Taylor & Francis
Taylor & Francis Group

Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/taut20

Unveiling the IoT's dark corners: anomaly detection enhanced by ensemble modelling

Jisha Jose & J. E. Judith

To cite this article: Jisha Jose & J. E. Judith (2024) Unveiling the IoT's dark corners: anomaly detection enhanced by ensemble modelling, Automatika, 65:2, 584-596, DOI: 10.1080/00051144.2024.2304369

To link to this article: https://doi.org/10.1080/00051144.2024.2304369

AUTOMATIKA
2024, VOL. 65, NO. 2, 584–596
https://doi.org/10.1080/00051144.2024.2304369

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

Unveiling the IoT's dark corners: anomaly detection enhanced by ensemble modelling

Jisha Jose and J. E. Judith

Department of Computer Science and Engineering, Noorul Islam Centre for Higher Education, Kumarakovil, India

ABSTRACT

The growing Internet of Things (IoT) landscape requires robust security; traditional rule-based systems are insufficient, driving the integration of machine learning (ML) for effective intrusion detection. This paper provides an inclusive overview of research efforts focused on harnessing ML methodologies to fortify intrusion detection within IoT. Tailored feature extraction techniques are pivotal for achieving high detection accuracy while minimizing false positives. The study employs the IoT23 dataset from Kaggle and incorporates four optimization algorithms – Particle Swarm Optimizer, Whale-Pearson optimization algorithm, Harris-Hawks Optimizer, and Support Vector Machine with Particle Swarm optimization algorithm (SVM-PSO) – for feature extraction and selection. A comparison with ML algorithms such as logistic regression, decision tree and naïve Bayes classifier highlights Harris-Hawks Optimizer as the most effective. Furthermore, ensemble methods, particularly the fusion of random forest with HHO optimization, yield an impressive accuracy of 99.97%, surpassing AdaBoost and XGBoost approaches. This paper underscores the application of diverse ensemble learning techniques to enhance intrusion detection precision and efficiency within the intricate IoT landscape, effectively tackling the challenges posed by its complex and ever-changing nature.

ARTICLE HISTORY
Received 26 September 2023
Accepted 10 December 2023

KEYWORDS
Harris-Hawks optimizer; particle swarm optimizer; logistic regression; Whale-Pearson optimization algorithm; Naïve Bayes classifier

1. Introduction

The IoT revolutionizes technology interaction by interconnecting physical objects through sensors and internet connectivity. This enables data exchange and automation across sectors like healthcare [1], transportation, agriculture and smart cities, as shown in Figure 1. Wearables and connected devices enhance healthcare monitoring, while smart transportation systems improve traffic management. Agriculture benefits from IoT-driven precision, and smart cities optimize services. Despite these benefits, IoT faces security [2], privacy and interoperability challenges. Robust cybersecurity, privacy regulations and standardized approaches are vital. The future holds 5G-enabled IoT, edge computing and AI-driven analytics, reshaping industries and daily tech interactions.

1.1. Security issues in IoT

Security issues in IoT stem from the vast interconnectedness of devices, raising concerns about data privacy, unauthorized access and potential breaches [3]. Vulnerabilities arise due to diverse device types and communication protocols, often lacking robust security measures. Without proper safeguards, IoT devices can be exploited, leading to compromised personal data, disruption of services and even broader cyber threats.

1.1.1. Authentication, data privacy and encryption

Security issues in the IoT ecosystem stem from its vast network of interconnected devices. One major concern is weak authentication and authorization mechanisms [4]. Many devices come with default credentials or lack proper authentication, making them susceptible to unauthorized access. Additionally, data privacy is a significant worry. IoT devices collect a wealth of personal and sensitive data, raising concerns about how this data is stored, transmitted and utilized. Encryption [5] is essential to protect data from interception during transmission and storage, but its implementation varies widely across IoT devices. Ensuring strong encryption protocols and secure key management is crucial to maintain data confidentiality.

1.1.2. Software vulnerabilities, lack of updates and device management

Software security is a significant challenge in IoT. Many devices run on outdated or unpatched software, leaving them vulnerable to known exploits [6]. Regular software updates are essential to address vulnerabilities and improve overall security, but IoT devices often lack robust mechanisms for applying updates. Additionally, the diversity of devices and manufacturers makes ensuring consistent and timely updates challenging. Device management is another issue; managing a

CONTACT Jisha Jose ✉ jishajose.cse@gmail.com; jisha.jose@mbcet.ac.in, Department of Computer Science and Engineering, Noorul Islam Centre for Higher Education, Kumarakovil, Thuckalay, Kanyakumari District, Tamil Nadu 629180, India

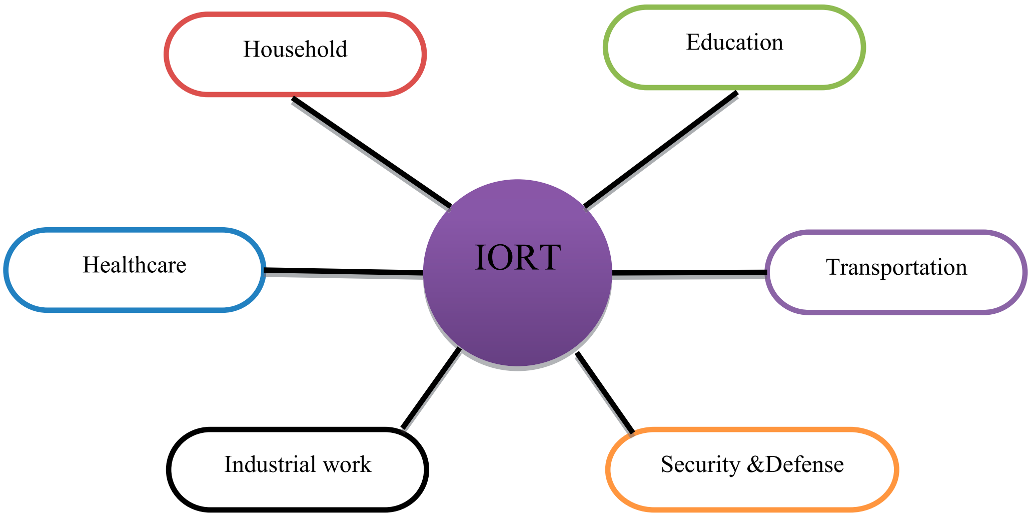

Figure 1. Daily life application of IoT.

vast number of devices and ensuring they are all properly configured and updated is complex and resource-intensive [7]. Inadequate device management can lead to security gaps that attackers can exploit.

1.1.3. Interoperability, supply chain risks and regulatory gaps

Interoperability between different IoT devices and protocols is a significant concern. Incompatibilities between devices and protocols can introduce vulnerabilities that attackers might exploit to gain unauthorized access or disrupt communication. Moreover, the global supply chain for IoT components introduces risks. Compromised or counterfeit components can find their way into devices, potentially enabling backdoors or other security vulnerabilities. Regulatory gaps are also apparent, with different regions having varying levels of legislation and standards for IoT security. A lack of consistent regulations can result in varying security practices across devices, leaving some more vulnerable than others. Table 1 shows the merits and demerits of IoT.

Addressing these security issues requires a collaborative effort involving manufacturers, service providers, policymakers and end-users. Industry-wide standards for security practices, strong authentication mechanisms, robust encryption protocols, regular software updates and comprehensive device management are essential steps to build a more secure IoT landscape. As IoT continues to evolve, proactive security measures must evolve with it to ensure the potential benefits of this interconnected ecosystem are not overshadowed by security risks.

