning with all features (no feature selection). The relation of each class to other classes, together with their distribution, can be seen on histograms that are also provided in Fig. 14.

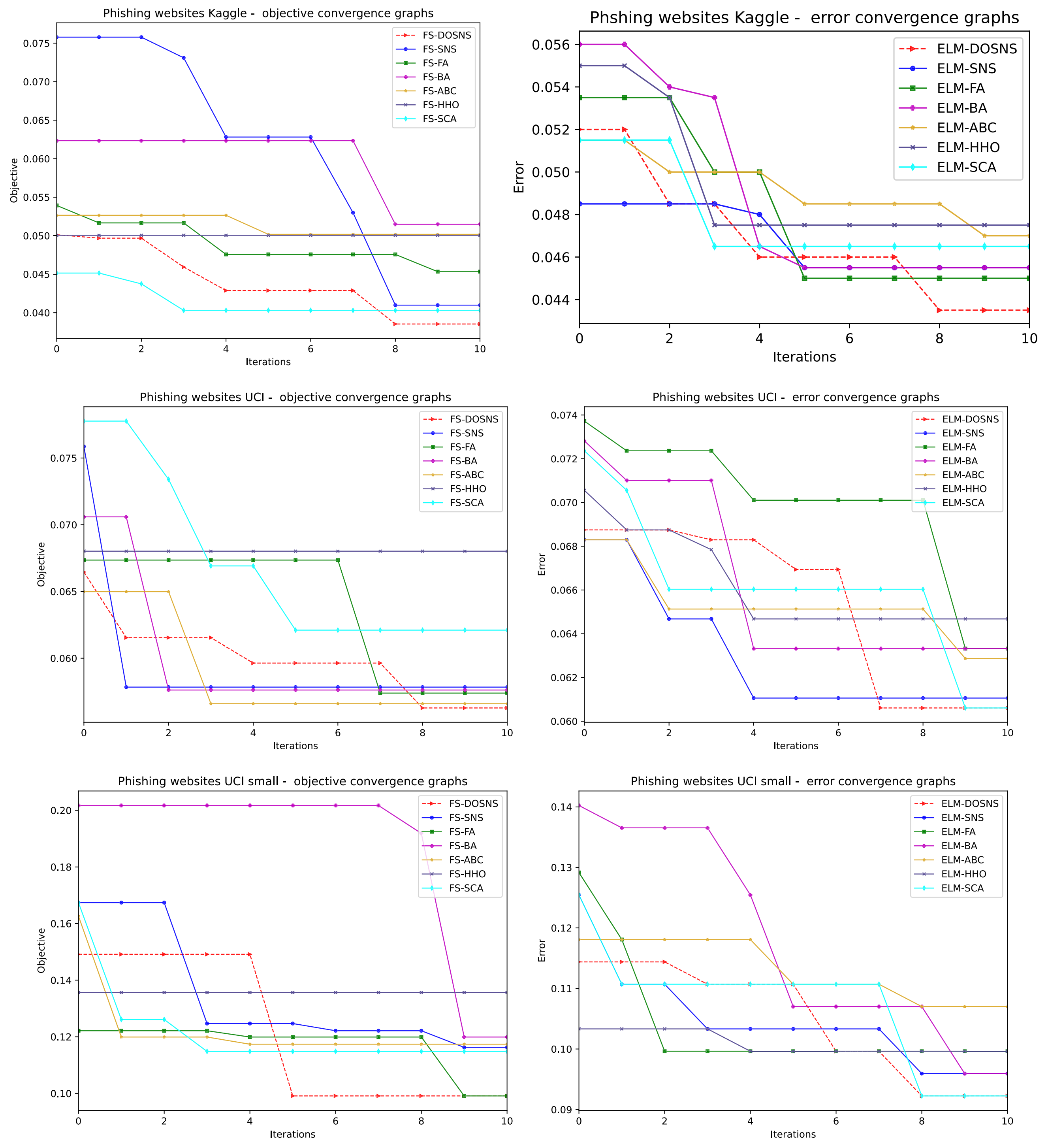


Fig. 8 Experiment I—convergence graphs of objective function for L1 simulations, and error for L2 experiments, for all observed methods and all three data sets

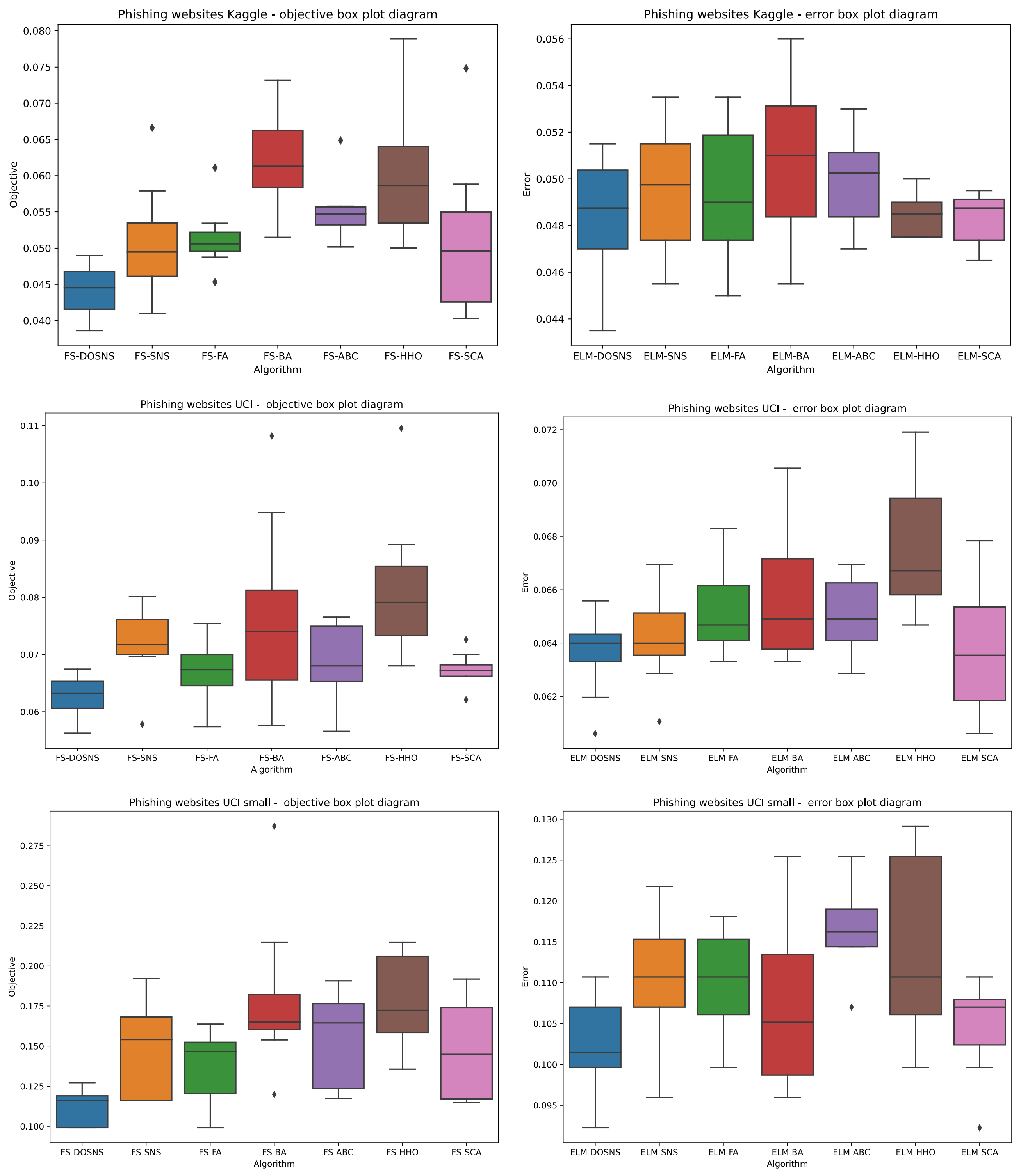


Fig. 9 Experiment I—box and whiskers diagrams of objective function for L1 simulations, and error for L2 experiments, for all observed methods and all three data sets

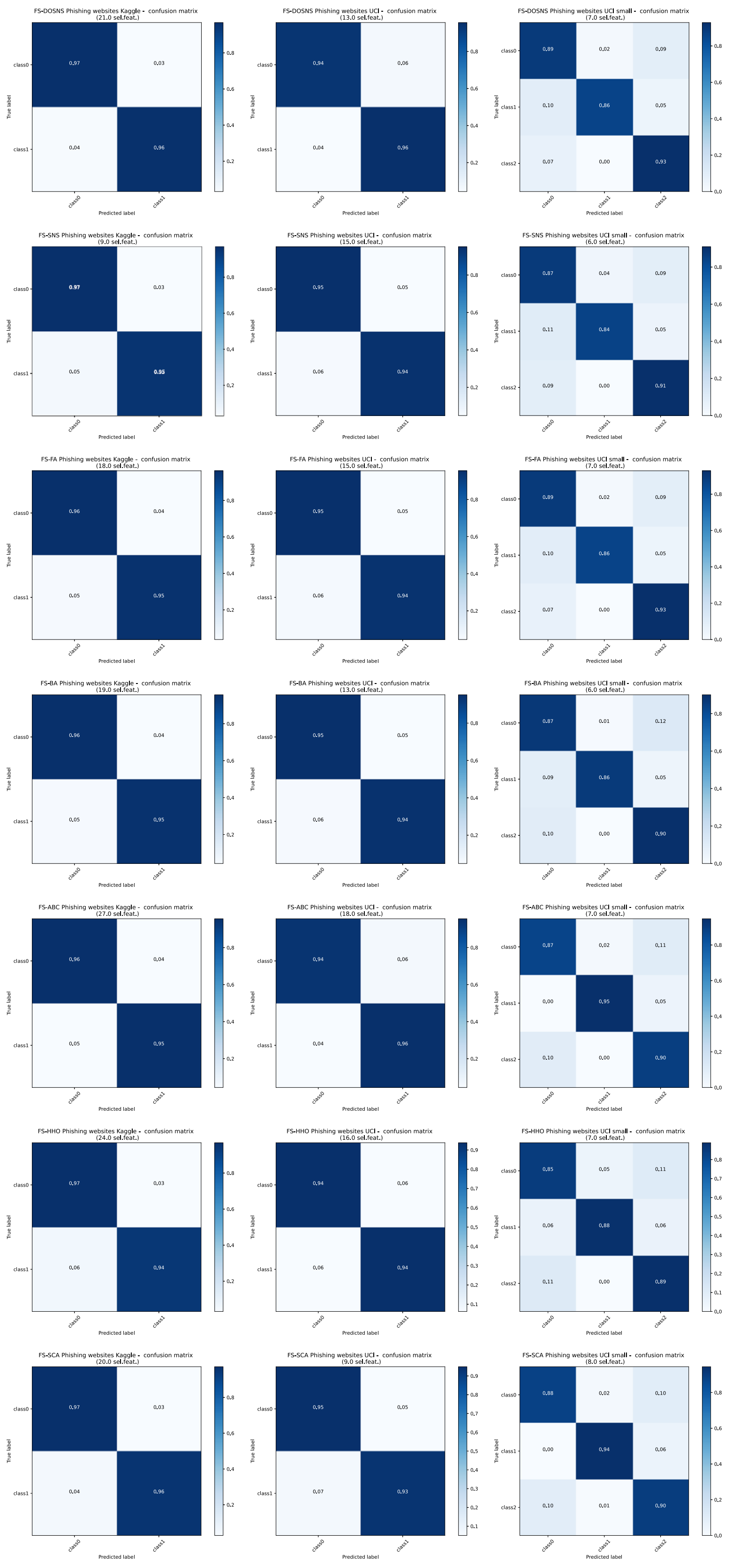


Fig. 10 Experiment I—confusion matrices for L1 output for feature selection, for all observed methods and all three data sets

