Telecommunication Systems
https://doi.org/10.1007/s11235-020-00739-w

Phishing website detection using support vector machines and nature-inspired optimization algorithms

Sagnik Anupam[1] · Arpan Kumar Kar[2]

Accepted: 5 November 2020

Abstract
Phishing websites are amongst the biggest threats Internet users face today, and existing methods like blacklisting, using SSL certificates, etc. often fail to keep up with the increasing number of threats. This paper aims to utilise different properties of a website URL, and use a machine learning model to classify websites as phishing and non-phishing. These properties include the IP address length, the authenticity of the HTTPs request being sent by the website, usage of pop-up windows to enter data, Server Form Handler status, etc. A Support Vector Machine binary classifier trained on an existing dataset has been used to predict if a website was a legitimate website or not, by finding an optimum hyperplane to separate the two categories. This optimum hyperplane is found with the help of four optimization algorithms, the Bat Algorithm, the Firefly Algorithm, the Grey Wolf Optimiser algorithm and the Whale Optimization Algorithm, which are inspired by various natural phenomena. Amongst the four nature-inspired optimization algorithms, it has been determined that the Grey Wolf Optimiser algorithm's performance is significantly better than that of the Firefly Algorithm, but there is no significant difference while comparing the performance of any other pair of algorithms. However, all four nature-inspired optimization algorithms perform significantly better than the grid-search optimized Random Forest classifier model described in earlier research.

Keywords Phishing · Machine learning · Swarm intelligence · Classification · Cybersecurity

1 Introduction

Phishing is one of the most challenging security problems faced by the world today, in part due to the large number of online transactions that take place daily. It refers to the practice of trying to obtain sensitive information, like usernames, passwords and credit card details for malicious reasons by mimicking a trustworthy entity, like a well-known and trusted website. It can be carried out by email spoofing, or instant messaging, and generally appears to be from social networking websites, auction sites as well as online payment processing websites. Phishing websites deceive users, and exploit weaknesses of web security technologies. The September 2017 Webroot Data [1] estimates that approximately 1.385 million such phishing websites are created on a monthly basis. Phishing is a difficult problem for social networking websites to tackle, because it relies on tricking users into revealing confidential data, as opposed to finding exploits to gain access to their accounts. Phishing scams often try to convince visitors to release confidential information in various ways, often using social engineering techniques, malvertising, voice messages, SMS and through social media [2], or through the installation of malware on personal devices [3].

It has been demonstrated in the literature that standard security indicators, including browser features such as the padlock icons, etc. are insufficient for protecting users from phishing websites, as visual deception attacks can fool even sophisticated users. This is primarily because even while seeing cues known to be associated with phishing websites, people rely on visual similarities in order to realise whether the website is legitimate or not, as is displayed by conducting experiments on selections of people to check their ability to differentiate phishing websites from real ones [4]. Further research in this area using eye-tracking software has demonstrated that users spend more time looking at web-

✉ Sagnik Anupam
sagnikanupam@gmail.com

Arpan Kumar Kar
arpan_kar@yahoo.co.in

1 DPS RK Puram, New Delhi 110022, India

2 Department of Management Studies, Indian Institute of Technology, New Delhi 110016, India

Published online: 19 November 2020

site content as opposed to browser security indicators when assessing website legitimacy and hence shown that existing browser cues remain insufficient to protect users from phishing websites [5]. Another experiment was conducted to check the response of people when an email was sent to them with information that their bank account had been compromised, with a link which led to a phishing website asking them for their account details [6]. The results of these studies indicate that a significant percentage of unsuspecting people do fall prey to the traps laid by phishing websites, even those who have spent a significant period of time working with computers or working in IT departments. Despite significant improvements in the filtering mechanisms of anti-spam software, many non-executable file attachments often reach users' inboxes, and these can often be exploited to lead users to phishing websites [7]. Thus, there is a need for a machine-learning approach to determine the legitimacy of websites and devise a classifier trained in order to accurately do so, taking in parameters identified in the prior literature as indicators of phishing websites [8].

The objective of the paper is to propose and evaluate an improved machine learning model for phishing detection which comprises of a Support Vector Machine (SVM) optimised by a nature-inspired optimization algorithm. However, in order to determine a suitable nature-inspired optimization algorithm, a comparative study has also been performed between four nature-inspired optimization algorithms, and the results of this study have also been presented, after comparing the various models using different metrics. This paper uses an innovative machine learning approach to build a classifier that classifies websites based on a variety of characteristics using SVMs trained on nature-inspired optimization algorithms. The nature-inspired algorithms were used for the hyperparameter optimization of SVMs. It is also a comparative study which aims to obtain the optimal model combination, amongst the discussed variations. The study was conducted across four nature-inspired optimization algorithms, the Bat Algorithm (BA), the Firefly Algorithm (FA), the Whale Optimization Algorithm (WOA), and the Grey Wolf Optimiser (GWO) algorithm, each of which was used to train an SVM to classify phishing and legitimate websites. We show that while the GWO algorithm yielded the highest mean performance, its performance was only statistically better than that of the FA. The performances of any other pair of algorithms could not be said to show a statistically better performance. However, all four nature-inspired optimization algorithms performed significantly better than the optimized Random Forest classifier which had previously been determined to have achieved the highest accuracy on this dataset [9].

This paper is divided into different sections as follows: Sect. 2 discusses various anti-phishing techniques, including work done on the usage of other machine learning techniques in phishing website detection and classification, as well as methods of optimizing existing machine learning algorithms. Section 3 explains the feature selection, and Sect. 4 explains the computational methodology employed to design the classifier. Section 5 presents the findings of the study, and explains the comparison of the accuracy of the algorithms using various metrics, and lastly, Sect. 6 discusses the conclusions of the study, its practical implications and topics for future research.

2 Literature review

Over the years, numerous phishing website detection techniques have been developed in response to the various kinds of phishing scams designed to trick users into giving away their personal data. Many of the earliest methods relied on linguistic techniques, blacklists, and whitelists. However, many modern approaches have taken a content-based route, electing to analyze features of different websites and their URLs to determine the legitimacy of a website. In this section, we have provided an overview of existing phishing website detection techniques and highlighted innovative approaches for the same.

2.1 Spam filters and digitally signed emails

A popular anti-phishing approach is the use of spam filters to filter out requests that ask for personal information and are from unauthorized or fraudulent sources. Such emails generally contain links to phishing websites imitating genuine websites, and thus spam filters play a pivotal role in protecting users from visiting such unsafe sites [10]. Other approaches to detecting phishing emails include data mining techniques which have achieved 89% accuracy after analysing both harmless and phishing-related emails [11]. Similarly, digitally signed emails are also important mechanisms in verifying whether the emails are authentic or not, and mark unverified requests as potentially dangerous. However, while these are important in preventing the spreading of such links through email and other sources, they fail to identify websites which have not come through these routes. Also, many companies do not use digitally signed emails due to difficulty in implementation. Meanwhile, accounts can always be compromised, and trusted users can be tricked into forwarding links to phishing websites. Spam filters also rely on other information contained within emails, such as the language used, etc. and would be unable to gauge the authenticity of a website based on features of the website alone.

2.2 Blacklisting and whitelisting

Several online tools use blacklisting to determine whether a website is a legitimate website or a phishing website. Blacklisting refers to the compilation of lists of websites which are known phishing websites, and running a check on the target website to see if it is present on the list or not. If present, then it is marked as a phishing website, whereas if absent, it is, as per the blacklist being used, declared as a legitimate website. This approach is used in different web technologies, including PhishTank, Google Safe Browsing, Internet Explorer 9's anti-phishing protection, SiteAdvisor, NetCraft, etc. [12]. This approach essentially turns the scenario into a never-ending race between the security researchers and the phishing website creators, which is a highly tedious and difficult task as each site needs to be reviewed thoroughly before being declared as a phishing website, and added onto a list. Thus new phishing websites are often absent on such lists, and this can be harmful to users.

Whitelisting, on the other hand, prompts users to add verified sites with successful login attempts to a whitelist, and treats all other websites as though they are suspicious websites.

Both blacklists and whitelists, however, can be very easily integrated into web browser extensions, and this technology can be easily used and integrated onto different platforms.