Table 1. Merits and demerits of IoT.

Advantages	Disadvantages
Minimizes the human work and effort	Increased privacy concerns
Saves time and effort	Increased unemployment rates
Good for personal safety and security	Highly dependent on the internet
Useful in traffic and other tracking or monitoring systems	Lack of mental and physical activity by humans leading to health issues
Beneficial for the healthcare industry	Complex system for maintenance
Improved security in homes and offices	Lack of security
Reduced use of many electronic devices as one device does the job of a lot of other devices	Absence of international standards for better communication

1.1. Intrusion detection system

In the expansive landscape of the IoT, cybersecurity is of utmost importance due to the seamless communication between devices [8]. Intrusion detection, a vital component of cybersecurity, plays a pivotal role in safeguarding IoT networks against unauthorized access and malicious activities. IDS are integral in monitoring network traffic and system behaviour, quickly identifying suspicious actions that could compromise the security and integrity of IoT devices and data [9]. The dynamic and diverse nature of IoT devices presents challenges for traditional security measures, making tailored intrusion detection mechanisms essential. ML and artificial intelligence techniques are being leveraged to create adaptive, context-aware IDS that can learn normal device behaviour and swiftly detect deviations, providing effective defense against a wide range of threats [10].

As the cybersecurity landscape intensifies, the importance of IDS grows significantly. Organizations face various attacks like malware [11], ransomware and data breaches, highlighting the need for vigilant defense mechanisms. IDS continually monitor network traffic [12], system behaviour and data access patterns, alerting security personnel to any anomalies. They can also uncover unusual patterns indicative of zero-day vulnerabilities, offering early warnings and reducing risks. Complying with regulations and industry standards, IDS play a pivotal role in meeting security requirements and upholding data integrity. Ultimately, IDSs offer proactive protection by strengthening defenses, identifying breaches and preserving sensitive information in a rapidly evolving cyber environment.

2. Literature review

2.1. Deep learning approaches

Roopak et al. [13] introduced innovative deep learning models aimed at enhancing the cybersecurity of IoT networks. Despite the rapid expansion of IoT technology, its vulnerability to cyber threats remains a significant concern. The paper addressed this issue by presenting deep learning solutions for IoT network security. Notably, the growing frequency of DDoS attacks on IoT networks is highlighted as a major threat. The proposed models are rigorously evaluated using the CICIDS2017 dataset, demonstrating an impressive accuracy rate of 97.16% in detecting DDoS attacks. A comparative analysis is conducted against conventional ML algorithms.

Verma et al. [14] delves into the feasibility of employing ML classification algorithms to bolster the security of IoT networks against DoS attacks. Through an extensive investigation, the study focuses on enhancing the development of anomaly-based IDS. Key metrics and validation approaches are used to evaluate the effectiveness of these models. Noteworthy datasets like CIDDS-001, UNSW-NB15 and NSL-KDD serve as benchmarks for classifier assessment. Statistical tests such as Friedman and Nemenyi are utilized to scrutinize significant differences between classifiers. The study incorporates Raspberry Pi to gauge classifier response times within the context of IoT hardware. In order to develop IoT security measures, the article also provides a method for choosing the best classifier depending on specific needs.

Otoum et al. [15] introduced a novel Deep Learning-based IDS tailored for IoT environments. This innovative system leverages a combination of the Spider Monkey Optimization algorithm (SMO) and the Stacked-Deep Polynomial Network (SDPN) to achieve heightened accuracy in detecting security threats. SMO is employed for optimal feature selection within the datasets, while SDPN is responsible for classifying data into normal and anomaly categories. The DL-ID system is capable of identifying a range of anomalies, including Denial of Service (DoS), User to Root (U2R) attacks, probe attacks and Remote to Local (R2L) attacks. By amalgamating these advanced techniques, the proposed DL-ID system showcases potential for enhanced intrusion detection accuracy in IoT environments, thereby contributing to heightened cybersecurity in this dynamic and interconnected landscape.

2.2. Machine learning approaches

Mahmood et al. [16] addressed challenges arising from the implementation of IoT systems and proposes solutions through ML techniques. It focuses on an RFID system, crucial for IoT, comparing various technologies to select optimal ones based on functionality and security. Using a prototype IoT system exemplified by baggage tracking at an airport, the research highlights five main differences between IoT and traditional systems: technical limitations of IoT devices, the significant influence of the physical environment, inadequate security focus during design, susceptibility of IoT devices to attacks like DDoS, and heightened privacy sensitivity of IoT use cases. The study utilizes the KDD Cup 1999 dataset, a renowned IoT and cybersecurity dataset, for training, testing and validation purposes, utilizing the MATLAB R2019a software. By identifying challenges and implementing solutions, the paper contributes to enhancing the understanding and effective implementation of IoT systems while addressing critical security and privacy concerns.

Saheed et al. [17] addressed the challenge by proposing a ML-based IDS for IoT network attacks. It focuses on applying supervised ML algorithms to detect attacks, employing feature scaling and dimensionality reduction techniques. Six ML models were tested on the UNSW-NB15 dataset, containing various attack types and normal activities. Experimental results, including accuracy (99.9%), MCC (99.97%), and other metrics, demonstrated the effectiveness of the ML-IDS. The paper contributes to enhancing IoT security and privacy by utilizing ML approaches for robust intrusion detection, thereby mitigating the challenges posed by IoT device limitations and network scalability.

2.3. Hybrid approaches

Sahu et al. [18] introduced an innovative security framework and attack detection mechanism centred around a Deep Learning model to effectively identify malicious devices. This approach addresses existing gaps by utilizing a Convolutional Neural Network (CNN) to extract precise feature representations from data, followed by classification through a Long Short-Term Memory (LSTM) Model. The experimental evaluation employs a dataset collected from twenty compromised IoT devices utilizing Raspberry Pi. Notably, the study demonstrates impressive empirical results, with a 96% accuracy rate for detecting attacks. By leveraging the combined power of CNN and LSTM, the proposed mechanism offers a promising solution for enhancing the detection of malicious activities in IoT environments, contributing to heightened security and reliability in the rapidly expanding IoT landscape. Kumar et al. [19] presented an intelligent cyber-attack detection system customized for IoT networks using a novel hybrid feature reduction technique. This method involves three key steps: initiating feature ranking, random forest means decrease accuracy, gain ratio, resulting in distinct sets of features. These sets are combined using a specialized mechanism known as the AND operation to create a singular optimized feature set. This condensed feature set is then inputted into three well-established ML algorithms – random forest, K-nearest neighbour and XGBoost – to identify cyber-attacks. The efficacy of the proposed framework is assessed using established datasets like NSL-KDD, as well as contemporary IoT-centric datasets like BoT-IoT and DS2OS. By adopting this strategy, the paper advances the field of intelligent cyber-attack detection in IoT networks, enhancing security through refined feature selection and robust ML algorithms.