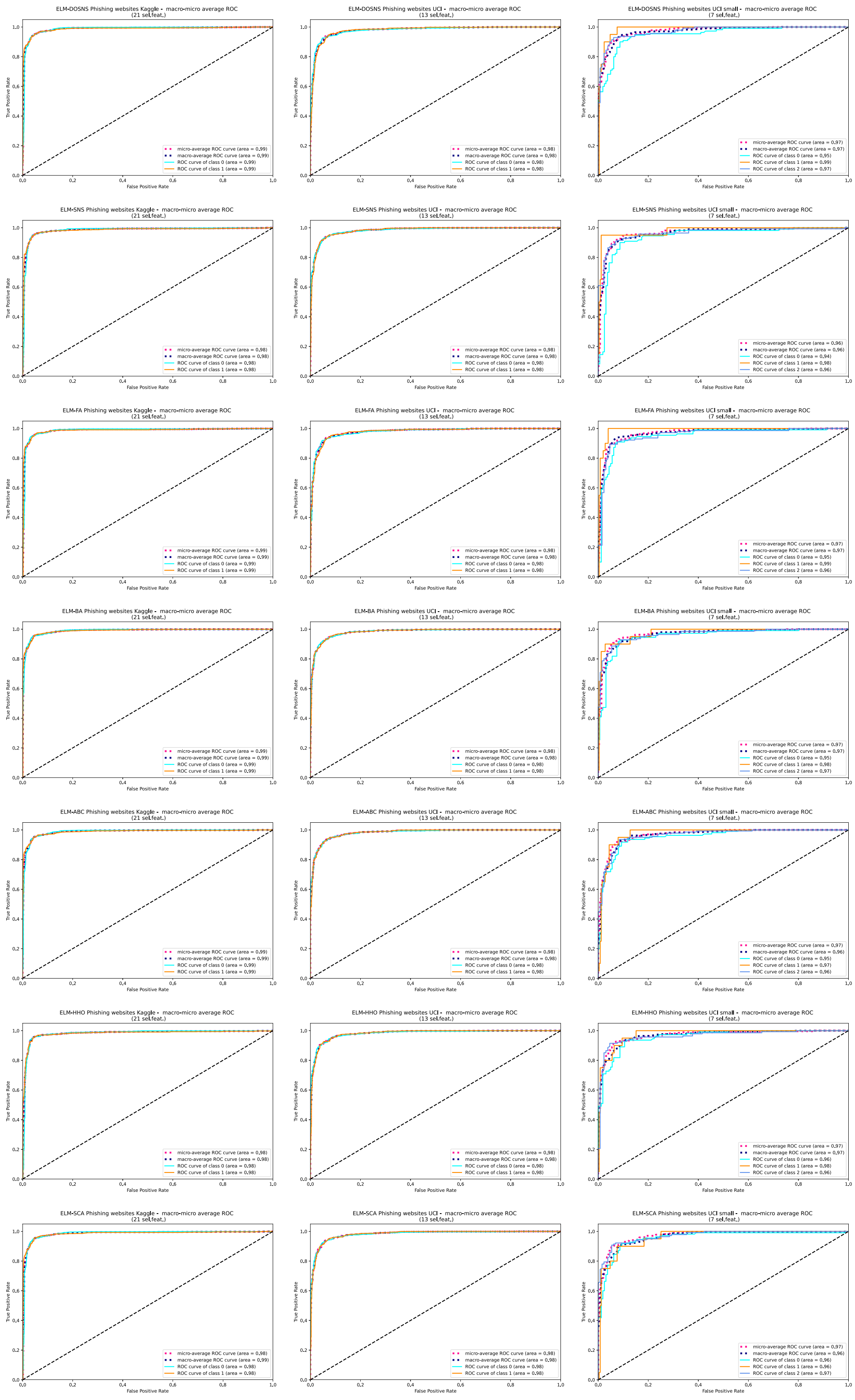


Fig. 11 Experiment I—ROC AUC with micro and macro averages for the best solutions (output) of L2 for all observed methods and all three data sets

Table 2 Experiment I—overall metrics for L2 results (ELM tuning) with respect to classification error

Method	FS–DOSNS	FS–SNS	FS–FA	FS–BA	FS–ABC	FS–HHO	FS–SCA
Kaggle							
Best	**0.043500**	0.045500	0.045000	0.045500	0.047000	0.047500	0.046500
Worst	0.051500	0.053500	0.053500	0.056000	0.053000	0.050000	**0.049500**
Mean	0.048500	0.049375	0.049438	0.050938	0.049938	0.048438	**0.048313**
Median	0.048750	0.049750	0.049000	0.051000	0.050250	**0.048500**	0.048750
Std	0.002500	0.002701	0.002822	0.003340	0.001861	**0.000845**	0.001088
Var	0.000006	0.000007	0.000008	0.000011	0.000003	0.000001	0.000001
Best no. feat	21	21	21	21	21	21	21
Best no. neurons	313	315	315	291	298	313	315
UCI							
Best	0.060606	0.061058	0.063320	0.063320	0.062867	0.064677	0.060606
Worst	**0.065581**	0.066938	0.068295	0.070556	0.066938	0.071913	0.067843
Mean	**0.063602**	0.064224	0.065185	0.065751	0.065072	0.067673	0.063885
Median	0.063998	0.063998	0.064677	0.064903	0.064903	0.066712	**0.063546**
Std	0.001482	0.001766	0.001541	0.002349	**0.001384**	0.002588	0.002475
Var	0.000002	0.000003	0.000002	0.000006	0.000002	0.000007	0.000006
Best no. feat	13	13	13	13	13	13	13
Best no. neurons	195	188	118	181	181	189	195
UCI small							
Best	0.092251	0.095941	0.099631	0.095941	0.107011	0.099631	0.092251
Worst	0.110701	0.121771	0.118081	0.125461	0.125461	0.129151	0.110701
Mean	**0.102399**	0.110701	0.110240	0.107472	0.116697	0.113930	0.104705
Median	**0.101476**	0.110701	0.110701	0.105166	0.116236	0.110701	0.107011
Std	0.005458	0.007380	0.006240	0.010508	**0.005198**	0.010508	0.005816
Var	0.000030	0.000054	0.000039	0.000110	**0.000027**	0.000110	0.000034
Best no. feat	7	7	7	7	7	7	7
Best no. neurons	105	105	94	105	105	104	105

The visualization of results clearly indicates the superior performance of the proposed ELM–DOSNS method. However, it is required to execute additional statistical tests to prove that the results are statistically significantly better than the results obtained by other considered approaches.

It is also interesting to compare results from Table 3, that shows detailed findings with feature selection employed, to the results from Table 5, where all features have been utilized. From this side-by-side comparison, it can be seen that in majority of cases the obtained accuracy is better when all features are used (no feature selection), however, with drastically higher computational costs. Therefore, it is justified to perform the feature selection and reduce the computational complexity. For example, the proposed ELM–DOSNS achieves the best accuracy of 95.65% on the Kaggle data set, 93.94% on UCI data set, and finally, 90.78% on UCI small data set when the feature selection is utilized. The same approach achieves accuracy of 97.25%, 94.75%, and 90.04%, respectively, with all features used (no feature selection). Therefore, the same approach delivers better accuracy on the Kaggle and UCI data sets without feature selection (second experiment), while on the UCI small data set the accuracy is better when feature selection is used (first experiment). The same conclusion applies for all utilized approaches on these three particular data sets. From these findings, a conclusion that the UCI small data set contains noisy features, which disrupt classification, can be derived.

Findings validation and best models interpretation

Reported findings from experiments I and II, showed in Sects. 4.3.1 and 4.3.2, respectively, prove that in average proposed DOSNS showed better results' quality and convergence speed than other opponent cutting-edge metaheuristics. However, the experimental results are not sufficient to determine if one algorithm has significantly better performance compared to the others and there is an urge to conduct statistical tests.

Table 3 Experiment I—detailed metrics for the best solutions with feature selection and ELM tuning (L1+L2)

	ELM–DOSNS	ELM–SNS	ELM–FA	ELM–BA	ELM–ABC	ELM–HHO	ELM–SCA
Kaggle dataset							
Accuracy (%)	**95.65**	95.45	95.50	95.45	95.30	95.25	95.35
Precision 0	0.957874	0.959555	0.959596	**0.961421**	0.953000	0.953862	0.952144
Precision 1	**0.955135**	0.949555	0.950495	0.947783	0.953000	0.951147	0.954865
M.Avg. Precision	**0.956504**	0.954555	0.955046	0.954602	0.953000	0.952504	0.953504
Recall 0	0.955000	0.949000	0.950000	0.947000	0.953000	0.951000	0.955000
Recall 1	0.958000	0.960000	0.960000	**0.962000**	0.953000	0.954000	0.952000
M.Avg. Recall	**0.956500**	0.954500	0.955000	0.954500	0.953000	0.952500	0.953500
F1 Score 0	**0.956435**	0.954248	0.954774	0.954156	0.953000	0.952429	0.953570
F1 Score 1	**0.956565**	0.954749	0.955224	0.954839	0.953000	0.952571	0.953430
M.Avg. F1 Score	**0.956500**	0.954499	0.954999	0.954497	0.953000	0.952500	0.953500
UCI dataset							
Accuracy (%)	93.94	93.89	93.67	93.67	93.71	93.53	93.94
Precision 0	0.936017	0.928862	0.940189	**0.944857**	0.934783	0.938155	0.939646
Precision 1	0.942029	**0.947025**	0.934022	0.930599	0.938956	0.933174	0.939200
M.Avg. Precision	0.939367	0.938983	0.936753	0.936912	0.937108	0.935380	**0.939398**
Recall 0	0.926456	**0.933606**	0.915220	0.910112	0.922370	0.914198	0.922370
Recall 1	0.949675	0.943182	0.953734	**0.957792**	0.948864	0.952110	0.952922
M.Avg. Recall	0.939394	0.938942	0.936680	0.936680	0.937133	0.935323	0.939394
F1 Score 0	0.931211	**0.931228**	0.927536	0.927159	0.928535	0.926022	0.930928
F1 Score 1	0.945837	0.945100	0.943775	0.944000	0.943884	0.942547	**0.946011**
M.Avg. F1 Score	**0.939361**	0.938957	0.936585	0.936543	0.937087	0.935230	0.939333
UCI small dataset							
Accuracy (%)	90.77	90.41	90.04	90.41	89.3	90.04	90.77
Precision 0	0.877193	0.883929	0.877193	0.883929	0.873874	0.869565	**0.885965**
Precision 1	**0.928571**	0.875000	0.823529	0.833333	0.812500	0.866667	0.812500
Precision 2	0.930070	0.923077	0.928571	0.929078	0.916667	0.929078	**0.936170**
M.Avg. Precision	**0.908496**	0.903638	0.899965	0.903686	0.891609	0.900316	0.906665
Recall 0	0.909091	0.900000	0.909091	0.900000	0.881818	0.909091	**0.918182**
Recall 1	0.650000	0.700000	0.700000	**0.750000**	0.650000	0.650000	0.650000
Recall 2	**0.943262**	0.936170	0.921986	0.929078	0.936170	0.929078	0.936170
M.Avg. Recall	0.907749	0.904059	0.900369	0.904059	0.892989	0.900369	0.907749
F1 Score 0	0.892857	0.891892	0.892857	0.891892	0.877828	0.888889	**0.901786**
F1 Score 1	0.764706	**0.777778**	0.756757	0.789474	0.722222	0.742857	0.722222
F1 Score 2	**0.936620**	0.929577	0.925267	0.929078	0.926316	0.929078	0.936170
M.Avg. F1 Score	0.906169	0.903078	0.899675	0.903681	0.891572	0.899022	**0.906424**

Various statistical tests are available to establish whether or not rendered improvements by referenced approach are statistically significant. In this paper, 7 methods (including proposed DOSNS) were compared with respect to measure taken as fitness (objective function and error in case of L1 and L2 tests, respectively), which falls into the domain of multiple-approaches multi-problem analysis [129].