2.3 Methods based on visual similarity

There are alternative methods that have been proposed to use comparisons between the visual profiles of a trusted website and the phishing website imitating this website, as visual similarity is one of the key ways in which phishing websites gain the trust of unsuspecting users. One method includes comparing phishing websites with the "victim" websites, which are being imitated, and also with other phishing websites imitating the same victim website [13].

Another method, PhishZoo, uses the visual profiles of trusted websites and checks how much websites match against these profiles, thus combining both whitelisting and blacklisting methods [14]. It relies on the websites' content to detect their corresponding phishing sites, and can thus be adapted to identify newly created phishing websites which are not yet listed on different blacklists.

A different approach to comparing two websites was in terms of its content and properties, where each website is said to comprise of three elements - the visible text, the visible elements, and the overall look of the website. Features of all three are extracted and are used to generate an overall signature, and the signatures are compared by matching elements pairwise. The comparison yields a similarity score, which is then used to assess the legitimacy of a website [15].

Another approach involves detecting similarity at three levels: a block level to determine similarity in an image or text block, a layout similarity which compares the weighted number of matched blocks to the total number of blocks on the web page, and an overall style similarity which is determined using a histogram of style feature values for the web page [16]. Subsequently, a similarity score is computed, and then websites beyond a predefined threshold are classified as phishing websites.

Another method involves converting the web pages into low-resolution images, and using colour and coordinate features for the representation of the image signatures [17]. The Earth Mover's Distance (EMD) is then used for computing the signature distances of the images of the web pages, and an EMD threshold vector is trained for classifying the web page as a phishing or normal webpage.

These methods, however, operate on the assumption that users can detect phishing websites which appear significantly different from the legitimate websites they try to imitate, and focus on using a vision solution to match images of the legitimate as well as the phishing website. This would not cover phishing websites imitating websites who don't have a profile at the moment, and would require extensive coverage of all the websites which could be potentially imitated, as well as updating the visual profiles regularly in case any of these companies change their website's design.

2.4 Dynamic security skins

Various methods have been proposed which would enable users to detect phishing websites with the aid of browser plugins and other tools. One method includes that of Dynamic Security Skins [18], which involves an authentication procedure between the server and the user using the Secure Remote Password protocol (SRP). Following a successful authentication, customized windows with specific background images are generated which would allow the user to verify that it is indeed a trusted website to which information can be safely given. However, this method has two key drawbacks, the first of which is that it requires modifications on the side of many servers. Secondly, it also requires the user to determine whether the website is a phishing website based on whether the browser displays the dynamic skin of the website they are supposed to be visiting. If the user fails to recognise the website, the mechanism fails. Apart from that, it is also possible that phishing websites may generate their own, false, visual hashes, and if the user fails to spot these counterfeit hashes, they are again at risk of a phishing attack.

2.5 Machine learning techniques

The limitations of the methods mentioned previously have led to machine learning, as well as data mining techniques being used for website classification. A combined approach of fuzzy logic and neural networks has been used to classify phishing websites [19], while a combined fuzzy logic and data mining approach has been used to identify e-banking phishing websites [20]. An Adaptive Neuro-Fuzzy Inference System (ANFIS) model has also been used for phishing website detection [7]. Associative classification mining has been used to classify phishing websites using the Multi-label Classifiers-based Associative Classification (MCAC) algorithm [12]. The Genetic Algorithm (GA), has been used in order to evolve rules for the prevention of phishing attacks [21]. A single-hidden layer feedforward neural network with two neurons has also been trained for classification of phishing websites [22]. SVMs, Naive Bayes Classifiers, K-Means, K-Nearest Neighbours and Affinity Propagation techniques have also been used for malicious website detection [23]. SVMs have been used in the past for phishing website detection [24], taking in a few semantic features, such as the content and description of the web page, along with some other features such as Server Form Handle status, Request URL, etc. which are well established in the literature. Their classifier could obtain an 84% accuracy rate, on a dataset comprising of 279 phishing websites and 100 legitimate websites.

Random Forest classifiers have also been used as a machine-learning methodology in the classification of phishing websites. One paper compared the performance of linear and non-linear SVMs alongside that of Random Forest classifiers in order to evaluate their performance on classifying phishing websites on the same publicly available UCI phishing website dataset as our paper [9]. Their Dataset II had been divided in the 70:30 ratio and the performance of linear SVMs, non-linear SVMs and Random Forest classifiers on this dataset was compared. The hyperparameter tuning in their model was conducted using a grid-search algorithm. However, their results show that their Linear SVM model displayed a classification accuracy of 85.19% on their test set, while their non-linear SVM model displayed an accuracy of 89.63% and their Random Forest classifier displayed an accuracy of 90.12%. Another approach presented a comparison between several machine-learning methodologies, such as AdaBoost, neural networks, SVMs, Bayesian additive regression trees, random forests and others [25]. They tested these methodologies on a dataset comprising of 8 features used by CANTINA, the state-of-the-art technique for detection of phishing websites which does not use a blacklist, instead computing the likelihood of a website being a phishing website using a weighted majority technique, without any machine learning algorithms. They trained the model on a dataset comprising of 1500 phishing sites and 1500 legitimate sites. Their research also demonstrated that the SVM model, with a 14.29% error rate and 0.8562 F1 score, performed better than Random Forest model, which obtained a 14.45% error rate and 0.8554 F1 score on the basis of these two metrics. They also demonstrated that the CANTINA model obtained an F1 score of 0.7607, and an error rate of 20.52%. On the other hand, both the CANTINA model and the Random Forest model outperformed the SVM model on the basis of the Area Under the Curve (AUC) metric. While these results do make SVMs an interesting aspect of machine learning to be evaluated in the field of phishing websites, this methodology does not mention hyperparameter tuning using advanced, nature-inspired optimization algorithms. Developments in the field of SVM optimization indicate that it is now necessary to re-evaluate the usage of SVMs in phishing website detection to determine if better-optimized SVMs can improve on existing results and outperform other machine-learning models.

3 Features utilised

This paper uses a content-based approach for phishing website detection, and was trained on a binarized version of the existing public Website Phishing Dataset [12] available on the UCI Machine Learning Repository [26]. The binarized version identifies suspicious or phishing websites separately from legitimate ones. The features were selected from the dataset for training the model have been described in Table 1. In Table 1, features have been named and described under the Features and Feature Description columns. Their Feature Value, or x, denotes the distinct class that a website can belong to for a given feature. It should be noted that as we are not assigning any rules in our model to denote that a particular feature value is inherently suspicious or legitimate, as we prefer that our machine-learning model make these associations on its own. As such there is no universal significance of the numbers we have assigned to different classes across different features, and the assignment of numbers is only for the purpose of marking a class as distinct from others.

4 Computational methodology

This section is used to introduce the algorithms and the hybrid models used in the creation of the classifier and also to explain why these methodologies were chosen for usage in phishing website classification. Section 4.1 explains the working of SVMs, and the non-linear RBF kernel variant that was chosen for the classifier, and Sect. 4.2 introduces nature-inspired optimization algorithms, and the four algorithms used in the study. Each algorithm was used to find optimal hyperparameters for the SVM, and thus used to create a hybrid classification model.