3. Materials and method

This section provides a comprehensive overview of the foundational components driving the study. The IoT 23 dataset, formatted as a CSV file, forms the basis for the investigation's effectiveness. The study capitalizes on four distinct optimization algorithms: PSO [20], WOA [21], Harris-Hawks Optimizer [22] and SVM-PSO. Employing these approaches, various types of features are extracted from the dataset, with ML techniques assessing the most effective feature set via logistic regression [23], decision tree classifier [24] and naïve Bayes classifier [25]. This assessment determines that the HHO algorithm yields the optimal feature selection. To further bolster intrusion detection, the study integrates three ensemble models: Adaboost classifier [26], XG Boost classifier [27] and random forest classifier [28]. The proposed system's schematic is presented in Figure 2, illustrating the sequence of operations encompassing the dataset, optimization techniques, ML methods and ensemble models.

3.1. Dataset description

Curated for rigorous study, the IoT 23 dataset represents a substantial compilation meticulously structured to propel research and innovations in IoT security.

Comprising diverse and realistic IoT network traffic, this dataset plays a pivotal role in enabling the development and evaluation of IDS and cybersecurity solutions. It encompasses various attack scenarios and normal activities, enhancing its utility for training and testing ML models. The IoT 23 dataset is crucial

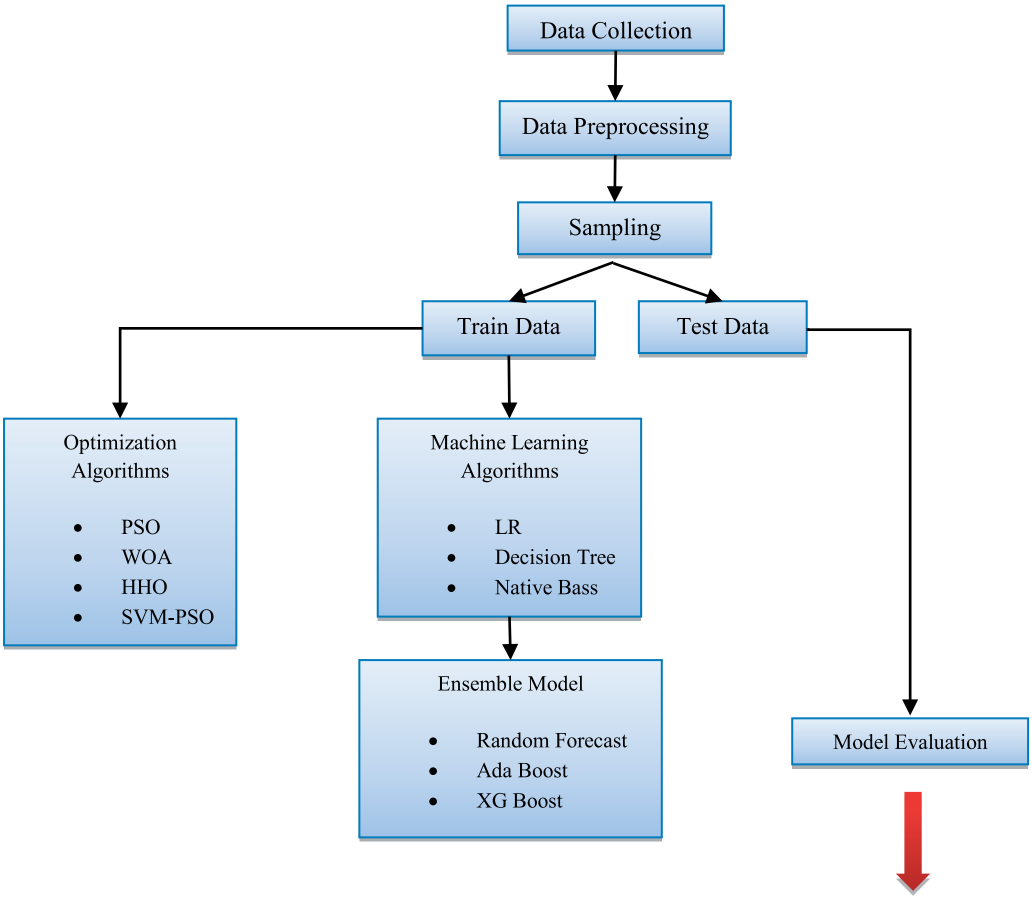

Figure 2. Proposed system.

for fostering a deeper understanding of the evolving threat landscape within IoT environments and for fostering innovation in cybersecurity measures to ensure the integrity and privacy of interconnected devices and systems.

3.2. Data preprocessing

Data preprocessing of the dataset involves preparing the CSV data for analysis. Initially, the dataset is loaded using pandas, and exploratory data analysis is conducted to understand its structure and identify issues. Missing values are addressed by either removing or imputing them, categorical variables are transformed using techniques like one-hot encoding, and feature scaling. The data is split into training, and test sets, and the preprocessed data for future use. This comprehensive process ensures the dataset is cleansed, transformed and structured in a way that optimizes its usability. The dataset consists of 21 features which are described in the Table 2.

Figure 3 illustrates the visual representation of the dataset, where 0 corresponds to the count of instances classified as non-malicious cases, and 1 signifies the count of instances categorized as malicious cases.

Table 2. Features in the dataset.

Attribute Number	Features	Description
1	Fields-ts	Flow start time
2	Uid	Unique ID
3	id.orig-h	Source IP address
4	id.orig-p	Source port
5	id.resp-h	Destination IP address
6	id.resp-p	Destination port
7	proto	Transaction protocol
8	Service	http, ftp, smtp, ssh, dns, etc.
9	Duration	Record total duration
10	Orig-bytes	Source 2 destination transaction bytes
11	Resp-bytes	Destination 2 source transaction bytes
12	Conn-state	Connection state
13	Local-orig	Source local address
14	Local-resp	Destination local address
15	Missed-bytes	Missing bytes during transaction
16	History orig-pkts	History of source packets
17	Orig-ip-bytes	Flow of source bytes
18	Resp-pkts	Destination packets
19	Resp-ip-bytes	Flow of destination bytes
20	Tunnel-parents	Traffic tunnel
21	label	Attack label

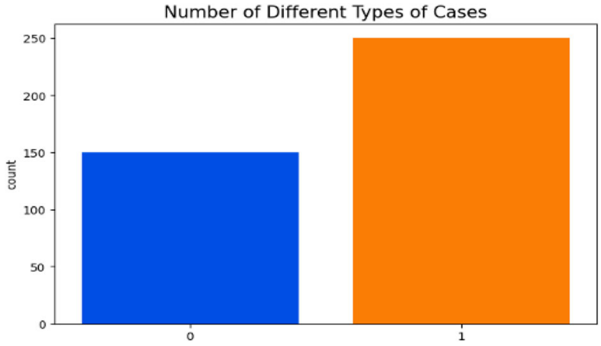

Figure 3. Data visualization.