Following related literature recommendations [129–131], to conduct statistical tests, a results sample for each approach is constructed by taking average values of measured objectives over multiple independent runs for each problem. The downside of this approach can be observed in cases when the measured variable has outliers, not following a normal distribution and in such scenarios, misleading results can be generated. Whether the average objective function value should be taken for the purpose of statistical tests when comparing stochastic methods still remains an open question [129].

Therefore, to check whether or not it is safe to use the mean objective value as the base for statistical tests, Shapiro–

Table 4 Experiment II—overall metrics for L2 results (ELM tuning) without feature selection

Method	ELM–DOSNS	ELM–SNS	ELM–FA	ELM–BA	ELM–ABC	ELM–HHO	ELM–SCA
Kaggle dataset							
Best	**0.027500**	0.030000	0.030000	0.030000	0.037500	0.040000	0.037500
Worst	**0.037500**	0.047500	0.042500	0.042500	0.045000	0.045000	0.045000
Mean	**0.033125**	0.038750	0.037500	0.039375	0.041563	0.042188	0.041250
Median	**0.033750**	0.037500	0.037500	0.041250	0.042500	0.042500	0.041250
Std	0.002997	0.005449	0.004146	0.004098	0.002142	**0.001952**	0.002165
Var	0.000009	0.000030	0.000017	0.000017	0.000005	**0.000004**	0.000005
Best no. feat	48	48	48	48	48	48	48
Best no. neurons	225	552	317	259	452	292	216
UCI dataset							
Best	**0.052465**	0.055631	0.054726	0.054726	0.056536	0.059249	0.054274
Worst	**0.054274**	0.059249	0.057440	0.058345	0.059249	0.062415	0.057440
Mean	**0.053068**	0.057440	0.056008	0.056234	0.057892	0.060681	0.056008
Median	**0.052917**	0.057214	0.056083	0.056083	0.057892	0.060380	0.056083
Std	**0.000622**	0.001454	0.000920	0.001422	0.000977	0.001025	0.000991
Var	**0.000000**	0.000002	0.000001	0.000002	0.000001	0.000001	0.000001
Best no. feat	30	30	30	30	30	30	30
Best no. neurons	418	411	450	450	391	395	446
UCI small dataset							
Best	**0.099631**	0.114391	0.110701	0.110701	0.114391	0.118081	0.110701
Worst	**0.114391**	0.132841	0.132841	0.125461	0.129151	0.177122	0.136531
Mean	**0.107011**	0.122386	0.121771	0.118696	0.121771	0.143296	0.123001
Median	**0.107011**	0.121771	0.121771	0.119926	0.119926	0.142066	0.121771
Std	**0.004764**	0.006876	0.009286	0.004958	0.005637	0.019341	0.008159
Var	**0.000023**	0.000047	0.000086	0.000025	0.000032	0.000374	0.000067
Best feature size	9	9	9	9	9	9	9
Params	135	135	116	135	127	133	135

Wilk [132] test for single-problem analysis [129] was first performed in the following way: for each algorithm and every problem, a data sample is constructed by taking the results obtained in each run, and respective p values are calculated for every method—problem pair. Such generated p values are shown in Table 6.

As can be see from the test results, all p values are higher than the threshold significance level $\alpha = 0.05$; therefore, the null hypothesis, which states that the data samples come from normal distribution, cannot be rejected. Therefore, data samples for all method—problem pairs are originating from a normal distribution, and it is safe to use average objective in the statistical tests.

Afterwards, multi-problems multiple-methods statistical analysis was conducted and the data sample for each method was constructed by taking the average objective function value over 30 independent runs for each problem instance. First, requirements for safe use of the parametric tests conditions, including independence, normality, and homoscedasticity of the variances of the data, were checked [133]. Each run was executed independently starting with unique pseudo-random number, confirming that the condition of independence was satisfied. By again using the Shapiro–Wilk test [132], the normality condition was checked and the results for compared methods are shown in Table 7.

To check homoscedasticity based on means, Levene's test [134] is applied, and the p value of 0.64 is obtained, which leads to a conclusion that the homoscedasticity is satisfied.

On the other hand, the calculated p values from the Shapiro–Wilk test for all methods are smaller than $\alpha = 0.05$ (Table 7), providing the conclusion that the safe use of parametric tests is not satisfied; therefore, it was proceeded with non-parametric tests, where the proposed DOSNS was designated as the control method.

To determine the significance of the proposed algorithm performance over other algorithms, the Friedman test [135, 136] and a two-way variance analysis by ranks were conducted, as suggested in [130]. The Friedman test results are

Table 5 Experiment II—detailed metrics for the best solutions of ELM tuning without feature selection (L2)

	ELM–DOSNS	ELM–SNS	ELM–FA	ELM–BA	ELM–ABC	ELM–HHO	ELM–SCA
Kaggle dataset							
Accuracy (%)	**97.25**	97.00	97.00	97.00	96.25	96.00	96.25
Precision 0	**0.984615**	0.937824	0.965347	0.974747	0.955665	0.964646	0.960199
Precision 1	0.960976	0.908213	**0.974747**	0.965347	0.969543	0.955446	0.964824
M.Avg. Precision	**0.972795**	0.923018	0.970047	0.970047	0.962604	0.960046	0.962512
Recall 0	0.960000	0.905000	**0.975000**	0.965000	0.970000	0.955000	0.965000
Recall 1	**0.985000**	0.940000	0.965000	0.975000	0.955000	0.965000	0.960000
M.Avg. Recall	**0.972500**	0.922500	0.970000	0.970000	0.962500	0.960000	0.962500
F1 Score 0	**0.972152**	0.921120	0.970149	0.969849	0.962779	0.959799	0.962594
F1 Score 1	**0.972840**	0.923833	0.969849	0.970149	0.962217	0.960199	0.962406
M.Avg. F1 Score	**0.972496**	0.922476	0.969999	0.969999	0.962498	0.959999	0.962500
UCI dataset							
Accuracy (%)	**94.75**	94.44	94.53	94.53	94.35	93.94	94.57
Precision 0	**0.954689**	0.951477	0.947808	0.952532	0.944792	0.939646	0.951630
Precision 1	0.942155	0.939034	**0.943336**	0.939826	0.942446	0.939200	0.941270
M.Avg. Precision	**0.947705**	0.944544	0.945316	0.945452	0.943485	0.939398	0.945857
Recall 0	0.925434	0.921348	**0.927477**	0.922370	0.926456	0.922370	0.924413
Recall 1	**0.965097**	0.962662	0.959416	0.963474	0.956981	0.952922	0.962662
M.Avg. Recall	**0.947535**	0.944369	0.945274	0.945274	0.943464	0.939394	0.945726
F1 Score 0	**0.939834**	0.936170	0.937532	0.937208	0.935534	0.930928	0.937824
F1 Score 1	**0.953488**	0.950701	0.951308	0.951503	0.949658	0.946011	0.951846
M.Avg. F1 Score	**0.947442**	0.944267	0.945208	0.945173	0.943404	0.939333	0.945637
UCI small dataset							
Accuracy (%)	**90.04**	88.56	88.93	88.93	88.56	88.19	88.93
Precision 0	**0.883929**	0.868421	0.900000	0.857143	0.852174	0.866071	0.867257
Precision 1	1.00000	0.800000	0.727273	1.00000	0.846154	0.733333	0.833333
Precision 2	0.90411	0.908451	0.893333	0.910345	**0.916084**	0.909722	0.910959
M.Avg. Precision	**0.902995**	0.884199	0.883784	0.895367	0.884982	0.878987	0.887491
Recall 0	0.900000	0.900000	0.9	0.927273	0.890909	0.881818	0.890909
Recall 1	**0.650000**	0.600000	0.400000	0.350000	0.550000	0.550000	0.50000
Recall 2	0.93617	0.914894	**0.950355**	0.93617	0.929078	0.929078	0.943262
M.Avg. Recall	**0.900369**	0.885609	0.889299	0.889299	0.885609	0.881919	0.889299
F1 Score 0	0.891892	0.883929	**0.900000**	0.89083	0.871111	0.873874	0.878924
F1 Score 1	**0.787879**	0.685714	0.516129	0.518519	0.666667	0.628571	0.625000
F1 Score 2	0.919861	0.911661	0.920962	0.923077	0.922535	0.919298	**0.926829**
M.Avg. F1 Score	**0.898768**	0.883729	0.882577	0.880131	0.882779	0.879404	0.885109

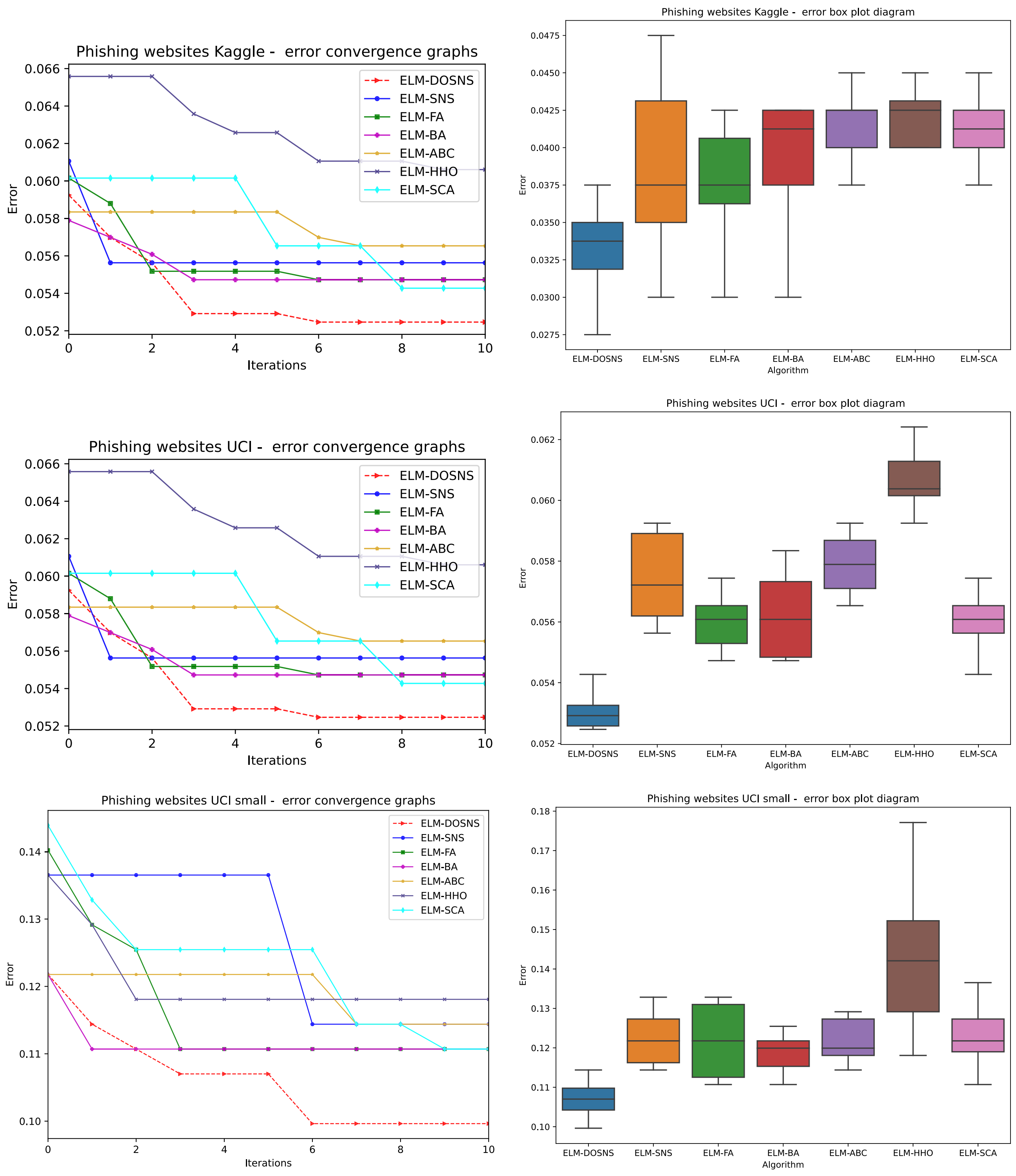


Fig. 12 Experiment II—convergence graphs and box plots of the error for all observed methods and all three data sets