Table 1 Description of selected features

Features	Feature description	Feature value (x)
Presence of IP address in the domain name	This is generally an indicator of attempts to collect personal data, as many phishing websites employ this trick. All the websites employing this trick present in the dataset are associated with phishing websites.	$x = \begin{cases} 0, & \text{if IP address is present} \\ 1, & \text{otherwise} \end{cases}$
Age of domain	Websites that have been online for less than a year could be risky, as opposed to long established legitimate websites. Websites have been classified into two classes, based on whether their age was less than or equal to 6 months, or whether it was more than six months. Most short-lived phishing websites would thus be placed in the first class.	$x = \begin{cases} 1, & \text{if age } \leq \text{ 6 months} \\ -1, & \text{otherwise} \end{cases}$
URL length	Phishing websites often hide suspicious parts of their URLs at the end of long URLs which redirect the information submitted by users or redirect the web page itself. A length of 54 characters has been suggested in previous research [27] to separate phishing websites from legitimate ones.	$x = \begin{cases} 1, & \text{if URL Length } < 54 \\ -1, & \text{if } 54 \leq \text{ URL Length } \leq 75 \\ 0, & \text{otherwise} \end{cases}$
Web traffic	Legitimate websites often have high traffic as they are visited regularly, but most short-lived phishing websites are low-ranked and do not attract many visitors. The Alexa Ranks of websits have been used to evaluate this parameter, which are computed taking into consideration daily unique visitors and page views over a time period of 3 months [28]. Legitimate websites generally have ranks less than or equal to 150,000 [12].	$x = \begin{cases} 1, & \text{if Alexa Rank } \leq 150,000 \\ 0, & \text{if Alexa Rank } > 150,000 \\ -1, & \text{otherwise (no data available)} \end{cases}$
URL anchor	This feature refers to the percentage of links within a webpage that point to a different domain name than the one in the address bar.	$x = \begin{cases} -1, & \text{if URL Anchor } < 31\% \\ 0, & \text{if } 31\% \leq \text{ URL Anchor } \leq 67\% \\ 1, & \text{otherwise} \end{cases}$
Request URL	This feature caculates the percentage of obects, such as videos, and images are loaded from a domain other than the one typed in the URL address bar.	$x = \begin{cases} -1, & \text{if Request URL } < 22\% \\ 0, & \text{if } 22\% \leq \text{ Request URL } \leq 61\% \\ 1, & \text{otherwise} \end{cases}$
SSL final state	SSL (Secure Sockets Layer) is amongst the most widely used security protocols today. SSL certificates display indicators on typing an address into the status bar, such as the usage of the HTTPS Protocol, which is more secure and uses port 443 instead of port 80 used by HTTP Protocol. However, many phishing websites use fake HTTPS protocols to deceive users, and this parameter checks if the protocol is offered by a trusted issuer or not. [12]	$x = \begin{cases} 1, & \text{if uses HTTPS \& trusted issuer \& age } \geq \text{ 2 years} \\ -1, & \text{if uses HTTPS \& issuer is not trusted} \\ 0, & \text{otherwise} \end{cases}$
Pop-up window	Authenticated websites do not ask users to enter confidential information in the form of pop up windows.	$x = \begin{cases} -1, & \text{if right click is disabled} \\ 0, & \text{if right click shows alert} \\ 1, & \text{otherwise} \end{cases}$
Server form handle (SFH)	When information is entered into a legitimate website, the information is processed from the same domain from where the website is loaded. Phishers resort to either transferring data to a different domain or leave the server form handle empty, and both of these options are identified separately from the one used by legitimate websites.[12]	$x = \begin{cases} 1, & \text{if blank or empty} \\ -1, & \text{if redirects to a different domain} \\ 0, & \text{otherwise (information is processed from same domain)} \end{cases}$

4.1 Support vector machines

Support Vector Machines (SVMs) are a supervised learning model which can be used for binary classification of data into two classes. SVMs have several benefits when it comes to solving non-linear classification problems, through their use of a kernel function. No assumptions are made about the nature of the data, as the kernel itself contains a non-linear transform, and as a consequence provides a robust model. They also deliver a unique solution, as the optimality problem is convex when compared to models such as neural networks, which lack robustness over different samples as there are multiple solutions associated with local optima [29]. Moreover, SVMs have been used effectively in the past in classification problems such as those of classifying phishing websites [24] as well as review spammer detecton [30], and hence provide an excellent starting point for improvement using nature-inspired optimization algorithms.

The two classes into which the websites are classified are denoted by $y_i \in \{1, -1\} \ \forall \ x_i$. The training dataset contains all the x_i samples with corresponding y_i values which denote the class to which the data belongs. The SVM plots the data onto a graph, and tries to find an optimal hyperplane which can divide the graph into two regions such that the entire positive class lies in one region, whereas the negative class lies in the other region. Predictions of future values can be made by checking the position of the new data, x_j on the graph, and returning a positive or negative value through a sign function on its position.

However, it is rarely possible that a binary classification problem can be solved linearly, i.e. in such a manner that a d-dimensional graph can be separated by a $d-1$ dimensional hyperplane. Thus, kernel functions are used in order to transpose the data to a higher dimension. In case it is still not separable after the usage of the kernel function, the objective becomes to minimize the error rate of the SVM.

Thus given a vector X consisting of input vectors $x_i \in \mathbb{R}^p$ for $i = 1$ to n, where $p =$ no. of input features, and a vector $Y \in \{-1, 1\}^n$, consisting of classes y_i, where $n =$ no. of input vectors, the SVM solves the following problem:

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to } y_i(w^T x_i + b) \geq 1 - \xi_i, \ \ i = 1, \ldots, n \\ &\qquad \xi_i \geq 0, \ \ i = 1, \ldots, n \end{aligned} \tag{1}$$

where the hyperplane is written as $y = w^T X + b$, w is a vector and b is a scalar quantity [31].

C is a parameter which determines the amount by which each misclassification is penalized, and determines the bias-variance tradeoff of the model. Lower values of C increase the bias of the model, whereas larger values increase its variance.

ξ_i are positive slack variables for regularization, to prevent the hyperplane from overfitting the dataset.

Its dual problem can be obtained by forming the Lagrangian equation of the problem, and substituting the values using the Karush-Kuhn-Tucker conditions, one obtains the following problem:

$$\begin{aligned} &\max_{\alpha} W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\ &\text{subject to } 0 \leq \alpha_i \leq C, i = 1, \ldots, n \\ &\qquad \sum_{i=1}^{n}\alpha_i y_i = 0 \end{aligned} \tag{2}$$

where α is a vector consisting of all the Lagrangian multipliers α_i, and $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel function of the feature mapping ϕ, which maps from an input x to its features in a higher dimension, $\phi(x)$.

Equivalently, we can express the dual in the form of vectors as:

$$\begin{aligned} &\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ &\text{subject to } 0 \leq \alpha_i \leq C, i = 1, \ldots, n \\ &\qquad Y^T\alpha = 0 \end{aligned} \tag{3}$$

where $Q_{ij} = y_i y_j K(x_i, x_j)$ and e is a vector of ones [32].

The Scikit Learn library's sklearn.svm.svc class was used in Python for building the SVM model. This library uses the Sequential Minimal Optimization (SMO) Algorithm to solve the dual problem, because of the computational efficiency in solving the large quadratic programming (QP) problem by breaking it into a series of smallest possible QP Problems, which are then solved analytically [33]. The RBF (Radial Basis Function) kernel was used as the kernel function for the mapping of data to higher dimensions. The RBF kernel function has become the choice of several researchers in different fields, due to its accuracy and reliable performance [34]. The RBF kernel function is defined as [35]:

$$K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2) \tag{4}$$

The sklearn.svm.svc class used has an inbuilt predict function for the prediction of the class of the test data, x, which is used to find the side of the hyperplane on which the data point lies. It is defined as [36]:

$$f(x) = sgn\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + b\right) \tag{5}$$

where $sgn(z)$ is the sign function, i.e. it returns whether z is positive or negative.

Thus, for the construction of the SVM model, it takes in two parameters: C and γ, which can be optimized using optimization algorithms [37]. Four nature-inspired optimization algorithms were used to obtain the optimal values for these parameters.

4.2 Nature-inspired optimization algorithms

The domain of bio-inspired computing has seen a rise in interest in current times, with various studies exploring their applications, and others highlighting improvements in the field [38]. Application of bio-inspired algorithms are necessary for addressing complex, real-world problems [39]. Nature-inspired optimization algorithms are generally used for solving complex optimization problems, using inspiration from natural processes [40]. The Particle Swarm Optimization (PSO) algorithm creates a particle swarm where the collective intelligence of the group is more than an individual's intelligence, which improves the efficiency of the search for an optimal solution [41]. A regression-based variant of the Particle Swarm Optimization has been used for the optimal feature selection problem [42]. The Artificial Bee Colony (ABC) optimization algorithm uses a model of bees hunting for food sources in a swarm, and communicating with each other in order to compute the optimal solution [43]. The Cuckoo Search (CS) algorithm models the brood parasitism of cuckoos, when they lay eggs in the nests of other birds, and uses a method of ranking the best nests in order to find the optimal solution [44].