3.3. Meta heuristic algorithms

3.3.1. Particle swarm optimization

PSO is a computational optimization technique inspired by the social behaviour of birds or fish. In PSO, a population of potential solutions (particles) navigates through a search space to find the optimal solution for a given problem. Researchers commonly simplify PSO algorithm as a random search challenge in a space with multiple dimensions (D-dimensional space). The primary objective is to optimize the objective function. Within a D-dimensional space, a population is formed by n particles represented as $p_k = (p_{k1}, p_{k2}, \ldots, p_{kD})$ T and the Kth particle holds a d-dimensional position vector $x_k = (x_{k1}, x_{k2}, \ldots, x_{kd})$ T. The fitness of each particle in the population is assessed using a fitness function. An introduced hyperparameter α manages the connection between classifier performance Q and the proportion of the feature subset N_g in relation to the total number of features N_u.

$$F(X) = \alpha\,(1\ -\ \mathrm{Q})\ +\ (1\ -\ \alpha)\left(1 - \frac{N_g}{N_u}\right) \qquad (1)$$

As particle k explores the D-dimensional space, it commences from a set of randomly positioned particles, gradually converging towards an optimal solution through iterative processes. During the ongoing particle search, the self-found optimal position $p_k = (p_{k1}, p_{k2}, \ldots, p_{kD})$ T serves as the local optimal solution, characterized by its associated velocity vector $v_k = (v_{k1}, v_{k2}, \ldots, v_{kd})$ T. In contrast, the global optimal solution is represented by the optimal position $P_g = (P_{g1}, P_{g2}, \ldots, P_{gd})$ T, which is established by the entire particle swarm's collective search. Throughout each iteration, a particle adjusts both its position and velocity based on the tracking of two optimal solutions, namely (P_i, P_g). The update mechanism follows a formula as shown in Equations (2) and (3).

$$V_{kd}(T+1) = \omega V_{kd}(T) + clr1(P_{kd}(T) - xkd(T)) + c2r2(P_{gd}(T) - xkd(T)) \qquad (2)$$

$$xkd(T+1) = x(T) + vkd(T+1), \quad k = 1, 2, \ldots, N : d = 1, 2, \ldots, D \qquad (3)$$

Here, N denotes the complete count of particles within the population, and d signifies the specific d-th dimension of particle k. T represents the current iteration number, while ω stands for a non-negative inertia factor that governs the balance between global and local optimization capacities. Higher values of ω amplify global optimization while diminishing local optimization strength, and the reverse holds true. The PSO algorithm is structured as outlined below.

Algorithm 1 Particle Swarm Optimization (PSO)

```
Input: N: population size
   pi: local optimal position
   pg: group optimal position
   fit: fitness function
Output: pg
   Randomly initialize the position xi
   i
   while criterion is not met do
      for i = 1 to N do
         calculate the fitness value of each particle according to the fitness
            function
         if fit(xi) is greater than fit (pi) then
            $pi \leftarrow xi$
         If fit (pi) is greater than fit (ps) then
            $ps \leftarrow pi$
         Update the position
return pg
```

Table 3. Features selected by the PSO algorithm.

id.orig_h	id.orig_p	id.resp_p	service	orig_bytes	
0	17576	2291682261	1	0.000005	0
1	17576	1023409237	1	0.000002	0
2	17832	2626702293	1	0.000005	0
3	17576	1308744916	1	0.000003	0
4	17576	2555643759	1	0.000002	0
…	…	…	…	…	…
99994	17576	550286945	1	0.000002	0
99995	17576	1259799706	1	0.000006	0

The PSO algorithm strategically identifies and selects five features from the given dataset. Through iterative optimization, PSO effectively evaluates numerous combinations of features to determine the optimal subset. By leveraging its swarm intelligence-inspired mechanism, PSO hones in on the most relevant attributes that contribute significantly to the study's objectives. Table 3 tabulates the features being selected by the PSO algorithm.

3.3.2. Whale-Pearson optimization algorithm

The Whale Pearson optimization algorithm represents an enhanced iteration of the Binary Whale swarm algorithm [29]. This new version incorporates the concept of simulated annealing for updating the positions. The foundation of the Whale optimization algorithm is rooted in imitating the foraging movements of whales. The revised algorithm maintains the fundamental stages of its forerunner for the exploration procedure. However, the original position updation mechanism has been substituted with an innovative correlation-based selection algorithm. This new method integrates both correlation and classifier-guided fitness evaluations, categorizing it as an embedded selection approach. The operational mechanics of the novel correlation design are elucidated in the subsequent discussion.

Assume X_o represents the local optimal solution achieved from the Binary Whale wrapper at an iteration's conclusion. The process of updating positions relies on X_o and a predetermined maximum iteration count. This function produces I_p random solutions, each of which undergoes correlation evaluation using Pearson correlation method. In this context, f_a represents the attribute class, and x_j signifies the feature attribute where j spans from 0 to T. The mean correlation between the features and the class attribute is computed using Equations (4) and (5).

$$m = \sum co(X_j, f_a) \tag{4}$$

$$co(x, y) = \frac{\sum_{j=1}^{n} (X_j - \bar{X})(Y_j - \bar{Y})}{\sqrt{\sum_{j=1}^{n} (X_j - \bar{X})^2}\sqrt{\sum_{j=1}^{n} (Y_j - \bar{Y})^2}} \tag{5}$$

Equation (5) involves x, the input and y symbolizing the attribute that holds the output classification. By applying the Mutation function to the prevailing Gbest solution, a set of I_p random position vectors is generated. Each vector is assessed using the unique correlation-based objective function outlined in 3. The most optimal solution among the obtained set is adopted as the present position, and the quest for finding food persists until the predefined maximum iteration count is reached.

$$obj(in) = co(in, class) \tag{6}$$

Algorithm 2 Whale Pearson Feature Selection wrapper

```
Initialize: lb = 0; ub = 1;//upper and lower boundaries
Initialize: whales, itermax
Initialize: whale-position, food-position
Initialize: whale-fitness, food-fitness
Initialize: i = 1 //initial iteration
While $i \le itermax$do
   Calculate fitness of each whale with objective function
   F = Best_Whale
   X = Position of the Best_Whale
   update whale positions with steering function
   foreach whale (xi) do
      if obj(whale_position)<obj(food_position)
         then
         | food_position = whale_position
         else
         |continue:
         end
   end
   return food_position
   mutate (food_position,max_iter)
   if (cor(new_position, class)>
      cor(food_position, class))
      (fitness(new_position)<
      fitness (food_position)) then
      | food_position = new_position:
      end
end
```

The Whale Swarm Wrapper technique yields a smaller set of selected features in comparison to the PSO approach. This suggests that the Whale Swarm Wrapper prioritizes a more focused subset of attributes from the dataset. The contrast in the number of selected features underscores the distinct feature evaluation strategies employed by the two algorithms, potentially highlighting the different ways they assess feature

Table 4. Features selected by the WOA algorithm.

	uid	id.orig-h	local-resp
0	3232261231	17576	0
1	3232261231	17576	0
2	3232261231	17832	0
3	3232261231	17576	0
4	3232261231	17576	0

importance and relevance. Table 4 shows the features selected by the WOA algorithm.

3.3.3. Harris-Hawks optimizer (HHO)

The population-based Harris' Hawks Optimization (HHO) method [30], harnesses the cooperative behaviour exhibited by groups of Harris' hawks, along with their distinct hunting tactics such as pursuing prey, establishing blockades, and executing surprise dives. The algorithm operates within two primary phases: exploration, where potential prey is identified, and exploitation, which involves strategizing attacks, including blockades and surprise dives.

The algorithm involves several steps. First, it estimates the population vector of hawks and calculates their fitness values, along with identifying the best position vector for the prey. Following this, it proceeds to modify the initial energy (E_0) and the resistance strength (J) of the prey, along with adjusting its escaping energy, during every iteration. These updates are performed using Equations (7)–(9). This approach allows the algorithm to dynamically adapt and refine its tactics to optimize the search process for improved performance.

$$E_0 = 2rand() - 1 \tag{7}$$

$$J = 2(1 - ramd()) \tag{8}$$

$$E = 2E_0\left(1 - \frac{t}{t_{\max}}\right) \tag{9}$$

The exploration phase is characterized by achieving a prey escaping energy value greater than 1. During this phase, the hawk position vector is iteratively updated using Equation (9) to determine its blockade position. Xm(t) signifies average population of hawk, UB and LB denote upper and lower bounds, representing the best-positioned and least-fit hawk in iteration t. In the exploitation phase, four modes are distinguished:

Soft blockade: The escaping energy and unsuccessful escape chance exceed 0.5. The victim tyres out due to successive hawk sieges, eventually falling prey to a surprising dive.