Fig. 13 Experiment II—the PR AUC curves for the best solutions (smallest error) of L2 without feature selection for all observed methods and all three data sets

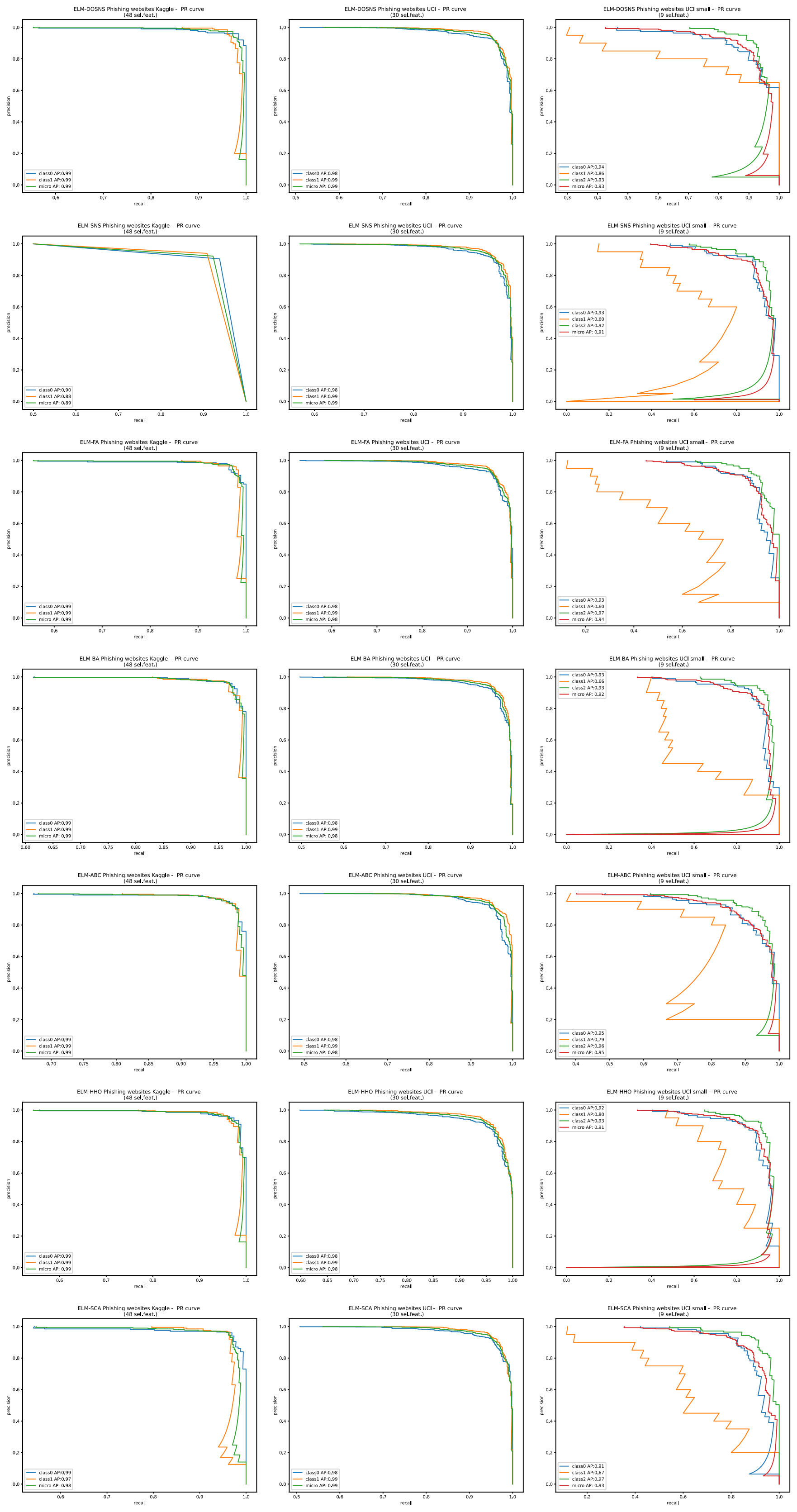


Fig. 14 Experiment II—OvR ROC curves for the best solutions (smallest error) of L2 without feature selection for all observed methods and all three data sets

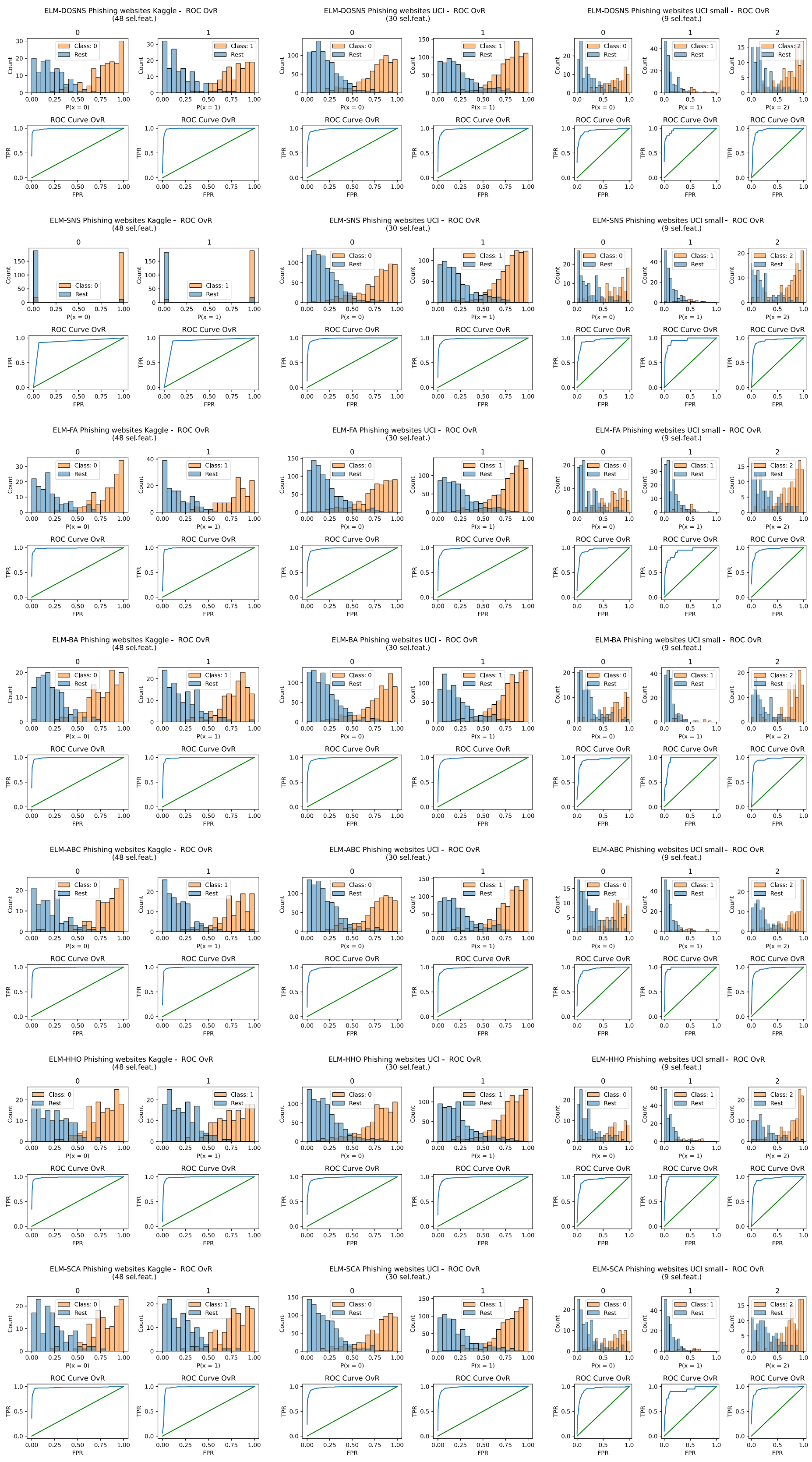


Table 6 Shapiro–Wilk test results for single-problem analysis

Problem	DOSNS	SNS	FA	BA	ABC	HHO	SCA
PW Kaggle FS	0.6753	0.3256	0.2743	0.4561	0.3234	0.4562	0.5283
PW UCI FS	0.4511	0.3335	0.2152	0.4052	0.1236	0.2567	0.5183
PW UCI small FS	0.0975	0.0825	0.1743	0.3561	0.1221	0.3252	0.0945
PW Kaggle FS+HT	0.1742	0.2237	0.1805	0.2465	0.0974	0.3462	0.1285
PW UCI FS+HT	0.1251	0.2357	0.3210	0.5592	0.1245	0.2574	0.2456
PW UCI small FS+HT	0.5552	0.0821	0.1844	0.2564	0.1253	0.2764	0.4253
PW Kaggle HT	0.2751	0.2254	0.1642	0.3564	0.4256	0.1925	0.5281
PW UCI HT	0.3551	0.2275	0.1845	0.2569	0.1936	0.2590	0.1251
PW UCI small HT	0.4752	0.0988	0.1725	0.1554	0.2251	0.1599	0.4241

Table 7 Shapiro–Wilk test results for multi-problems multiple-methods analysis

DOSNS	SNS	FA	BA	ABC	HHO	SCA
0.0129	0.0162	0.0127	0.0213	0.0318	0.0133	0.0325

reported in Table 8. Moreover, the Friedman aligned test was also utilized, and these findings are shown in Table 9.

The results from Tables 8 and 9 statistically indicate that the proposed DOSNS method obtained superior performance in comparison with other algorithms by achieving an average rank value of 1. The second-best result was achieved by FA, with an average rank of 3. The original SNS accomplished an average ranking of 3.94; therefore, the superiority of the proposed DOSNS over original method is also proven. Furthermore, the Friedman statistics, $\chi_r^2 = 21.27$, is greater than the χ^2 critical value, with 6 degrees of freedom (12.59), at significance level $\alpha = 0.05$, and the Friedman p value is 2.22×10^{-16}, inferring that significant differences in results between different methods exist. Consequently, it is possible to reject the null hypothesis (H_0) and state that the proposed DOSNS obtained performance were significantly different from other competitors. Similar conclusions can be derived from the Friedman aligned test results.