There are several different machine learning techniques available which can be used for classification and regression problems, and nature-inspired optimization algorithms have been used to find optimal solutions in several cases because of their ability to converge towards such solutions in multi-dimensional hyperplanes [39]. These optimization algorithms have been used to optimize the performance of SVMs in many different applications, by generating hybrid models. Out of these nature-inspired optimization algorithms, four Swarm Intelligence (SI) algorithms were selected for comparison of their performance on classifying phishing websites. These algorithms take into account the collective behaviour of the swarm of animals as they fulfil the requirements of the optimization problem [45]. The four algorithms taken into account are the Bat Algorithm (BA), the Firefly Algorithm (FA), the Whale Optimization Algorithm (WOA), and the Grey Wolf Optimiser (GWO) algorithm. These were chosen due to superiority of the BA in obtaining required accuracy values faster than other standard bio-computing algorithms [46], the capability of multimodal optimization of the FA [47], and the competitive performance of the WOA [48] and the GWO algorithm [49] when compared against standard bio-computing algorithms like Genetic Algorithms and the Particle Swarm Optimization algorithm on benchmark functions. Furthermore, hybrid SVM-FA models have been used for predicting global solar radiation [50] and malaria transmission [34], while the BA [51] and the GWO algorithm [52] have been used in SVM hyperparameter optimization. The WOA has also been used to tune SVM hyperparameters for detecting spam profiles on social networks [53].

4.2.1 Bat algorithm

This algorithm is inspired by the echolocation of microbats. Microbats produce a loud sound pulse, and listen for the echo that bounces back from the neighbouring objects. These pulses vary in their properties across species and often correlate to their hunting strategies. Using some of these properties, one can approximate these rules to form the BA [46]. The rules governing the BA are:

1. All the bats use echolocation to sense the distances. They can differentiate between the food, the prey and the background barriers.
2. Bats fly randomly, with velocity v_i, at position x_i, and with a frequency f_i. They can vary wavelength λ and loudness A_0 to search for prey, and automatically adjust the wavelength (or frequency) of the pulses they emit. They can also adjust the rate of pulse emission $r \in [0, 1]$.
3. While the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

For all the virtual bats which are being simulated, we need to specify rules for how their positions x_i and velocities v_i in a d -dimensional space are being updated. These are updated by the following equations [46]:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{6}$$

$$v_i^{t+1} = v_i^t + (x_i^t - x_*) f_i \tag{7}$$

$$x_i^{t+1} = x_i^t + v_i^t \tag{8}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution, and x_* is the current optimum solution, which is chosen after comparing all of the solutions amongst all of the bats in the current iteration. Initially, each bat is randomly assigned a frequency value that is drawn uniformly from $[f_{min}, f_{max}]$.

4.2.2 Firefly algorithm

Many fireflies produce short, rhythmic flashes of light, and their pattern of flashes is often unique to certain species. The main functions of these flashes are to attract mates, and also potential prey. The light intensity from these flashes follows the inverse-square rule, i.e. the light intensity i decreases with an increase in distance r in the form of $I \propto 1/r^2$. The air also absorbs the light, and because of these two reasons, fireflies are restricted in their communication to a few hundred metres at night [47]. The rules governing the FA are:

1. All the fireflies are taken to be unisex, so one firefly will be attracted to another regardless of sex.
2. Attractiveness is proportional to the brightness of a firefly. While comparing two fireflies, the less bright one will move towards the brighter one. If no such candidates are available for a particular firefly, then it will move randomly.
3. The firefly's brightness is affected by the landscape of its objective function.

Now, the light intensity I(r) varies according to the inverse-square law as

$$I(r) = \frac{I_s}{r^2} \tag{9}$$

where I_s is the light intensity at the source. For a given medium, with a fixed coefficient of light absorption (γ), the combined effect of the inverse square law and absorption can be approximated to show that the light intensity I varies with distance r as follows:

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

where I_0 is the original light intensity.

As a firefly's attractiveness is proportional to the light intensity seen by the adjacent fireflies, the attractiveness β of a firefly is defined as:

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

where β_0 is the attractiveness at $r = 0$.

The distance between any two fireflies at x_i and x_j is the Cartesian distance r_{ij},

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \tag{12}$$

where $x_{i,k}$ is the k^{th} component of a d-dimensional vector x_i. The final movement of a firefly(i), towards a more attractive firefly(j) is determined as:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha\epsilon_i^t \tag{13}$$

where the second term is due to the attraction, and the third term due to randomization, with α as the randomization parameter and ϵ_i^t is a vector of random numbers drawn from a uniform distribution. Another improvement which has been used is to vary the randomization parameter α so that it decreases gradually as the optima are approaching, which improves the convergence of the algorithm [47].

4.2.3 Whale optimization algorithm

Humpback whales have a special, unique, hunting method, which is known as the bubblenet method. This method involves the creation of bubbles in a spiral shape around the prey, and swimming upwards towards the surface. It is a unique method that is observed in humpback whales, and it has been mathematically modelled in order to solve optimization problems [48].

As humpback whales can detect the location of prey and encircle them, this optimization algorithm assumes that the current best solution is close to the optimum. Once the current optimum solution is identified, the other search agents, or the whales, update their positions towards the current best solution.

$$D = |C \cdot X^*(t) - X(t)| \tag{14}$$

$$X(t+1) = X^*(t) - A \cdot D \tag{15}$$

where t indicates the current iteration, A and C are coefficient vectors, X^* is the position vector of the best solution obtained so far, X is the position vector. A and C are defined as in the following equations:

$$A = 2a \cdot r - a \tag{16}$$

$$C = 2 \cdot r \tag{17}$$

where a decreases linearly from 2 to 0 over the course of iterations and r is a random vector in [0, 1].

In order to mathematically model the bubblenet attacking method of the humpback whales, two approaches have been designed:

1. The Shrinking Encircling Mechanism: It shrinks the circle in which the whales are swimming downwards, and the new value of X lies anywhere in between the original position and the current optimum X^*. This is achieved by decreasing the value of a.
2. The Spiral Updating Position: This approach calculates the distance between the whale and the prey, and creates a spiral equation to imitate the helix-shaped movement of the humpback whales.

$$X(t+1) = D' \cdot e^{bl} \cdot cos(2\pi l) + X^*(t) \tag{18}$$

where

$$D' = |X^*(t) - X(t)| \tag{19}$$

D' indicates the distance of the i^{th} whale to the best solution obtained so far, b is a constant for defining the shape of the logarithmic spiral and l is a random number in $[-1, 1]$.

As humpback whales simultaneously swim in spirals as well as in shrinking circles, the algorithm assumes that there is a 50% chance of either of the mechanisms occurring during optimization. The final model obtained is:

$$X(t+1) = \begin{cases} X^*(t) - A \cdot D, & \text{if } p < 0.5 \\ D' \cdot e^{bl} \cdot cos(2\pi l) + X^*(t) & \text{if } p \geq 0.5 \end{cases} \tag{20}$$

where p is a random number in [0, 1] [48].

4.2.4 Grey Wolf optimiser algorithm

Grey wolves are pack animals and apex predators, and have a strict, dominant social hierarchy. The leaders of the pack are a male and a female, known as the alphas, and they are mostly responsible for making decisions in a pack. The betas are the subordinates of the alphas, and respect the alpha wolves. However, they command other lower level wolves as well. The beta reinforces the commands of the alpha, and gives feedback to the alpha as well. The omegas are the lowest ranking wolves in the pack, and follow the directions of all the wolves hierarchically senior to them. They are also the last wolves allowed to eat, and also act as babysitters at times. Another category is the delta wolf, which does not belong to the alpha, beta or omega. These wolves submit to alphas and betas, but can dominate over the omegas. Scouts, sentinels, hunters, elders and caretakers generally are delta wolves [49].

The hunting method of a pack of grey wolves, which serves as the source of inspiration for the algorithm, has three phases:

1. Tracking, chasing and approaching the prey.
2. Harassing, encircling, and pursuing the prey.
3. Attacking the prey when it stops moving.

While searching for the optimum solution, designate the fittest solution, at any given point of time t as the alpha (α), and the second and third best solutions as the beta (β) and the delta (δ) respectively. The rest of the solutions are assumed to be the omega (ω) wolves. The following equations are proposed to simulate the encircling movement of the wolves:

$$D = |C \cdot X_p(t) - X(t)| \tag{21}$$
$$X(t+1) = X_p(t) - A \cdot D \tag{22}$$

where $X(t)$ indicates the current iteration, A and C are coefficient vectors, X_p is the position vector of the prey and X indicates the position vector of a grey wolf.

$$A = 2a \cdot r_1 - a \tag{23}$$
$$C = 2 \cdot r_2 \tag{24}$$

where r_1 and r_2 are both random vectors in [0, 1] and a decreases linearly from 2 to 0.