Hard blockade: Prey's escaping energy is less than 0.5, but its unsuccessful escape chance is better. The prey's energy diminishes, and the hawk hunts it unimpeded, incorporating a surprising dive.

Soft blockade (different scenario): Prey's escaping energy surpasses 0.5, yet its successful escape chance is below 0.5. The prey attempts deceptive escape, but the hawks tyre of the ruse and ultimately hunt it down through various blockades and movements.

Hard blockade (limited energy): Both parameters fall below 0.5, indicating the prey's lack of energy.

The algorithm further updates the hawk's position vector using Equations (10) to (12). The algorithm concludes after multiple iterations, with the fittest hawk successfully capturing the prey, signifying the termination.

$$\vec{X}(t+1) = \Delta\vec{X}(t) - E\left|JV_{prey}(t) - \vec{X}(t)\right|, \Delta\vec{X}(t) \tag{10}$$

$$= \vec{X}_{prey}(t) - \vec{X}(t) \tag{11}$$

$$\vec{X}(t+1) = \vec{X}_{prey}(t) - E\left|\Delta\vec{X}(t)\right| \tag{12}$$

$$\vec{X}(t+1) = \begin{cases} Y, F(Y) < F(X(t)) \\ Z, F(Z) < F(X(t)) \end{cases} \tag{13}$$

Figure 4 shows Flowchart illustrating HHO process the HHO algorithm excels in feature selection compared to other algorithms. This indicates that HHO adeptly identifies and ranks the most pertinent features from the dataset. Its capability to yield the optimal feature subset underscores its effectiveness in recognizing attributes that significantly contribute to the study's objectives, potentially leading to enhanced model performance. The features selected by the HHO algorithm are shown in Table 5.

3.3.4. Support vector machine with particle swarm optimization algorithm (SVM-PSO)

The SVM kernel employs a technique called the "kernel trick" to address non-linear problems using a linear classifier. This approach transforms data from being linearly inseparable to becoming separable. The kernel function is applied to each data instance, converting the initial non-linear observations into a higher-dimensional space where they become separable. This process enhances the SVM's ability to effectively classify complex data [14].

Support Vector Machine with SVM-PSO is a hybrid approach that combines the power of SVM for classification tasks with the optimization capabilities of PSO. Figure 5 shows the Input space to feature space conversion in SVM-PSO using kernel functions. In SVM-PSO, PSO is used to automatically search for the optimal parameters of the SVM algorithm, such as the kernel parameters and regularization parameter. By leveraging PSO's ability to explore and exploit parameter space, SVM-PSO aims to enhance the accuracy and generalization of SVM models by fine-tuning these parameters for improved performance on classification problems.

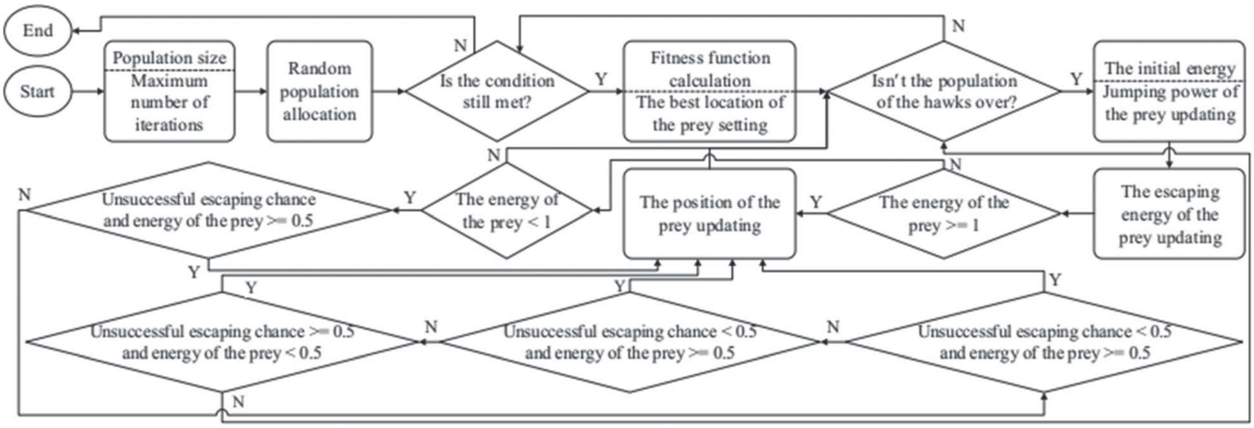
End
Start
Population size
Maximum number of iterations
Random population allocation
Is the condition still met?
Fitness function calculation
The best location of the prey setting
Isn't the population of the hawks over?
The initial energy
Jumping power of the prey updating
The escaping energy of the prey updating
The energy of the prey >= 1
The position of the prey updating
The energy of the prey < 1
Unsuccessful escaping chance and energy of the prey >= 0.5
Unsuccessful escaping chance >= 0.5 and energy of the prey < 0.5
Unsuccessful escaping chance < 0.5 and energy of the prey >= 0.5
Unsuccessful escaping chance < 0.5 and energy of the prey >= 0.5
Y
N

Figure 4. Flowchart illustrating HHO process.

Table 5. Features selected by the HHO algorithm.

	id.orig_h	id.resp_h	service	local_resp	missed_bytes	orig_ip_bytes
0	17578	8081	0.000005	0	2	0
1	17576	8081	0.000002	0	2	0
2	17832	37215	0.000005	0	2	0
3	17576	8081	0.000003	0	2	0
4	17576	8081	0.000002	0	2	0
…	…	…	…	…	…	…
99994	17576	8081	0.000002	0	2	0
99995	17576	8081	0.000006	0	2	0
99996	17576	8081	0.000002	0	2	0
99997	17576	8081	0.000002	0	2	0
99998	17576	8081	0.000005	0	2	0

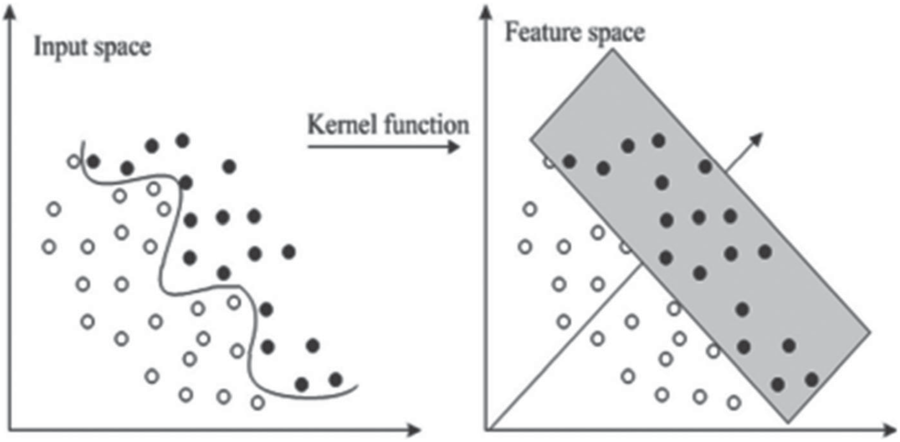

Figure 5. Input space to feature space conversion in SVM-PSO using kernel functions.