As indicated in [137], the Iman and Davenport's test [138] could give results with more precision than the χ^2. The Iman and Davenport's test result is 3.25×10^0, which is significantly larger than the critical value of the F-distribution (2.09×10^0). In addition, the Iman and Devenport p value is 6.73×10^{-2}, which is smaller than the level of significance. Finally, it is concluded that this test also rejects H_0.

Finally, the non-parametric post-hoc Holm's step-down procedure was applied based on the fact that both conducted tests rejected the null hypothesis. These finding are reported in Table 10. In this test, the observed algorithms are sorted in respect of their p values and evaluated to $\alpha/(k-i)$, where k and i represent the degree of freedom ($k = 6$ for this research) and the algorithm number, respectively, after sorting according to the p value in ascending order (corresponding to rank). In this research α values of 0.05 and 0.1 are used in this experiment. The outcomes from Table 10 clearly indicate that the suggested DOSNS significantly outperformed all competing algorithms at both significance levels.

To comprehend the modeling process and identify the most effective model, the explainable artificial intelligence method called SHAP was employed. This approach overcomes the usual trade-off between accuracy and interpretability by offering a precise and significant explanation of the model's choices. By utilizing a game-theory approach that evaluates the influence of individual features on predictions, the SHAP technique determines feature importance through Shapley values [139]. These values distribute the disparity between predictions and the mean predictions among the features and represent a just allocation of payouts to collaborating features with respect to their individual contributions to the combined payout.

SHAP can interpret the impact of a feature in relation to a model's prediction by assigning each feature an importance measure that indicates its contribution to a particular prediction, compared to the prediction in case that feature was set to the baseline value. By generalizing Shapley values and preserving local faithfulness, this technique offers insights into the model's behavior and solves the significant issue of inconsistency while reducing the likelihood of undervaluing a feature with a specific attribution value. It also accounts for interactions between features and enables the interpretation of the model's overall behavior [140].

Aiming to interpret the model and determine the influence features have on the outcome, SHAP diagrams were generated for Kaggle and UCI small data sets. The results from the experiment 1—overall metrics for L2 results (described in Sect. 4.3.1) were used, where the features were chosen by L1 framework, and the best performing ELM–DOSNS model was subjected to the SHAP analysis.

In case of Phishing websites Kaggle data set, 21 features were chosen and this data set was used in SHAP analysis. Details about this data set and description of each feature

Table 8 Friedman statistical test results

Functions	DOSNS	SNS	FA	BA	ABC	HHO	SCA
PW Kaggle FS	1	2	4	7	5	6	3
PW UCI FS	1	5	2	6	4	7	3
PW UCI small FS	1	4	2	7	5	6	3
PW Kaggle FS+HT	1	3.5	2	3.5	6	7	5
PW UCI FS+HT	1	3	5	6	4	7	2
PW UCI small FS+HT	1	5	4	3	7	6	2
PW Kaggle HT	1	3	2	4	6	7	5
PW UCI HT	1	5	2.5	4	6	7	2.5
PW UCI small HT	1	5	3.5	2	3.5	7	6
Average ranking	1	3.94	3	4.72	5.17	6.67	3.5
Rank	1	4	2	5	6	7	3

Table 9 Friedman aligned statistical test results

Functions	DOSNS	SNS	FA	BA	ABC	HHO	SCA
PW Kaggle FS	4	14	16	59	47	57	15
PW UCI FS	5	43	10	56	19	60	13
PW UCI small FS	1	18	3	63	53	62	12
PW Kaggle FS+HT	17	32.5	27	32.5	45	48	41
PW UCI FS+HT	22	24	36	40	35	51	23
PW UCI small FS+HT	6	46	42	20	58	55	8
PW Kaggle HT	7	31	21	37	50	52	49
PW UCI HT	11	39	28.5	30	44	54	28.5
PW UCI small HT	2	34	25.5	9	25.5	61	38
Average ranking	8.33	31.28	23.22	38.5	41.83	55.56	25.28
Rank	1	4	2	5	6	7	3

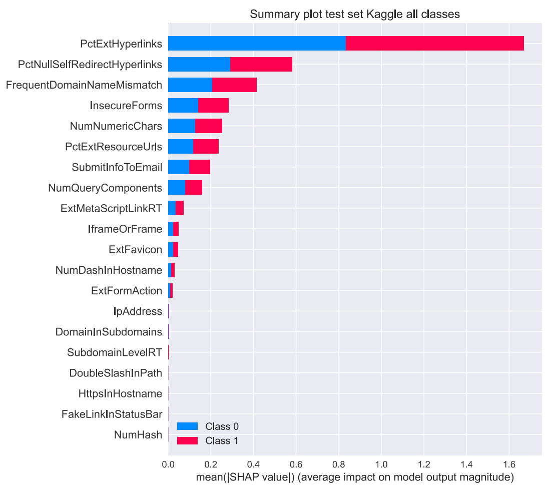
Table 10 Holm's step-down procedure statistical test results

Comparison	*p*_values	Ranking	Alpha = 0.05	Alpha = 0.1	H1	H2
DOSNS vs HHO	0.000000	0	0.0083	0.0167	True	True
DOSNS vs ABC	0.000021	1	0.01	0.02	True	True
DOSNS vs BA	0.000129	2	0.0125	0.025	True	True
DOSNS vs SNS	0.001918	3	0.0167	0.0333	True	True
DOSNS vs SCA	0.007045	4	0.025	0.05	True	True
DOSNS vs FA	0.024767	5	0.05	0.1	True	True

are available on official Kaggle repository[4] [121]. SHAP diagrams by default show 20 features that are the most influential, and as such are provided in this section. Figure 15 brings forward the summary plot of all classes and waterfall chart for class 1 (phishing), while Fig. 16 presents the summary plots for class 0 and class 1 (phishing).

Looking at the SHAP waterfall chart for class 1 (phishing) from Fig. 15, it can be noted that the PctExtHyperlinks attribute is the most influential, followed by the features NumNumericChars and NumQueryComponents. Analyzing the summary plot for class 1, shown in Fig. 16, it is possible to note that PctExtHyperlinks attribute is in direct correlation with class 1, as the increased amount of the external hyperlinks will highly likely indicate that the particular website is phishing. In addition, probability of classification as class 1 increases with the increase of properties PctNullSelfRedirectHyperlinks, FrequentDomainNameMismatch and InsecureForms, as well as NumNumericChars feature. All these observations are in line with the practice, where the phishing websites typically have large number of external links, insecure forms, and large numbers of numeric characters.

[4] https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning.



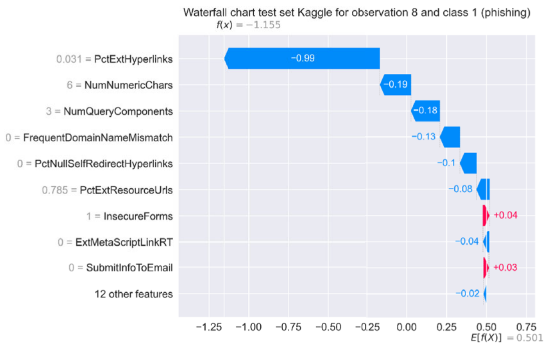
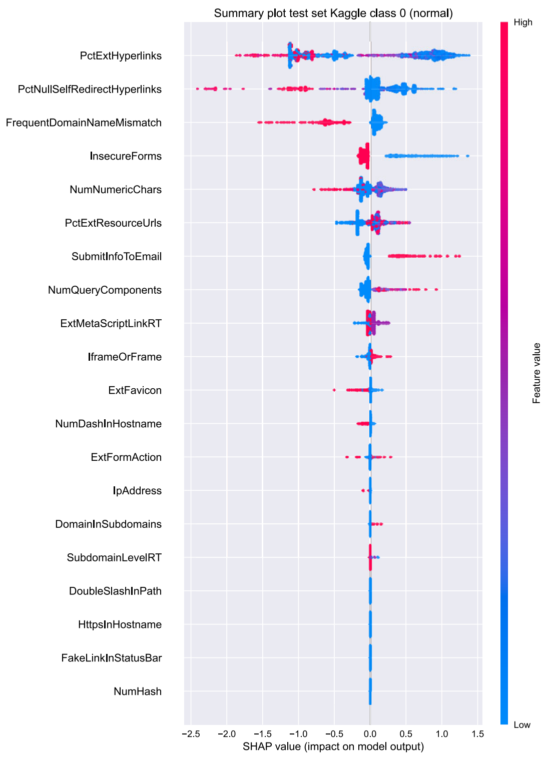


Fig. 15 SHAP summary plot (left side) and waterfall chart for Kaggle data set (right side)



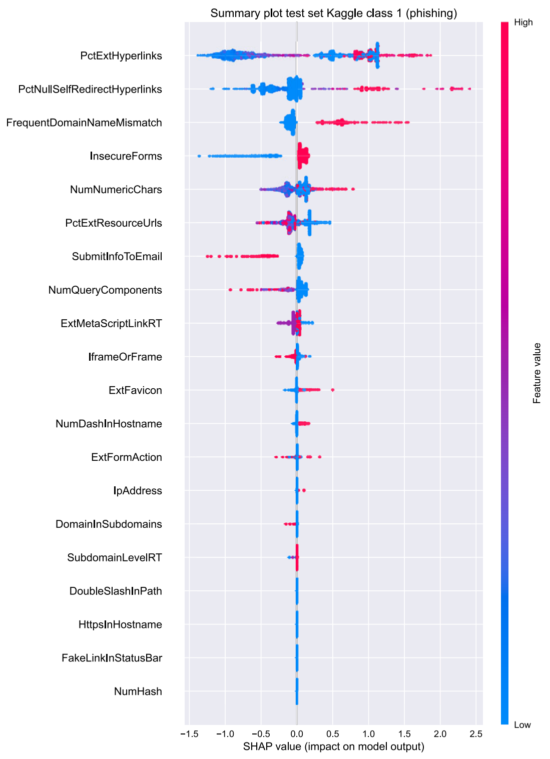


Fig. 16 SHAP summary plots for class 0 (left side) and class 1 of Kaggle data set (right side)

In case of UCI small data set, 7 features were chosen and this data set was used in SHAP analysis. Details about this data set and description of each feature are available on UCI Machine Learning Repository [5] [125, 126]. Figure 17 brings forward the summary plot of all classes and waterfall chart for class 1 (phishing), while Fig. 18 presents the summary plots for class 0 (normal), class 1 (suspicious) and class 2 (phishing) for the UCI small data set.

According to the waterfall diagram shown in Fig. 17, it is possible to observe that the most important features regarding the UCI small data set are SFH (server form handler), URL_of_Anchor and Request_URL, followed by popUpWindow, URL_Length and SSL final state. All these features are in direct correlation with the classification as class 2 (phishing), as the increase of these features will also increase the probability of the web site being classified as phishing. Once again, these observations are confirmed in the practical

[5] https://archive.ics.uci.edu/ml/datasets/Website+Phishing.