The hunting in the real world is generally guided by the alpha, with the beta and delta occasionally participating. However, as the location of the prey (or optimum solution) is unknown in the search space, it is assumed that the alpha, beta and delta know the most about the location of the prey, and the rest of the pack are programmed to follow their lead. So we take the most optimum solution in the current iteration as the alpha, the second most as the beta and third most as the delta.

$$D_\alpha = |C_1 \cdot X_\alpha - X| \tag{25}$$
$$X_1 = X_\alpha - A_1 \cdot D_\alpha \tag{26}$$

Similarly, we define $D_\beta = |C_2 \cdot X_\beta - X|$ and $D_\delta = |C_3 \cdot X_\delta - X|$, and $X_2 = X_\beta - A_2 \cdot D_\beta$, and $X_3 = X_\delta - A_3 \cdot D_\delta$. Finally, we define $X(t+1)$ as:

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{27}$$

Continuously running these updates for a maximum number of iterations will eventually converge upon the optimum solution [49].

5 Findings

5.1 Dataset

All the features were taken from a public dataset, hosted on the UCI Machine Learning Repository [26]. The creators of this dataset had classified all the features taken from the website into various classes, and we trained our model on these classes without any interaction with the original websites [12]. Most of these features, like the pop-up window feature, or the SFH status, are static features of the website measured while visiting the site, whereas others, such as the age of the domain, seem to have been measured during the period of data collection. All the features provided in the dataset have been used for training our models. The original public dataset classified samples into three classes: legitimate, suspicious and phishing. In order to convert it into a binary classification problem, the data was first relabelled into two classes: phishing or suspicious websites, and legitimate websites. The source of the original dataset mentions PhishTank as a source for the phishing and suspicious websites, and states that the legitimate websites were collected from Yahoo and starting point directories [26]. The division of the websites in the original database was 548 legitimate websites, 702 phishing websites, and 103 suspicious websites [12]. After conversion into a binary classification problem, the final dataset used containing 805 websites in the category of phishing and suspicious websites, and 548 in the category of legitimate websites.

The dataset, which contained 1353 samples, was divided into training and test datasets of approximately 8:2 ratio, i.e. the training datasets had 1082 samples, whereas the test datasets had 271 samples. Three such training and test dataset pairs were created out of the dataset, and the performance of each optimization algorithm was tested on the three test datasets for 100 iterations, keeping in line with previous comparative studies [46]. For our study, if the training dataset's size was too small, dimensionality reduction techniques would have to be employed to reduce the number of variables being considered, which would be relatively large [54]. Hence, we selected a training set size that would be more than 1000, which would provide an ideal number of dataset samples for training, and would also provide a test set of an appropriate size. As this was achievable with an 8:2 ratio between the training and test sets, this ratio was employed for the creation of the datasets. The performance of each algorithm was validated five times to account for fluctuation in the value of random coefficients, and its mean performance was reported in terms of classification accuracy, and F1 scores. Prior literature has established that multiple validations are necessary across different parameters for improvement of results [55], and a minimum of five validations are recommended for the verification of results [56–60].

The trials were conducted on a Lenovo Yoga 500 with an Intel Core i7 5th Gen processor having a speed of 2.4 GHz, and 8GB DDR3 RAM. The divisions of classification data in the three training-test pairs of datasets have been described in Table 2.

5.2 Performance of algorithms

The performance of each of the algorithms in finding the optimum parameters for the SVMs has been shown in the following tables, which list the mean and standard deviation of the classification accuracy and F1 scores of the SVM models using the optimal values found for the SVM parameters C and RBF-kernel function parameter γ.

Finally, the mean classification accuracy and mean F1 score of the four nature-inspired optimization algorithms across the five validations have been shown, along with their standard deviations, in Tables 3 and 4 for a comparative study. The mean and standard deviation of the five validations have been used to show the performance of the algorithms in each dataset, and the mean and standard deviations across all three datasets have been taken to show the overall performance of the four nature-inspired optimization algorithms.

One of the approaches described in existing literature, comparing the performance of grid-search optimized SVMs and Random Forest (RF) classifiers has used the same dataset as ours. Hence, we are presenting their results alongside ours as a comparison with existing research. Their results show that their Linear SVM model displayed a classification accuracy of 85.19% on their test set, while their non-linear SVM model displayed an accuracy of 89.63% and their Random Forest classifier displayed an accuracy of 90.12% [9]. However, as we do not know the manner in which they divided this common dataset into training and testing datasets, we cannot directly compare their results to ours. In order to prove the superiority of nature-inspired optimization algorithms-optimized SVM models, we have compared the performance of our four algorithms with the grid-search optimized RF classifier that they had determined would give the best performance. According to their research, they had determined the optimal hyperparameter for the number of variables randomly sampled as candidates at each split of the RF classifier to be 5 for our dataset using grid-search optimization [9]. In Tables 3 and 4, we have also shown the accuracy and F1 score of an RF classifier which has been trained on the same train-test divisions as our other algorithms using the hyperparameter value determined via grid-search optimization in previous literature.

The convergence plots of the four nature-inspired optimization algorithms on the first dataset are shown in Fig. 1, with the value of the objective function (the error rate of the classifier) plotted against the number of iterations.

These plots show the least error rate obtained so far at each iteration, amongst all the agents. The remaining agents are optimized with respect to the values of the SVM hyperparameters that give this least error rate, until one of them obtains an error rate which is lesser than the current least value. Until then, they are optimized with respect to the least error rate obtained so far, and thus can be "trapped" for a number of iterations within this local optima, which is indicated by the plots. While the plots show that algorithms took different numbers of iterations to reach the global optima, the limitation of the plots is that they do not indicate the number of times the algorithms obtained the optimum value during validations. For example, while in these graphs the WOA obtains its minimum value faster than the GWO algorithm, the GWO algorithm obtained better values for mean classi-

Table 2 Class divisions in training and testing dataset

	Phishing/suspicious websites	Legitimate websites	Total
Training set 1	654	427	1082
Training set 2	647	435	1082
Training set 3	646	436	1082
Test set 1	150	121	271
Test set 2	158	113	271
Test set 3	159	112	271

Table 3 Classification accuracy: mean and standard deviation

	Test set 1		Test set 2		Test set 3		All test sets	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
BA	92.99	0.0000	87.53	0.4042	90.11	0.1650	90.21	2.3207
FA	92.69	0.4811	87.31	0.2021	90.11	0.1650	90.04	2.2960
WOA	92.62	0.4519	87.45	0.3690	90.26	0.2021	90.11	2.2105
GWO	92.99	0.0000	87.82	0.0000	90.33	0.1650	90.38	2.1851
RF	91.37	0.8084	86.13	0.4207	89.30	0.3690	88.93	2.2917

Table 4 F1 score: mean and standard deviation

	Test set 1		Test set 2		Test set 3		All test sets	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
BA	0.9183	0.0003	0.8462	0.0042	0.8797	0.0018	0.8814	0.0306
FA	0.9150	0.0054	0.8410	0.0021	0.8791	0.0022	0.8784	0.0314
WOA	0.9142	0.0058	0.8449	0.0041	0.8813	0.0022	0.8801	0.0296
GWO	0.9182	0.0004	0.8493	0.0000	0.8821	0.0018	0.8832	0.0291
RF	0.9021	0.0088	0.8278	0.0051	0.8695	0.0042	0.8665	0.0320

fication accuracy across multiple iterations than the WOA, though not enough to claim significant results.

Previous research has shown that it is difficult to use time-complexity analysis to evaluate nature-inspired optimization algorithms [61]. However, the number of times the cost function was called by the nature-inspired optimization algorithm has been suggested in the past as a suitable metric to compare the performance of different nature-inspired optimization algorithms. We kept the maximum number of iterations as 100 and the number of agents utilised to find optimum values as 40 and calculated the number of times the cost function was evaluated during execution. This comparison has been shown in Fig. 2. While the FA, the GWO, and the WOA called the cost function the same number of times, the BA required a slightly higher number of calls.