3.4. Machine learning methods

A ML model is a mathematical representation of a problem that learns patterns and relationships from data to make predictions or decisions. It involves selecting an appropriate algorithm, training the model on a labelled dataset, and fine-tuning its parameters to achieve optimal performance. The model then undergoes validation and testing on new, unseen data to ensure its generalization ability. ML models can range from simple linear regression to complex neural networks, and they're widely used across various domains to automate tasks, gain insights from data, and improve decision-making processes. Customized feature extraction methods play a crucial role in achieving accurate intrusion detection while mitigating false alarms. We employed three ML algorithms to determine the most effective optimization approach among the mentioned options.

3.4.1. Logistic regression

Logistic Regression is a binary classification algorithm used to predict the probability of an instance belonging to a certain class. Figure 6 shows the Schematic diagram of logistic regression, it models this probability using the logistic function, which transforms input features through a weighted sum. The model's parameters are learned from training data by minimizing the log loss (cross-entropy) between predicted probabilities and actual class labels. The resulting model can then

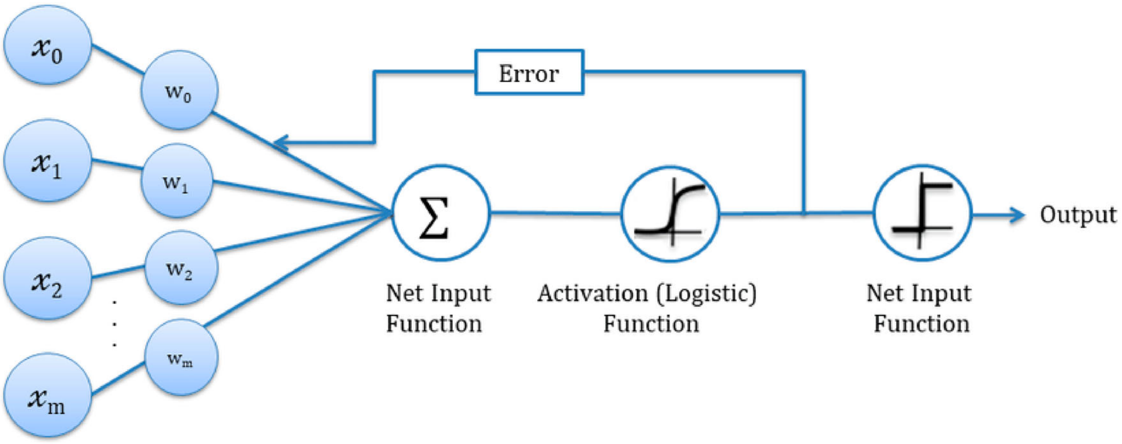

Figure 6. Schematic diagram of logistic regression.

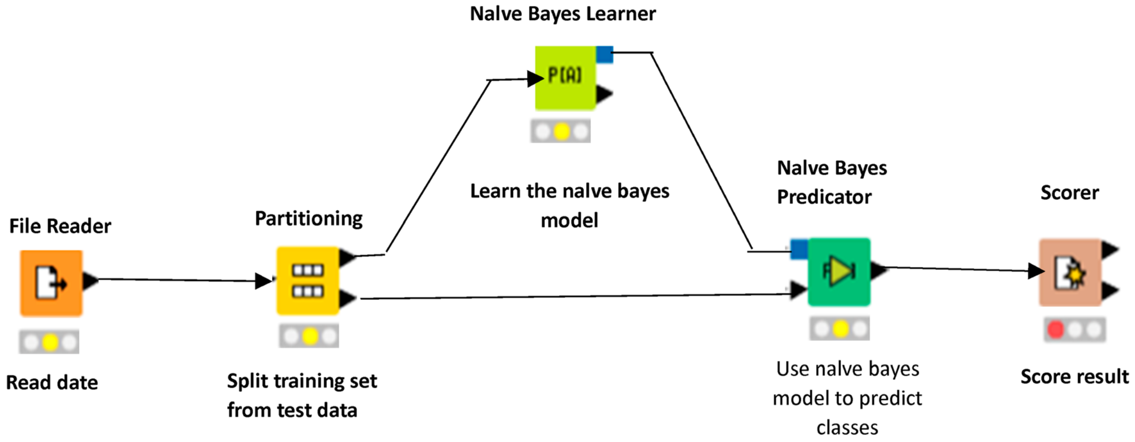
Nalve Bayes Learner
P(A)
Learn the nalve bayes model
File Reader
Read date
Partitioning
Split training set from test data
Nalve Bayes Predicator
Use nalve bayes model to predict classes
Scorer
Score result

Figure 7. Schematic diagram of Naïve Base classifier.

make predictions by comparing predicted probabilities to a threshold, typically 0.5. Logistic Regression is widely used for its simplicity, interpretability and effectiveness in various fields where binary classification is required.

3.4.2. Naive Bayes classifier

The Naive Bayes classifier is a probabilistic algorithm used for classification tasks. It's based on Bayes' theorem and the assumption of feature independence, often considered naive but simplifying. Figure 7 shows the Schematic diagram of logistic regression, it calculates the probability of an instance belonging to a particular class given its features. The classifier estimates class probabilities by multiplying conditional probabilities of individual features given the class. Naive Bayes is especially useful for text classification and spam filtering, where it models word frequencies. While the independence assumption might not hold in all cases, Naive Bayes is computationally efficient, interpretable and performs well on certain types of data.

3.4.3. Decision tree classifier

A Decision Tree Classifier is a ML algorithm used for classification tasks. It operates by recursively partitioning the dataset into subsets based on the values of input features, leading to a tree-like structure of decisions and outcomes. Figure 8 shows the Process to implement decision tree for intrusion detection, at each internal node of the tree, a feature is chosen as a split criterion, and the data is divided into branches based on its possible values. This process continues until a stopping condition is met, such as a maximum tree depth or a minimum number of instances per leaf. The leaves of the tree represent the predicted class labels. Decision trees are intuitive, easy to visualize, and can handle both categorical and numerical features.

3.5. Ensemble model

An ensemble model is a ML approach that combines the predictions of multiple individual models to improve overall performance and accuracy. Ensemble methods often involve training multiple models with different initializations, subsets of data, or algorithm variations, and then combining their predictions through techniques like averaging, voting, or weighted averaging. Examples of ensemble methods include Random Forests (combining decision trees), Gradient Boosting (iteratively improving weak learners) and AdaBoost (boosting weak learners). Ensemble models are known