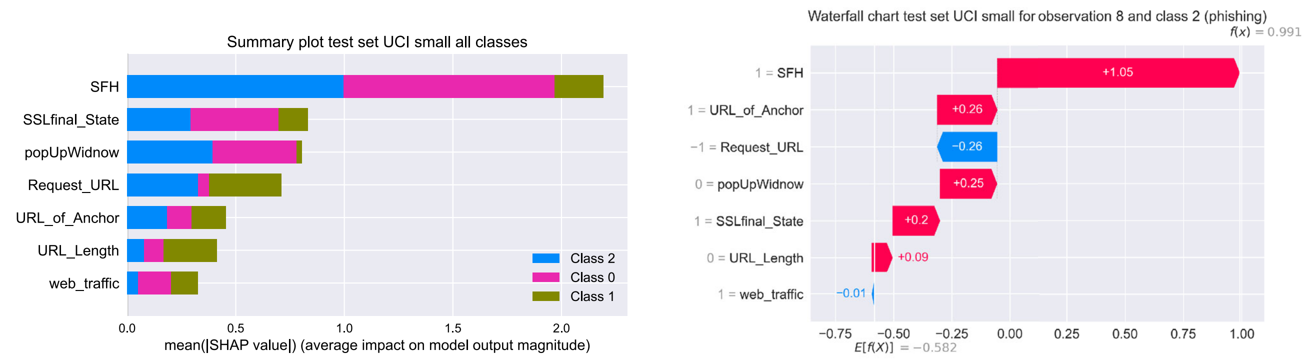


Fig. 17 SHAP summary plot (left side) and waterfall chart for UCI small data set (right side)

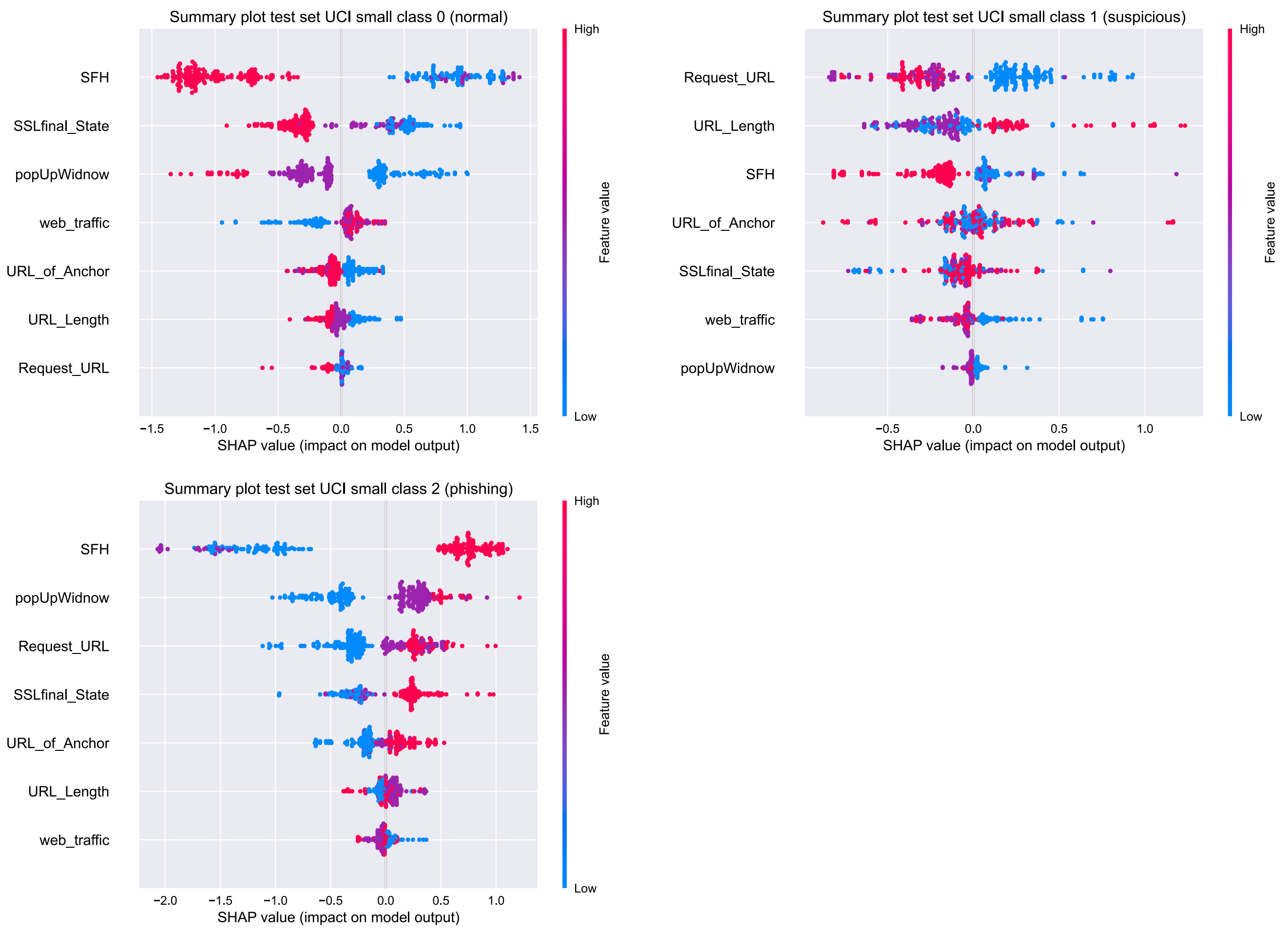


Fig. 18 SHAP summary plots for class 0, class 1 and class 2 of UCI small data set

applications, as data phishing sites commonly have indicators as URL length, request a URL, the URL of anchor, SFH, submitting to email, SSL final state and abnormal URL, as observed by [141].

Conclusion

The research proposed in this manuscript addresses two of the most important ML challenges, feature selection, and hyper-parameters optimization. The presented study tried to improve phishing website detection by tuning ELM that utilizes the most relevant subset of phishing website data sets features.

To accomplish this goal, a novel DOSNS has been developed and incorporated into devised two-level cooperative framework. The framework consists of two levels—L1, which deals with feature selection, and L2, which performs ELM tuning. Levels in the two-level framework can be used independently, i.e., performing only feature selection or ELM tuning. In addition, the L1 can execute in a cooperative or individual mode. When set to cooperative mode, all metaheuristics included in the framework perform feature selection independently; however, at the end of execution (after the predetermined number of runs), the selected feature subset generated by best-performing metaheuristics is used as the input to L2 and then all metaheuristics perform ELM tuning using the same set of selected features. Conversely, if the L1 is set to individual mode, then all metaheuristics use their own best set of selected features from L1, as an input to L2, regardless of the classifier performance with the chosen set of features.

The proposed DOSNS has been validated against 6 cutting-edge metaheuristics, that were also incorporated into the devised framework, over two experiments. The first experiment utilized both L1 and L2 of the proposed optimization framework, where the L1 was adjusted in cooperative mode. The second simulation is focused only on tuning the ELM with all employed features (without the feature selection), using only the L2 part of the framework. The goal of this simulation was to test the performance of the suggested DOSNS for large-scale global optimization with many parameters (components).

All methods were validated against three challenging phishing websites data sets, which represent one of the most important challenges in the web security domain. Data sets are available publicly and they were retrieved from UCI and Kaggle repositories. All methods were compared with respect to objective and error, separately for layers 1 and 2 over several independent runs, and detailed metrics of the final outcomes (output of layer 2), including precision, recall, f1 score, receiver operating characteristics, and precision recall area under the curves.

The rigid statistical tests that were conducted for reported experimental findings suggest that the proposed DOSNS is an efficient and robust optimizer, achieving on average better results than other state-of-the-art metaheuristics.

Some limitations of the proposed research refer to the fact that the DOSNS still hasn't been validated against tuning other ML models and that further investigation is required with different transformation functions for feature selection challenge. In addition, for handling moderately and highly imbalanced data sets, investigation with more promising fitness functions is required. Therefore, these domains will be included as part of the future work in this promising area.

Acknowledgements This work was supported by the Korea Technology and Information Promotion Agency (TIPA) for SMEs grant funded by the Korea government (Ministry of SMEs and Startups) (No.S3271954) and This work was supported by the National Research Foundation of Korea (NRF) grant funded by the korea government (MSIT) (No. 2022H1D8A3038040), and the Soonchunhyang University Research Fund.

Author Contributions N.B. and M.Z. implemented methods and performed simulations. M.M. and M.A. performed results' visualization. K.V. and J.L. analysed the results. Y.N. wrote original draft. M.A. revise and editing. All authors reviewed the manuscript.

Data availability The data sets used and/or analysed during the current study available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Piercy N (2014) Online service quality: Content and process of analysis. J Marketing Manag 30(7–8):747–785
2. Lee S, Lee S, Park Y (2007) A prediction model for success of services in e-commerce using decision tree: E-customer's attitude towards online service. Expert Syst Appl 33(3):572–581
3. Rita P, Oliveira T, Farisa A (2019) The impact of e-service quality and customer satisfaction on customer behavior in online shopping. Heliyon 5(10):02690
4. Westerlund M (2020) Digitalization, internationalization and scaling of online smes. Technology Innovation Management Review 10(4)
5. Bressan A, Duarte Alonso A, Kok SK (2021) Confronting the unprecedented: micro and small businesses in the age of covid-19. Int J Entrepreneurial Behav Res 27(3):799–820
6. Patel A, Shah N, Ramoliya D, Nayak A (2020) A detailed review of cloud security: issues, threats & attacks. In: 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, pp 758–764
7. Khan NA, Brohi SN, Zaman N (2020) Ten deadly cyber security threats amid covid-19 pandemic
8. Salahdine F, Kaabouch N (2019) Social engineering attacks: A survey. Future Internet 11(4). https://doi.org/10.3390/fi11040089