5.3 Discussions

After the algorithms' performance had been measured, statistical tests were conducted to determine whether the performance of the algorithms was significant using a one-way repeated measures Analysis of Variance (ANOVA) test to determine the impact of the choice of algorithms on the classification accuracies and F1 scores. Subsequently, it was found that while the difference in the performance of the algorithms was significant, it could only be said that GWO algorithm performed significantly better than the FA, and the difference in performance among the other algorithms could not be determined to be significant.

5.3.1 Theoretical contributions

Two one-way repeated measures ANOVA tests were conducted. The first test compared the effect of the use of the different algorithms on the classification accuracy across all the trials run across all the three datasets. The second test compared the effect of the use of the algorithms on the F1 score across all the trials run across all the three datasets.

In the first one-way repeated measures ANOVA test (for the classification accuracy), Mauchly's test returned a p-value of 0.044, i.e. $p < 0.05$. Hence the assumption of sphericity was violated. But as the Greenhouse-Geisser

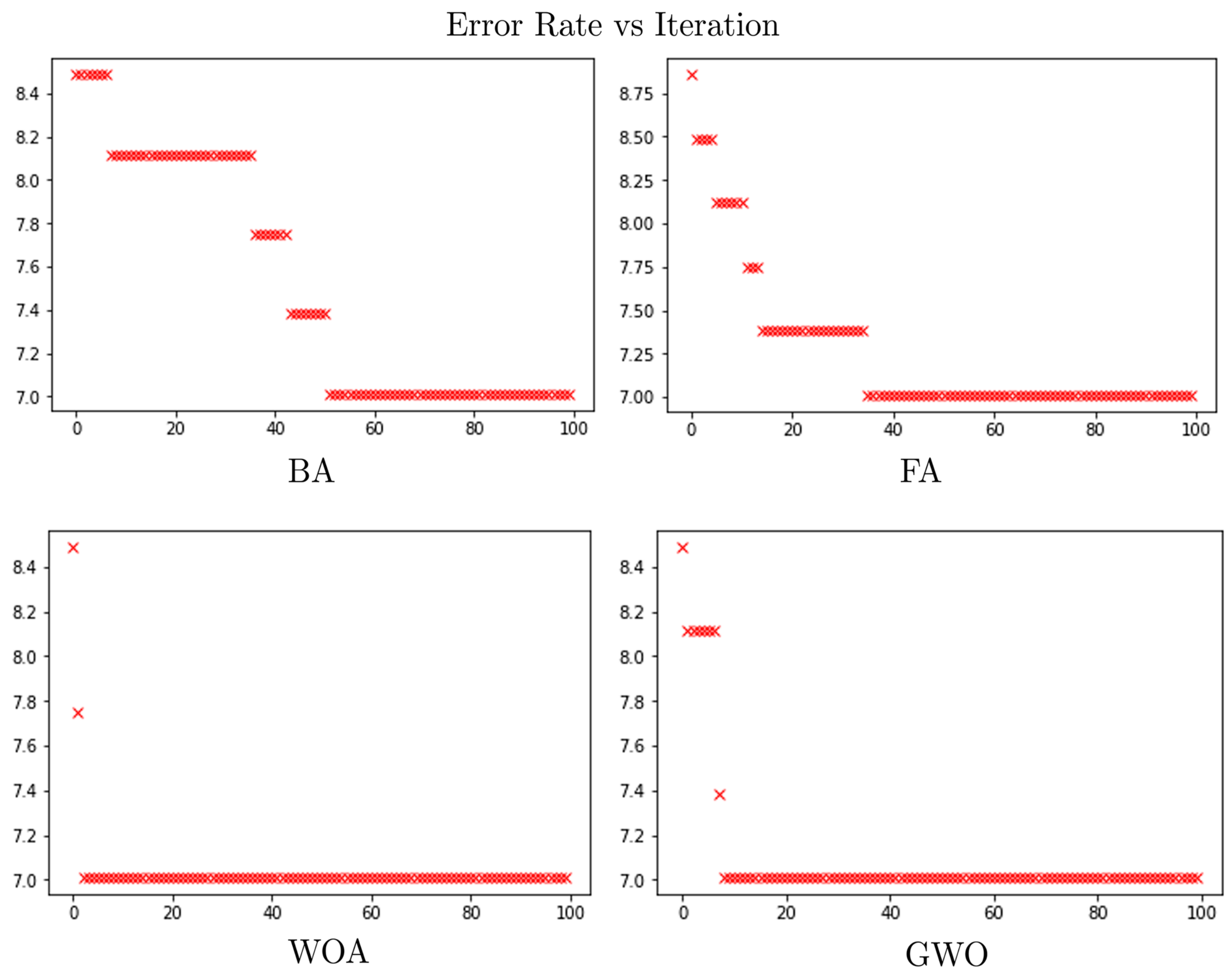
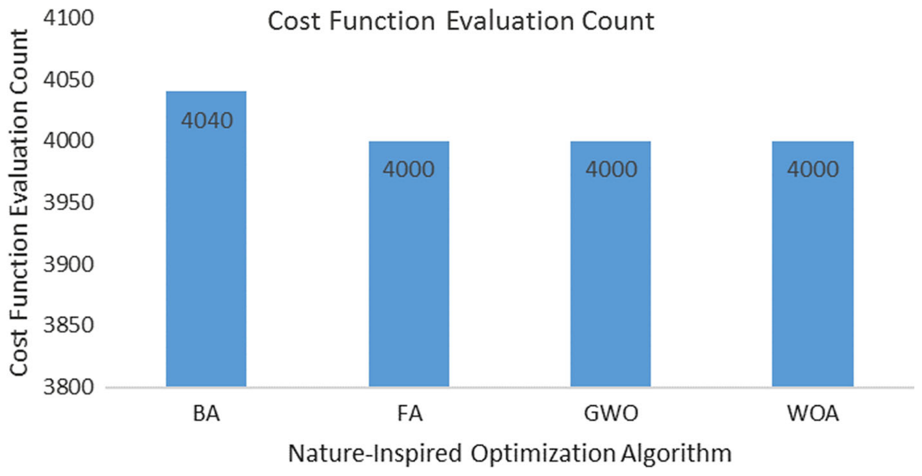


Fig. 1 Convergence plots

Fig. 2 Cost function evaluation count versus nature-inspired optimization algorithm



epsilon value was 0.629, which is less than 0.75, we used the Greenhouse-Geisser corrected results. The results of the test show that there were statistically significant effects of the choice of the algorithm on the classification accuracy, as $F(2.515, 35.212) = 37.349,\ p < 0.05,\ \eta^2 = 0.727$. Hence we can reject the null hypothesis, and claim that the effect of the choice of algorithm on the classification accuracy was significant.

In the second one-way repeated measures ANOVA test (for the F1 score), Mauchly's test returned a p-value of 0.083, i.e. $p > 0.05$. Hence the assumption of sphericity was not violated. The results of the test show that there were statistically significant effects of the choice of the algorithm on the F1 score, as $F(4, 56) = 37.432,\ p < 0.05,\ \eta^2 = 0.728$. Hence we can reject the null hypothesis, and claim that the effect of the choice of the algorithm on the F1 score was significant as well.

Post-hoc tests displayed interesting results, showing that each of the hybrid SVM-nature-inspired optimization algorithm models performed significantly better in terms of both accuracy as well as F1 score than the grid-search optimized RF classifier described in earlier literature. Furthermore, it was determined that that the differences in the performance of the GWO algorithm, and the FA were significant in terms of both the classification accuracy, and the F1 scores. The differences in performances between any other pair of nature-inspired optimization algorithms, however, could not be concluded as significant.

For our final model, there were no weights assigned to the features by us during preliminary data analysis. Consequently, as we were using a non-linear SVM with the RBF kernel function, we were transposing our features into an infinite-dimensional space, and computing the dot product rather than explicitly computing the values of the feature space for each feature in each data point. This is one of the advantages of using a non-linear SVM, as its data transposition into higher dimensions allows the creation of non-linear functions for aiding classification. As a result, there is no possible way to explicitly compute the weights assigned to the features in the infinite dimensional space after using the kernel function, and hence we could not determine which features were prioritised by the model in its classification. This was a limitation of the approach used in these models.