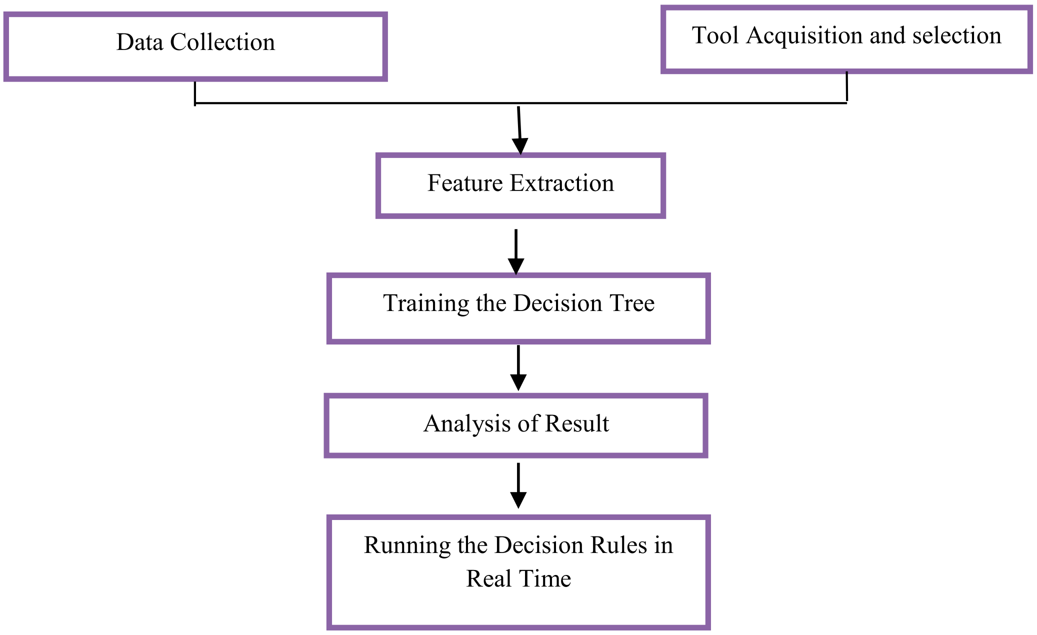
Data Collection
Tool Acquisition and selection
Feature Extraction
Training the Decision Tree
Analysis of Result
Running the Decision Rules in Real Time

Figure 8. Process to implement decision tree for intrusion detection.

for their ability to reduce overfitting, enhance generalization and produce more reliable results, making them popular in various ML tasks.

3.5.1. XG boost

Extreme Gradient Boosting (XG Boost) is a member of the boosting algorithm family and is a practical implementation of the gradient boosting approach. In the case of classification tasks, XGBoost constructs numerous trees in an iterative manner, utilizing knowledge from previously developed trees. This learning technique leverages errors from previous trees to enhance accuracy in subsequent iterations. To mitigate bias and the risk of overfitting, XGBoost incorporates the L1 (Least Absolute Shrinkage and Selection Operator) and L2 (Ridge Regression) regularization algorithms.

3.5.2. Random forest

Random Forest is an ensemble classification method comprising a multitude of Decision Tree classifiers. Through the construction of numerous decision trees on the training dataset and employing majority voting, the ultimate class prediction is determined, as depicted in Figure 9. Consequently, it yields enhanced and reliable predictions, leading to improved system performance in accuracy, recall, precision and false alarm rate.

3.5.3. Adaboost

The AdaBoost ensemble model classifier is a ML algorithm designed for classification tasks. It combines the predictions of multiple weak classifiers in an iterative manner, assigning greater weight to incorrectly classified instances to progressively improve accuracy. During each iteration, a new weak classifier is trained on a modified dataset where instance weights are adjusted. The final prediction is determined by aggregating the weighted predictions of all weak classifiers. AdaBoost's ability to focus on challenging instances and adaptively adjust instance weights results in a powerful ensemble model that performs well on a variety of classification problems. Below is the description of the Adaboost algorithm.

Algorithm 3 Adaboost algorithm.

```
1. Input: training data set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, with labels $y_j \in \{+1, -1\}$
2. Initialize the weight of the training samples $w_i(1) = \frac{1}{n}, i = 1, \ldots, n$
3. Do while t $= 1, \ldots$ ,T
    • For each feature, train a classifier hj which is restricted to using a single feature: hj
    • Calculate the error of the weak classifier: $\in_j = \sum_{j=1}^{n} w_i |h_j(x_i) - y_i|$
    • Choose the classifier, ht with the lower error $\in_1$
    • Update the weights of the training samples: $w_{i+1} = w_{i+1}\beta_t^1 - e_{\overline{C_1}}$ Where ei $=$ 0 if examples xi is classified correctly, ei $=$ 1 otherwise, $\beta_t = \frac{\varepsilon_t}{1} - \varepsilon_1 and C_1$ is a normalization constant.
4. Create a strong classifier:
    $H(x) = \begin{cases} 1\ if \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t & where\ \alpha_t = \frac{log1}{\beta_t} \\ 0\ otherwise \end{cases}$
```

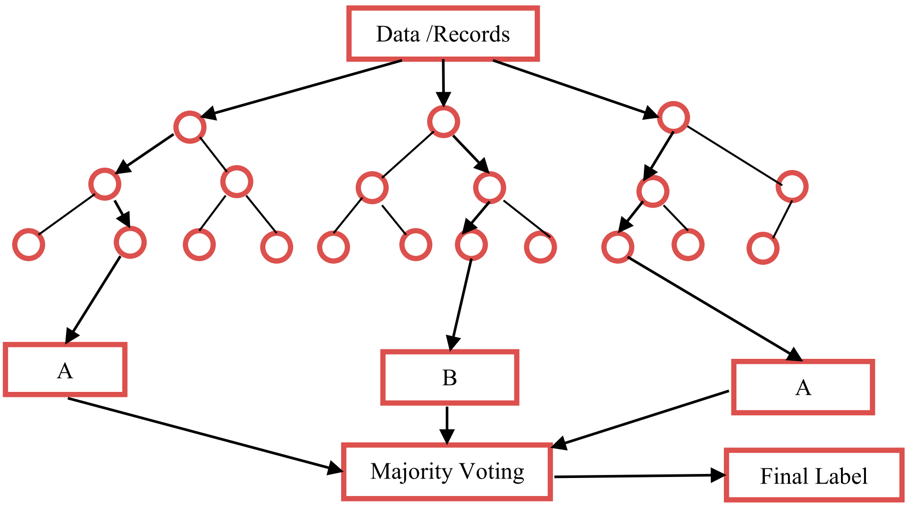

Figure 9. Schematic diagram of random forest classifier.

3.6. Proposed algorithm

Algorithm 4 Proposed Algorithm

```
Input: IoT23 dataset.
Step 1: Begin
Step 2: Feature Selection from the dataset
    Step 2.1: Using WOA
    Step 2.2: Using HHO
    Step 2.3: Using PSO
    Step 2.4: Using SVM-PSO
Step 3: Splitting dataset into training _data and testing _data.
Step 4: Build ML models for comparing the result of optimization
        algorithms.
    Step 4.1: Compare using LR Classifier.
    Step 4.2: Compare using DT Classifier.
    Step 4.3: Compare using DT Classifier.
Step 5: Performed Attack detection using Ensemble models
    Step 5.1: Detection using XGBoost classifier
    Step 5.2: Detection using Random forest classifier
    Step 5.3: Detection using AdaBoost classifier
Step 6: Comparison of the ensemble models
Step 7: ENDTop of Form
```

3.7. Performance parameters

Performance parameters in ML approaches encompass various metrics to assess model effectiveness. These include accuracy, measuring overall correct predictions; precision and recall, evaluating false positives and false negatives; and the F1 score, balancing precision and recall. The performance parameters used by this work are tabulated in Table 6.

Table 6. Performance parameters.