9. Safi A, Singh S (2023) A systematic literature review on phishing website detection techniques. J King Saud Univ—Comput Inform Sci 35(2):590–611. https://doi.org/10.1016/j.jksuci.2023.01.004
10. Akerkar R (2019) Artificial intelligence for business. Springer, Cham, Switzerland
11. Buchanan B (2019) Artificial intelligence in finance. The Alan Turing Institute, London, UK
12. Hamet P, Tremblay J (2017) Artificial intelligence in medicine. Metabolism 69:36–40
13. Dias R, Torkamani A (2019) Artificial intelligence in clinical and genomic diagnostics. Genome Med 11(1):1–12
14. Vijayalakshmi M, Mercy Shalinie S, Yang MH, U RM (2020) Web phishing detection techniques: a survey on the state-of-the-art, taxonomy and future directions. Iet Netw 9(5):235–246
15. Jain AK, Gupta B (2022) A survey of phishing attack techniques, defence mechanisms and open research challenges. Enterprise Inform Syst 16(4):527–565
16. Fredj OB, Cheikhrouhou O, Krichen M, Hamam H, Derhab A (2021) An owasp top ten driven survey on web application protection methods. In: Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France, November 4–6, 2020, Revised Selected Papers 15. Springer, pp 235–252
17. Tanasković TM, Živković MŽ (2011) Security principles for web applications. In: 2011 19th Telecommunications Forum (TELFOR) Proceedings of Papers. IEEE, pp 1507–1510
18. Dhaliwal SS, Nahid A-A, Abbas R (2018) Effective intrusion detection system using xgboost. Information 9(7):149
19. Kanimozhi V, Jacob TP (2019) Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing. In: 2019 International Conference on Communication and Signal Processing (ICCSP). IEEE, pp 0033–0036
20. Alqahtani H, Sarker IH, Kalim A, Hossain M, Md S, Ikhlaq S, Hossain S (2020) Cyber intrusion detection using machine learning classification techniques. In: International Conference on Computing Science, Communication and Security. Springer, pp 121–131
21. Alsariera YA, Adeyemo VE, Balogun AO, Alazzawi AK (2020) Ai meta-learners and extra-trees algorithm for the detection of phishing websites. IEEE Access 8:142532–142542
22. Alam MN, Sarma D, Lima FF, Saha I, Hossain S, et al (2020) Phishing attacks detection using machine learning approach. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, pp 1173–1179
23. Gangavarapu T, Jaidhar C, Chanduka B (2020) Applicability of machine learning in spam and phishing email filtering: review and approaches. Artificial Intell Rev 53(7):5019–5081
24. Doshi R, Apthorpe N, Feamster N (2018) Machine learning ddos detection for consumer internet of things devices. In: 2018 IEEE Security and Privacy Workshops (SPW). IEEE, pp 29–35
25. Injadat M, Moubayed A, Shami A (2020) Detecting botnet attacks in iot environments: An optimized machine learning approach. In: 2020 32nd International Conference on Microelectronics (ICM). IEEE, pp 1–4
26. Soe YN, Feng Y, Santosa PI, Hartanto R, Sakurai K (2020) Machine learning-based iot-botnet attack detection with sequential architecture. Sensors 20(16):4372
27. Makkar A, Garg S, Kumar N, Hossain MS, Ghoneim A, Alrashoud M (2020) An efficient spam detection technique for iot devices using machine learning. IEEE Trans Ind Inform 17(2):903–912
28. Zainab A, Refaat S, Bouhali O (2020) Ensemble-based spam detection in smart home iot devices time series data using machine learning techniques. Information 11(7):344
29. Kumar N, Sonowal S, et al (2020) Email spam detection using machine learning algorithms. In: 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, pp 108–113
30. Bishop CM (1994) Neural networks and their applications. Rev Sci Instruments 65(6):1803–1832
31. Bezdan T, Zivkovic M, Tuba E, Strumberger I, Bacanin N, Tuba M (2020) Glioma brain tumor grade classification from mri using convolutional neural networks designed by modified fa. In: international conference on intelligent and fuzzy systems. Springer, pp 955–963
32. Bacanin N, Zivkovic M, Jovanovic L, Ivanovic M, Rashid TA (2022) Training a multilayer perception for modeling stock price index predictions using modified whale optimization algorithm. In: Computational Vision and Bio-Inspired Computing. Springer, pp 415–430
33. Strumberger I, Tuba E, Bacanin N, Jovanovic R, Tuba M (2019) Convolutional neural network architecture design by the tree growth algorithm framework. In: 2019 international joint conference on neural networks (IJCNN). IEEE, pp 1–8
34. Bacanin N, Bezdan T, Tuba E, Strumberger I, Tuba M (2020) Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics. Algorithms 13(3):67
35. Strumberger I, Tuba E, Bacanin N, Zivkovic M, Beko M, Tuba M (2019) Designing convolutional neural network architecture by the firefly algorithm. In: 2019 International Young Engineers Forum (YEF-ECE). IEEE, pp 59–65
36. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
37. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
38. Zhang Q, Lu J, Jin Y (2021) Artificial intelligence in recommender systems. Complex Intell Syst 7(1):439–457
39. Bacanin N, Tuba E, Zivkovic M, Strumberger I, Tuba M (2019) Whale optimization algorithm with exploratory move for wireless sensor networks localization. In: International conference on hybrid intelligent systems. Springer, pp 328–338
40. Abdel-Basset M, Abdel-Fatah L, Sangaiah AK (2018) Metaheuristic algorithms: A comprehensive review. Computational intelligence for multimedia big data on the cloud with engineering applications 185–231
41. Talatahari S, Bayzidi H, Saraee M (2021) Social network search for global optimization. IEEE Access 9:92815–92863
42. Do NQ, Selamat A, Krejcar O, Herrera-Viedma E, Fujita H (2022) Deep learning for phishing detection: Taxonomy, current challenges and future directions. IEEE Access
43. Ahmed N, Amin R, Aldabbas H, Koundal D, Alouffi B, Shah T (2022) Machine learning techniques for spam detection in email and iot platforms: analysis and research challenges. Secur Commun Netw 2022:1–19
44. Rao S, Verma AK, Bhatia T (2021) A review on social spam detection: challenges, open issues, and future directions. Expert Syst Appl 186:115742
45. Chandrashekar G, Sahin F (2014) A survey on feature selection methods. Comput Electr Eng 40(1):16–28
46. Trunk GV (1979) A problem of dimensionality: a simple example. IEEE Trans Pattern Anal Mach Intell 3:306–307
47. Van Der Maaten L, Postma E, Van den Herik J et al (2009) Dimensionality reduction: a comparative. J Mach Learn Res 10(66–71):13
48. Levine MD (1969) Feature extraction: a survey. Proc IEEE 57(8):1391–1407
49. Bommert A, Welchowski T, Schmid M, Rahnenführer J (2022) Benchmark of filter methods for feature selection in high-dimensional gene expression survival data. Briefings Bioinform 23(1):354

50. Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: 2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541), vol 2. Ieee, pp 985–990
51. Alencar AS, Neto ARR, Gomes JPP (2016) A new pruning method for extreme learning machines via genetic algorithms. Appl Soft Comput 44:101–107
52. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2009) Op-elm: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
53. Zhu Q-Y, Qin AK, Suganthan PN, Huang G-B (2005) Evolutionary extreme learning machine. Pattern Recognit 38(10):1759–1763
54. Blum C, Li X (2008) Swarm intelligence in optimization. In: Swarm Intelligence. Springer, Berlin, Heidelberg, pp 43–85
55. Yang X-S (2014) Swarm intelligence based algorithms: a critical analysis. Evol Intell 7:17–28
56. Bansal JC, Singh PK, Pal NR et al (2019) Evolutionary and swarm intelligence algorithms, vol 779. Springer, Cham, Switzerland
57. Yang X-S, Deb S, Zhao Y-X, Fong S, He X (2018) Swarm intelligence: past, present and future. Soft Comput 22:5923–5933
58. Abdulrahman SM (2017) Using swarm intelligence for solving np-hard problems. Acad J Nawroz Univ 6(3):46–50
59. Tkatek S, Bahti O, Lmzouari Y, Abouchabaka J (2020) Artificial intelligence for improving the optimization of np-hard problems: a review. Int J Adv Trends Comput Sci Appl 9(5)
60. Pang W, Wang K-p, Zhou C-g, Dong L-j (2004) Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In: The fourth international conference on computer and information technology, 2004. CIT'04. IEEE pp 796–800
61. Karaboga D (2010) Artificial bee colony algorithm. Scholarpedia 5(3):6915
62. Bacanin N, Bezdan T, Venkatachalam K, Zivkovic M, Strumberger I, Abouhawwash M, Ahmed AB (2021) Artificial neural networks hidden unit and weight connection optimization by quasi-refection-based learning artificial bee colony algorithm. IEEE Access 9:169135–169155
63. Cuk A, Bezdan T, Bacanin N, Zivkovic M, Venkatachalam K, Rashid TA, Devi VK (2021) Feedforward multi-layer perceptron training by hybridized method between genetic algorithm and artificial bee colony. Data Sci Data Anal 279
64. Tuba M, Bacanin N (2014) Artificial bee colony algorithm hybridized with firefly algorithm for cardinality constrained mean-variance portfolio selection problem. Appl Math Inform Sci 8(6):2831
65. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
66. Strumberger I, Bacanin N, Tuba M, Tuba E (2019) Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. Appl Sci 9(22):4893
67. Strumberger I, Bezdan T, Ivanovic M, Jovanovic L (2021) Improving energy usage in wireless sensor networks by whale optimization algorithm. In: 2021 29th Telecommunications Forum (TELFOR). IEEE, pp 1–4
68. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
69. Bacanin N, Bezdan T, Tuba E, Strumberger I, Tuba M, Zivkovic M (2019) Task scheduling in cloud computing environment by grey wolf optimizer. In: 2019 27th telecommunications forum (TELFOR). IEEE, pp 1–4
70. Zivkovic M, Bacanin N, Zivkovic T, Strumberger I, Tuba E, Tuba M (2020) Enhanced grey wolf algorithm for energy efficient wireless sensor networks. In: 2020 Zooming Innovation in Consumer Technologies Conference (ZINC). IEEE, pp 87–92
71. Mohammed HM, Abdul ZK, Rashid TA, Alsadoon A, Bacanin N (2021) A new k-means gray wolf algorithm for engineering problems. World J Eng
72. Yang X-S (2009) Firefly algorithms for multimodal optimization. In: International Symposium on Stochastic Algorithms. Springer, pp 169–178
73. Jovanovic D, Antonijevic M, Stankovic M, Zivkovic M, Tanaskovic M, Bacanin N (2022) Tuning machine learning models using a group search firefly algorithm for credit card fraud detection. Mathematics 10(13):2272
74. Zivkovic M, Tair M, Venkatachalam K, Bacanin N, Hubálovskỳ Š, Trojovskỳ P (2022) Novel hybrid firefly algorithm: an application to enhance xgboost tuning for intrusion detection classification. PeerJ Comput Sci 8:956
75. Tair M, Bacanin N, Zivkovic M, Venkatachalam K (2022) A chaotic oppositional whale optimisation algorithm with firefly search for medical diagnostics. Comput. Mater. Contin 72:959–982
76. Bacanin N, Stoean R, Zivkovic M, Petrovic A, Rashid TA, Bezdan T (2021) Performance of a novel chaotic firefly algorithm with enhanced exploration for tackling global optimization problems: Application for dropout regularization. Mathematics 9(21):2705
77. Bezdan T, Cvetnic D, Gajic L, Zivkovic M, Strumberger I, Bacanin N (2021) Feature selection by firefly algorithm with improved initialization strategy. In: 7th Conference on the Engineering of Computer Based Systems, pp 1–8
78. Bacanin N, Zivkovic M, Sarac M, Petrovic A, Strumberger I, Antonijevic M, Petrovic A, Venkatachalam K (2022) A novel multiswarm firefly algorithm: An application for plant classification. In: International Conference on Intelligent and Fuzzy Systems. Springer, pp 1007–1016
79. Mirjalili S (2016) Sca: a sine cosine algorithm for solving optimization problems. Knowledge-Based Syst 96:120–133
80. AlHosni N, Jovanovic L, Antonijevic M, Bukumira M, Zivkovic M, Strumberger I, Mani JP, Bacanin N (2022) The XGBoost model for network intrusion detection boosted by enhanced sine cosine algorithm. In: Third International conference on image processing and capsule networks. Springer, pp 213–228. https://doi.org/10.1007/978-3-031-12413-6_17
81. Zivkovic M, Jovanovic L, Ivanovic M, Krdzic A, Bacanin N, Strumberger I (2022) Feature selection using modified sine cosine algorithm with covid-19 dataset. In: Evolutionary computing and mobile sustainable networks. Springer, pp 15–31
82. Bacanin N, Zivkovic M, Salb M, Strumberger I, Chhabra A (2022) Convolutional neural networks hyperparameters optimization using sine cosine algorithm. In: Sentimental Analysis and Deep Learning. Springer, pp 863–878
83. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609
84. Zivkovic M, Stoean C, Petrovic A, Bacanin N, Strumberger I, Zivkovic T (2021) A novel method for covid-19 pandemic information fake news detection based on the arithmetic optimization algorithm. In: 2021 23rd international symposium on symbolic and numeric algorithms for scientific computing (SYNASC). IEEE, pp 259–266
85. Khatir S, Tiachacht S, Le Thanh C, Ghandourah E, Mirjalili S, Wahab MA (2021) An improved artificial neural network using arithmetic optimization algorithm for damage assessment in fgm composite plates. Composite Struct 273:114287
86. Kaveh A, Hamedani KB (2022) Improved arithmetic optimization algorithm and its application to discrete structural optimization. In: Structures, vol 35. Elsevier, pp 748–764
87. Bacanin N, Petrovic A, Antonijevic M, Zivkovic M, Sarac M, Tuba E, Strumberger I (2023) Intrusion detection by XGBoost model tuned by improved social network search algorithm. In:

Modelling and Development of Intelligent Systems. Springer, pp 104–121. https://doi.org/10.1007/978-3-031-27034-5_7

88. Rao RV, Savsani VJ, Vakharia D (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput-aided Design 43(3):303–315
89. Srinivasan V, Palani P, Balamurugan S (2021) Experimental investigation on edm of si3n4-tin using grey relational analysis coupled with teaching-learning-based optimization algorithm. Ceramics Int 47(13):19153–19168
90. Alanazi MS (2021) A modified teaching-learning-based optimization for dynamic economic load dispatch considering both wind power and load demand uncertainties with operational constraints. IEEE Access 9:101665–101680
91. Dokeroglu T, Sevinc E (2021) Memetic teaching-learning-based optimization algorithms for large graph coloring problems. Eng Appl Artificial Intell 102:104282
92. Kashan AH (2009) League championship algorithm: a new algorithm for numerical function optimization. In: 2009 International conference of soft computing and pattern recognition. IEEE, pp 43–48
93. Xu W, Wang R, Yang J (2018) An improved league championship algorithm with free search and its application on production scheduling. J Intell Manuf 29(1):165–174
94. Abdulhamid SM, Latiff MSA, Idris I (2015) Tasks scheduling technique using league championship algorithm for makespan minimization in iaas cloud. arXiv preprint arXiv:1510.03173
95. Alimoradi MR, Kashan AH (2018) A league championship algorithm equipped with network structure and backward q-learning for extracting stock trading rules. Appl Soft Comput 68:478–493
96. Zivkovic M, Bacanin N, Venkatachalam K, Nayyar A, Djordjevic A, Strumberger I, Al-Turjman F (2021) Covid-19 cases prediction by using hybrid machine learning and beetle antennae search approach. Sustain Cities Soc 66:102669
97. Bezdan T, Zivkovic M, Bacanin N, Chhabra A, Suresh M (2022) Feature selection by hybrid brain storm optimization algorithm for covid-19 classification. Journal of Computational Biology
98. Budimirovic N, Prabhu E, Antonijevic M, Zivkovic M, Bacanin N, Strumberger I, Venkatachalam K (2022) Covid-19 severity prediction using enhanced whale with salp swarm feature classification. Computers, Materials and Continua 1685–1698
99. Strumberger I, Rakic A, Stanojlovic S, Arandjelovic J, Bezdan T, Zivkovic M, Bacanin N (2021) Feature selection by hybrid binary ant lion optimizer with covid-19 dataset. In: 2021 29th telecommunications forum (TELFOR). IEEE, pp 1–4
100. Zivkovic M, Petrovic A, Bacanin N, Milosevic S, Veljic V, Vesic A (2022) The covid-19 images classification by mobilenetv3 and enhanced sine cosine metaheuristics. In: Mobile computing and sustainable informatics. Springer, pp 937–950
101. Bezdan T, Milosevic S, Venkatachalam K, Zivkovic M, Bacanin N, Strumberger I (2021) Optimizing convolutional neural network by hybridized elephant herding optimization algorithm for magnetic resonance image classification of glioma brain tumor grade. In: 2021 Zooming Innovation in Consumer Technologies Conference (ZINC). IEEE, pp 171–176
102. Salb M, Zivkovic M, Bacanin N, Chhabra A, Suresh M (2022) Support vector machine performance improvements for cryptocurrency value forecasting by enhanced sine cosine algorithm. In: Computer vision and robotics. Springer, pp 527–536
103. AlHosni N, Jovanovic L, Antonijevic M, Bukumira M, Zivkovic M, Strumberger I, Mani JP, Bacanin N (2022) The xgboost model for network intrusion detection boosted by enhanced sine cosine algorithm. In: International Conference on Image Processing and Capsule Networks. Springer, pp 213–228
104. Tair M, Bacanin N, Zivkovic M, Venkatachalam K, Strumberger I (2022) Xgboost design by multi-verse optimiser: An application for network intrusion detection. In: Mobile Computing and Sustainable Informatics. Springer, pp 1–16
105. Zivkovic M, Bacanin N, Arandjelovic J, Rakic A, Strumberger I, Venkatachalam K, Joseph PM (2022) Novel harris hawks optimization and deep neural network approach for intrusion detection. In: Proceedings of International Joint Conference on Advances in Computational Intelligence. Springer, pp 239–250
106. Bacanin N, Stoean C, Zivkovic M, Jovanovic D, Antonijevic M, Mladenovic D (2022) Multi-swarm algorithm for extreme learning machine optimization. Sensors 22(11):4204
107. Zivkovic M, Vesic A, Bacanin N, Strumberger I, Antonijevic M, Jovanovic L, Marjanovic M (2022) An improved animal migration optimization approach for extreme learning machine tuning. In: International Conference on Intelligent and Fuzzy Systems. Springer, pp 3–13
108. Alshamiri AK, Singh A, Surampudi BR (2018) Two swarm intelligence approaches for tuning extreme learning machine. Int J Mach Learn Cybernet 9(8):1271–1283
109. Guha R, Ghosh M, Singh PK, Sarkar R, Nasipuri M (2021) A hybrid swarm and gravitation-based feature selection algorithm for handwritten indic script classification problem. Complex Intell Syst 7(2):823–839
110. Jain R, Joseph T, Saxena A, Gupta D, Khanna A, Sagar K, Ahlawat AK (2021) Feature selection algorithm for usability engineering: a nature inspired approach. Complex & Intelligent Systems 1–11
111. Alkan B, Kaniappan Chinnathai M (2021) Performance comparison of recent population-based metaheuristic optimisation algorithms in mechanical design problems of machinery components. Machines 9(12):341
112. Gnetchejo PJ, Essiane SN, Dadjé A, Wapet DM, Ele P (2022) Optimal design of the modelling parameters of photovoltaic modules and array through metaheuristic with secant method. Energy Conv Manag 10(15):100273
113. Yang X-S, Xingshi H (2013) Firefly algorithm: Recent advances and applications. Int J Swarm Intell 1(1):36–50
114. Rahnamayan S, Tizhoosh HR, Salama MMA (2007) Quasi-oppositional differential evolution. In: 2007 IEEE Congress on Evolutionary Computation, pp 2229–2236. https://doi.org/10.1109/CEC.2007.4424748
115. Cheng S, Shi Y (2011) Diversity control in particle swarm optimization. In: 2011 IEEE Symposium on Swarm Intelligence. IEEE, pp 1–9
116. Miche Y, Sorjamaa A, Lendasse A (2008) Op-elm: theory, experiments and a toolbox. In: International Conference on Artificial Neural Networks. Springer, pp 145–154
117. Mohd Yusof N, Muda AK, Pratama SF, Abraham A (2022) A novel nonlinear time-varying sigmoid transfer function in binary whale optimization algorithm for descriptors selection in drug classification. Molecular Diversity 1–10
118. Wang J, Khishe M, Kaveh M, Mohammadi H (2021) Binary chimp optimization algorithm (bchoa): A new binary meta-heuristic for solving optimization problems. Cognit Comput 13(5):1297–1316
119. Hassouneh Y, Turabieh H, Thaher T, Tumar I, Chantar H, Too J (2021) Boosted whale optimization algorithm with natural selection operators for software fault prediction. IEEE Access 9:14239–14258
120. Abdollahzadeh B, Gharehchopogh FS (2021) A multi-objective optimization algorithm for feature selection problems. Engineering with Computers 1–19
121. Tan CL (2018) Phishing dataset for machine learning: Feature evaluation. Mendeley Data 1:2018
122. Mohammad RM, Thabtah F, McCluskey L (2015) Phishing websites features. University of Huddersfield, School of Computing and Engineering

123. Mohammad RM, Thabtah F, McCluskey L (2014) Intelligent rule-based phishing websites classification. IET Inform Secur 8(3):153–160
124. Mohammad RM, Thabtah F, McCluskey L (2014) Predicting phishing websites based on self-structuring neural network. Neural Comput Appl 25(2):443–458
125. Dua D, Graff C (2017) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
126. Abdelhamid N, Ayesh A, Thabtah F (2014) Phishing detection based associative classification data mining. Expert Syst Appl 41(13):5948–5959
127. Yang X-S (2011) Bat algorithm for multi-objective optimisation. Int J io-Inspired Comput 3(5):267–274
128. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: Algorithm and applications. Future Generation Comput Syst 97:849–872
129. Eftimov T, Korošec P, Seljak BK (2016) Disadvantages of statistical comparison of stochastic optimization algorithms. Proceedings of the Bioinspired Optimizaiton Methods and their Applications, BIOMA, 105–118
130. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18
131. García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. J Heuristics 15(6):617–644
132. Shapiro SS, Francia R (1972) An approximate analysis of variance test for normality. J Am Stat Assoc 67(337):215–216
133. LaTorre A, Molina D, Osaba E, Poyatos J, Del Ser J, Herrera F (2021) A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. Swarm Evol Comput 67:100973
134. Glass GV (1966) Testing homogeneity of variances. Am Educ Res J 3(3):187–190
135. Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701
136. Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11(1):86–92
137. Sheskin DJ (2020) Handbook of parametric and nonparametric statistical procedures. Chapman and Hall/CRC, Boca Raton, Florida
138. Iman RL, Davenport JM (1980) Approximations of the critical region of the fbietkan statistic. Commun Satistics-Theory Methods 9(6):571–595
139. Lundberg SM, Lee S-I (2017) A unified approach to interpreting model predictions. Advances in neural information processing systems 30
140. Lundberg SM, Erion G, Chen H, DeGrave A, Prutkin JM, Nair B, Katz R, Himmelfarb J, Bansal N, Lee S-I (2020) From local explanations to global understanding with explainable ai for trees. Nat Mach Intell 2(1):56–67
141. Pratiwi M, Lorosae T, Wibowo F (2018) Phishing site detection analysis using artificial neural network. In: Journal of Physics: Conference Series, vol 1140. IOP Publishing, p 012048

Authors and Affiliations

Nebojsa Bacanin[1] · Miodrag Zivkovic[1] · Milos Antonijevic[1] · K. Venkatachalam[2] · Jinseok Lee[3] · Yunyoung Nam[4] · Marina Marjanovic[1] · Ivana Strumberger[1] · Mohamed Abouhawwash[5,6]

Miodrag Zivkovic
mzivkovic@singidunum.ac.rs

Milos Antonijevic
mantonijevic@singidunum.ac.rs

K. Venkatachalam
venkatachalam.k@ieee.org

Jinseok Lee
gonasago@khu.ac.kr

Marina Marjanovic
mmarijanovic@singidunum.ac.rs

Ivana Strumberger
istrumberger@singidunum.ac.rs

Mohamed Abouhawwash
abouhaww@msu.edu

[1] Department of Informatics and Computing, Singidunum University, Danijelova 32, Belgrade 11000, Serbia

[2] Department of Applied Cybernetics, University of Hradec Králové, Hradec Králové 50003, Czech Republic

[3] Department of Biomedical Engineering, Kyung Hee University, Yongin, South Korea

[4] Department of Computer Science and Engineering, Soonchunhyang University, Asan, South Korea

[5] Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt

[6] Department of Computational Mathematics, Science, and Engineering (CMSE), College of Engineering, Michigan State University, East Lansing 48824, USA