5.3.2 Practical implications

Apart from taxonomical classification on the basis of their inspiration and their behaviour [62], nature-inspired optimization algorithms have also been categorised under different classes in earlier literature based on the amount of existing work and applications developed using the algorithm [39]. Algorithms have been classified into four quadrants, defined as the zones of theory development, applications, rediscovery and commercialization. Among the algorithms used, the GWO algorithm has been classified into the zone of theory development, while the BA and the FA have been classified into the zone of applications.

Algorithms classified in the zone of theory development have a lot of scope for developments to the algorithm itself, as well as for conducting comparative studies between these algorithms in order to gauge the efficiency of these algorithms in different fields. On the other hand, algorithms classified in the zone of applications are considered to be mature in terms of their theory development. However, they need to be applied in engineering and business concepts in order to determine their efficiency for practical problems [39].

Other recent reviews that have evaluated different nature-inspired optimization algorithms on benchmark functions have shown that most of these algorithms show excellent performance in certain hard problems but not others [61]. Hence, more comparative studies are required to highlight the suitability of these algorithms in different domain-specific applications and to provide directions for future research perspectives in the domain [63].

By comparing the performance of the three algorithms in these two zones, along with a relatively newer algorithm, (the WOA) in building hybrid models, this study satisfies some of the objectives outlined for these algorithms in previous literature, by providing insights into how these models can be used in practice based on their accuracy and runtime in a new application. Moreover, through this paper, we have also proved the superiority of SVM models whose hyperparameters were tuned via nature-inspired optimization algorithms over the grid-search optimized RF classifier which had previously been shown to have the best performance on this dataset. Our results are hence also interesting for researchers aiming to determine whether models using nature-inspired optimization algorithms can perform significantly better than grid-search optimized models. Finally, a comparative study, which includes significance tests being conducted to determine the significance of the difference in their performance can help evaluate the suitability of these algorithms in this field, and for future engineering applications.

6 Conclusions

This paper shows that the Grey Wolf Optimiser Algorithm performed significantly better than the Firefly Algorithm, and its considerable accuracy in terms of both mean classification accuracy and mean F1 score, along with its low standard deviation, shows it to be a viable way to optimize the parameters of SVMs to classify phishing websites. While there was no significant difference in the performance of any other pair of optimization algorithms, the overall approach of using nature-inspired optimization algorithms to find optimal parameters for SVMs yielded high accuracy rates. An important role in achieving high accuracy rates is played by the optimization algorithms, which ensure optimal hyperparameters are selected while designing the machine-learning model. This study also follows the recommendations of various review papers in the nature-inspired optimization algorithms literature by evaluating the performance of multiple such algorithms in a comparative study to determine their suitability for the purpose of phishing website detection. By significantly outperforming the grid-search optimized RF classifier that had been shown to provide the best results while classifying phishing websites from this dataset, our work has also highlighted the importance of exploring applications of nature-inspired optimization algorithms in this field.

This approach has several practical implications, as it can be utilised by antivirus software, apps and browser extensions for users to verify the authenticity of websites which ask for personal information. This model could also be implemented as a classifier in real time, protecting users from phishing by parsing websites and feeding the necessary inputs into the model for immediate predictions. The model could also be used as an additional layer of security to spam filters, by scanning websites whose links are forwarded in the emails. As a tool, the model would also be helpful for web hosting services, cybersecurity firms, and other interested organizations who would require accurate classifiers for detecting phishing websites.

Further research would involve the addition of other parameters as well for classification, and expansion in the number of samples used. Additional parameters and constraints can also be applied to extend this classification to other malicious or unsecured websites which might prompt users to download malware, install unwanted browser extensions automatically, etc., thus reducing the strain on manual checking and need for human verification as well.

Acknowledgements We would like to thank Dr Reema Aswani, Research Associate at Indian Institute of Technology, New Delhi, for her support and guidance. We would also like to thank the organisers and the Scientific Review Committee of the Intel IRIS Science Fair, for their critical reviews, and the organisers of the Intel International Science and Engineering Fair, where this project was initially presented.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

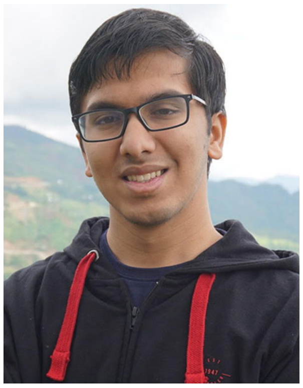
References

1. Webroot. Quarterly threat trends: Phishing attacks growing in scale and sophistication; 2017. Accessed 14 Nov 2017. https://s3-us-west-1.amazonaws.com/webroot-cms-cdn/8415/0585/3084/Webroot_Quarterly_Threat_Trends_September_2017.pdf.
2. Chiew, K. L., Yong, K. S. C., & Tan, C. L. (2018). A survey of phishing attacks: their types, vectors and technical approaches. *Expert Systems with Applications*, *106*, 1–20.
3. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). "Andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems.*, *38*(1), 161–190.
4. Dhamija, R., Tygar, J.D., & Hearst, M. (2006). Why phishing works. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, pp. 581–590.
5. Alsharnouby, M., Alaca, F., & Chiasson, S. (2015). Why phishing still works: User strategies for combating phishing attacks. *International Journal of Human-Computer Studies*, *82*, 69–82.
6. Aburrous, M., Hossain, M.A., Dahal, K., & Thabtah, F. (2010). Predicting phishing websites using classification mining techniques with experimental case studies. In *Information technology: new generations (ITNG), 2010 7th international conference on IEEE* pp. 176–181.
7. Adebowale, M. A., Lwin, K. T., Sanchez, E., & Hossain, M. A. (2019). Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text. *Expert Systems with Applications*, *115*, 300–313.
8. Sanglerdsinlapachai, N., & Rungsawang, A. (2010). Using domain top-page similarity feature in machine learning-based web phishing detection. In *Knowledge discovery and data mining, 2010. WKDD'10. 3rd international conference on IEEE*, pp. 187–190.
9. Jagadeesan, S., Kumar, A., & Kumar, S. (2018). URL phishing analysis using random forest. *International Journal of Pure and Applied Mathematics*, *118*(20), 4159–4163.
10. Fette, I., Sadeh, N., & Tomasic, A. (2007). Learning to detect phishing emails. In *Proceedings of the 16th international conference on world wide web. ACM*, pp. 649–656.
11. Şentürk Ş, Yerli E, Soğukpınar İ. (2017). Email phishing detection and prevention by using data mining techniques. In *2017 International conference on computer science and engineering (UBMK). IEEE*, pp. 707–712.
12. Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based associative classification data mining. *Expert Systems with Applications*, *41*(13), 5948–5959.
13. Hara, M., Yamada, A., & Miyake, Y. (2009). Visual similarity-based phishing detection without victim site information. In *Computational intelligence in cyber security, 2009. CICS'09. IEEE symposium on. IEEE*, pp. 30–36.
14. Afroz, S., & Greenstadt, R. (2011). Phishzoo: Detecting phishing websites by looking at them. In *Semantic computing (ICSC), 2011 5th IEEE international conference on. IEEE*, pp. 368–375.
15. Medvet, E., Kirda, E., & Kruegel, C. (2008). Visual-similarity-based phishing detection. In *Proceedings of the 4th international conference on security and privacy in communication networks*, ACM, p. 22.
16. Wenyin, L., Huang, G., Xiaoyue, L., Min, Z., & Deng, X. (2005). Detection of phishing webpages based on visual similarity. In *Special interest tracks and posters of the 14th international conference on world wide web*, ACM, pp. 1060–1061.
17. Fu, A. Y., Wenyin, L., & Deng, X. (2006). Detecting phishing web pages with visual similarity assessment based on Earth Mover's Distance (EMD). *IEEE Transactions on Dependable and Secure Computing*, *3*(4), 301–311.
18. Dhamija, R., & Tygar, J.D. (2005). The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 symposium on usable privacy and security*, ACM, pp. 77–88.
19. Barraclough, P. A., Hossain, M. A., Tahir, M., Sexton, G., & Aslam, N. (2013). Intelligent phishing detection and protection scheme

for online transactions. *Expert Systems with Applications*, *40*(11), 4697–4706.

20. Aburrous, M., Hossain, M. A., Dahal, K., & Thabtah, F. (2010). Intelligent phishing detection system for e-banking using fuzzy data mining. *Expert Systems with Applications*, *37*(12), 7913–7921.
21. Shreeram, V., Suban, M., Shanthi, P., Manjula, K. (2010). Anti-phishing detection of phishing attacks using genetic algorithm. In *Communication control and computing technologies (ICCCCT), 2010 IEEE international conference on, IEEE*, pp. 447–450.
22. Mohammad, R. M., Thabtah, F., & McCluskey, L. (2014). Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, *25*(2), 443–458.
23. Kazemian, H. B., & Ahmed, S. (2015). Comparisons of machine learning techniques for detecting malicious webpages. *Expert Systems with Applications*, *42*(3), 1166–1177.
24. Pan, Y., & Ding, X. (2006). Anomaly based web phishing page detection. In *2006 22nd annual computer security applications conference (ACSAC'06)*, pp. 381–392.
25. Miyamoto, D., Hazeyama, H., & Kadobayashi, Y. (2008). *An evaluation of machine learning-based methods for detection of phishing sites. Advances in neuro-information processing* (pp. 539–546). Berlin: Springer.
26. Dua, D., & Taniskidou, E.K. (2017). UCI Machine Learning Repository; 2017. University of California, Irvine, School of Information and Computer Sciences. Accessed 11 Oct 2017. http://archive.ics.uci.edu/ml.
27. Mohammad, R.M., Thabtah, F., & McCluskey, L. (2012) An assessment of features related to phishing websites using an automated technique. In *Internet technology and secured transactions, 2012 international conference for IEEE*, pp. 492–497.
28. Alexa Inc. How are Alexa's traffic rankings determined?; 2018. Accessed 16 Jan 219. https://support.alexa.com/hc/en-us/articles/200449744-How-are-Alexa-s-traffic-rankings-determined-.
29. Auria, L., & Moro, R.A. (2008). DIW. Support vector machines (SVM) as a technique for solvency analysis. DIW discussion papers.
30. Dewang, R. K., & Singh, A. K. (2018). State-of-art approaches for review spammer detection: A survey. *Journal of Intelligent Information Systems*, *50*(2), 231–264.
31. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.
32. Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 1–27.
33. Platt, J.C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. MSR-TR-98-14.
34. Ch, S., Sohani, S., Kumar, D., Malik, A., Chahar, B., Nema, A., et al. (2014). A support vector machine-firefly algorithm based forecasting model to determine malaria transmission. *Neurocomputing*, *129*, 279–288.
35. Chao, C. F., & Horng, M. H. (2015). The construction of support vector machine classifier using the firefly algorithm. *Computational Intelligence and Neuroscience*, *2015*, 2.
36. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.
37. Tuba, E., Mrkela, L., & Tuba, M. (2016) Support vector machine parameter tuning using firefly algorithm. In *26th International conference radioelektronika, IEEE*, pp. 413–418.
38. Chakraborty, A., & Kar, A.K. (2016). A review of bio-inspired computing methods and potential applications. In *Proceedings of the international conference on signal, networks, computing, and systems*. Springer, pp. 155–161.
39. Kar, A. K. (2016). Bio inspired computing-A review of algorithms and scope of applications. *Expert Systems with Applications*, *59*, 20–32.
40. Yang, X. S. (2014). *Nature-inspired optimization algorithms*. Amsterdam: Elsevier.
41. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - international conference on neural networks*, vol. 4, pp. 1942–1948.
42. Chen, K. H., Chen, L. F., & Su, C. T. (2014). A new particle swarm feature selection method for classification. *Journal of Intelligent Information Systems*, *42*(3), 507–530.
43. Karaboga D. (2005). An idea based on honey bee swarm for numerical optimization. TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.
44. Yang, X.S., & Deb, S. (2009). Cuckoo search via Lévy flights. In *Nature and biologically inspired computing, 2009. NaBIC 2009. World Congress on, IEEE*, pp. 210–214.
45. Chakraborty, A., & Kar, A. K. (2017). *Swarm intelligence: A review of algorithms. Nature-Inspired Computing and Optimization* (pp. 475–494). Berlin: Springer.
46. Yang, X.S. (2010) A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010), pp. 65–74.
47. Yang, X,S. (2009). Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*. Springer, pp. 169–178.
48. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.
49. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61.
50. Olatomiwa, L., Mekhilef, S., Shamshirband, S., Mohammadi, K., Petković, D., & Sudheer, C. (2015). A support vector machine-firefly algorithm-based model for global solar radiation prediction. *Solar Energy*, *115*, 632–644.
51. Tharwat, A., Hassanien, A. E., & Elnaghi, B. E. (2017). A BA-based algorithm for parameter optimization of support vector machine. *Pattern Recognition Letters*, *93*, 13–22.
52. Elhariri, E., El-Bendary, N., Hassanien, A.E., & Abraham, A. (2015) Grey wolf optimization for one-against-one multi-class support vector machines. In *Soft computing and pattern recognition (SoCPaR), 2015 7th international conference of, IEEE*, pp. 7–12.
53. Ala'M, A. Z., Faris, H., Hassonah, M. A., et al. (2018). Evolving support vector machines using Whale Optimization algorithm for spam profiles detection on online social networks in different lingual contexts. *Knowledge Based Systems*, *153*, 91–104.
54. Gupta, S., Kar, A. K., Baabdullah, A., & Al-Khowaiter, W. A. (2018). Big data with cognitive computing: A review for the future. *International Journal of Information Management*, *42*, 78–89.
55. Ali, H., & Kar, A. K. (2018). Discriminant analysis using ant colony optimization-an intra-algorithm exploration. *Procedia Computer Science*, *132*, 880–889.
56. Kar, A. K. (2015). A hybrid group decision support system for supplier selection using analytic hierarchy process, fuzzy set theory and neural network. *Journal of Computational Science*, *6*, 23–33.
57. Khalilpourazari, S., & Khalilpourazary, S. (2018). optimization of production time in the multi-pass milling process via a Robust Grey Wolf optimizer. *Neural Computing and Applications*, *29*(12), 1321–1336.
58. Mansouri, A., Aminnejad, B., & Ahmadi, H. (2018). Introducing modified version of penguins search optimization algorithm (PeSOA) and its application in optimal operation of reservoir systems. *Water Science and Technology Water Supply*, *18*(4), 1484–1496.
59. Xue, X., & Xiao, M. (2017). Deformation evaluation on surrounding rocks of underground caverns based on PSO-LSSVM. *Tunnelling and Underground Space Technology*, *69*, 171–181.

60. Yi, T. H., Zhou, G. D., Li, H. N., & Wang, C. W. (2017). Optimal placement of triaxial sensors for modal identification using hierarchic wolf algorithm. *Structural Control and Health Monitoring*, *24*(8), e1958.
61. Li, H., Liu, X., Huang, Z., Zeng, C., Zou, P., Chu, Z., et al. (2020). Newly emerging nature-inspired optimization-algorithm review, unified framework, evaluation, and behavioural parameter Optimization. *IEEE Access*, *8*, 72620–72649.
62. Molina D, Poyatos J, Del Ser J, García S, Hussain A, Herrera F. (2020). Comprehensive taxonomies of nature-and bio-inspired optimization: inspiration versus algorithmic behavior, critical analysis and recommendations. arXiv preprint arXiv:2002.08136.
63. Kar, A. K., & Dwivedi, Y. K. (2020). Theory building with big data-driven research-moving away from the what towards the why. *International Journal of Information Management*, *54*, 102205.



Sagnik Anupam is an undergraduate student at Massachusetts Institute of Technology, Cambridge, MA, USA. He is an alumnus of DPS R. K. Puram, and his research interests include optimization research and machine learning.

Arpan Kumar Kar is an Associate Professor in DMS, Indian Institute of Technology Delhi, India. His research interests are in the domain of data science, digital transformation, internet ecosystems, social media and ICT-based public policy. He has authored over 150 peer reviewed articles and edited 7 research books. He is the Editor in Chief of International Journal of Information Management Data Insights, published by Elsevier. He further actively supports reputed knowledge platforms like IJEGR, ISF, ATPEM, GJFSM, IJIM, TBL, PACIS, ICIS, ECIS and IFIP conferences as editorial advisory board, associate editor and editorial board member. Prior to joining IIT Delhi, he has worked for IIM Rohtak, IBM Research and Cognizant. He has received numerous awards for his research contributions from organizations like IFIP, TCS, PMI, AIMS, IIT Delhi, BK Birla (BimTech) and IIM Rohtak.