Performance metrics	Equation
Accuracy	$\frac{TP+FP}{TP+FP+TN+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-score	$2 \times \frac{Precicion \times Recall}{Precicion \times Recall}$

4. Results and analysis

4.1. Hardware and software setup

The system makes use of an IoT dataset comprising 21 attributes. To ensure consistent computational performance, Google Colaboratory and Microsoft Windows 10 are selected for this study. The setup includes an Intel Core i7-6850 K processor with a clock speed of 3.60 GHz and 12 cores, along with an NVIDIA GeForce GTX 1080 Ti GPU with a 2760 4MB memory. The dataset is partitioned into a training set, constituting 80% of the data, and a test set, encompassing the remaining 20%.

4.2. Experimental results

Among the evaluated feature selection methods in the table, the HHO algorithm stands out as the most effective for enhancing ML classification models. In direct comparison with alternative approaches, HHO consistently yields superior results. These findings underscore HHO’s proficiency in selecting pertinent features that significantly contribute to the model’s accuracy and predictive capabilities. This outcome highlights the algorithm’s potential for optimizing feature subsets, thereby elevating the overall performance of the classification models. Table 7 shows the results attained by different machine learning models.

The outcomes of the predictions indicate that our proposed feature selection method, utilizing the Harris

Table 7. Comparing the result of optimization algorithm.

Meta Heuristic Algorithm	Machine Learning Model	Accuracy	Precision	Recall	F1 Score	False Positive Rate
HHO	Logistic Regression	100	100	100	100	0
	Decision Tree	100	100	100	100	0
	Naïve Bayes	99.98	99.99	99.98	99.98	0.010
WOA	Logistic Regression	95.07	95.43	95.07	95.09	0.10120
	Decision Tree	98.98	98.92	98.78	98.35	0.0132
	Naïve Bayes	98.97	98.87	98.79	98.45	0.01325
PSO	Logistic Regression	95.09	95.48	95.09	95.13	0.10
	Decision Tree	98.76	98.91	98.98	98.35	0.01312
	Naïve Bayes	97.93	97.99	97.93	97.98	0.0141
SVM-PSO	Logistic Regression	95.17	95.61	95.61	95.23	0.10
	Decision Tree	98.97	98.97	98.92	98.95	0.0138
	Native Baves	98.67	98.67	98.97	98.97	0.0132

Table 8. Comparing the result of random forest with another ensemble model.

Meta HeuristicAlgorithm	Ensemble Model	Accuracy	Precision	Recall	F1 Score	False Positive Rate
HHO	XG Boostclassifier	97.65	97.75	97.75	97.65	0.0142
	Random Forest classifier	**99.97**	**99.98**	**99.97**	**99.97**	**0.0101**
	AdaBoost Classifier	97.99	97.89	97.98	97.88	0.01413
WOA	XG Boostclassifier	97.97	97.97	99.94	99.97	0.0142
	Random Forest classifier	99.93	99.87	99.93	99.92	0.0123
	AdaBoost Classifier	95.07	95.07	95.07	95.09	0.10
PSO	XG Boostclassifier	97.93	97.93	97.93	97.93	0.013
	Random Forest classifier	99.87	99.88	99.88	99.87	0.0135
	AdaBoost Classifier	98.93	98.91	98.91	98.85	0.014
SVM-OSO	XG BoostClassifier	98.95	98.93	98.93	98.95	0.0145
	Random Forest classifier	99.87	99.91	99.91	99.86	0.01236
	AdaBoost Classifier	97.17	97.17	97.17	97.23	0.0140

Note: Bold values indicate proposed value results.

Hawks Optimization algorithm in combination with the random forest classifier, yields the most favourable results when contrasted with alternative approaches. This amalgamation of techniques consistently demonstrates superior performance across various evaluation metrics. The synergy between the Harris Hawks Optimization algorithm and the random forest classifier showcases their collective potential in enhancing predictive accuracy and classification capabilities. These results accentuate the effectiveness of this combined approach in selecting salient features that substantially contribute to the model's robustness and precision. In essence, the study underscores the notable advantages of leveraging the Harris Hawks Optimization algorithm alongside the random forest classifier for optimizing feature selection, ultimately leading to enhanced outcomes in predictive modelling tasks. Table 8 compares the result of random forest with another ensemble model.

5. Conclusion

The surge in IoT devices underscores the urgency of fortifying the security and integrity of interconnected systems. The exploration of intrusion detection within the IoT landscape reveals the limitations of traditional rule-based systems in tackling the dynamic and diverse nature of threats. This has propelled the integration of ML techniques to bolster detection capabilities. The paper's emphasis on tailored feature extraction techniques and the utilization of diverse ML algorithms highlights the potential for accurate and efficient intrusion detection in IoT environments. The demonstrated success of ensemble methods further accentuates the viability of combining algorithmic strengths for enhanced robustness. The attainment of a remarkable 99.97% accuracy through the fusion of random forest and the Harris-Hawks Optimizer underscores the promising advancements in this domain. This paper underscores the crucial role of ML in countering the evolving challenges of intrusion detection in the intricate and interconnected world of IoT.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

[1] Acharya, A. D., & Patil, S. N. IoT based health care monitoring kit. In: 2020 fourth international conference on computing methodologies and communication (ICCMC). IEEE; 2020 March. p. 363–368.

[2] Atlam HF, Wills GB. IoT security, privacy, safety and ethics. Digital twin technologies and smart cities; 2020. p. 123–149.

[3] Chaudhary A, Kolhe S, Kamal R. An improved random forest classifier for multi-class classification. Inf Process Agric. 2016;3(4):215–222. doi:10.1016/j.inpa.2016.08.002

[4] Charbuty B, Abdulazeez A. Classification based on decision tree algorithm for machine learning. J Appl Sci Technol Trends. 2021;2(01):20–28. doi:10.38094/jastt20165

[5] Essa FA, Abd Elaziz M, Elsheikh AH. An enhanced productivity prediction model of active solar still using artificial neural network and Harris Hawks optimizer. Appl

Therm Eng. 2020;170:115020. doi:10.1016/j.applthermaleng.2020.115020

[6] Eid HF. Binary whale optimisation: an effective swarm algorithm for feature selection. Int J Metaheuristics. 2018;7(1):67–79. doi:10.1504/IJMHEUR.2018.091880

[7] Gibert D, Mateu C, Planes J. The rise of machine learning for detection and classification of malware: research developments, trends and challenges. J Netw Comput Appl. 2020;153:102526. doi:10.1016/j.jnca.2019.102526

[8] Hashmat F, Abbas SG, Hina S, et al. An automated context-aware IoT vulnerability assessment ruleset generator. Comput Commun. 2022;186:133–152. doi:10.1016/j.comcom.2022.01.022

[9] Houssein EH, Gad AG, Hussain K, et al. Major advances in particle swarm optimization: theory, analysis, and application. Swarm Evol Comput. 2021;63:100868. doi:10.1016/j.swevo.2021.100868

[10] Heidari AA, Mirjalili S, Faris H, et al. Harris hawks optimization: algorithm and applications. Future Gener Comput Syst. 2019;97:849–872. doi:10.1016/j.future.2019.02.028

[11] Le TTH, Oktian YE, Kim H. XGBoost for imbalanced multiclass classification-based industrial internet of things intrusion detection systems. Sustainability. 2022;14(14):8707. doi:10.3390/su14148707

[12] Lv Z, Qiao L, Kumar Singh A, et al. AI-empowered IoT security for smart cities. ACM Trans Internet Technol. 2021;21(4):1–21.

[13] Li J, Pan Z. Network traffic classification based on deep learning. KSII Trans Internet Inf Syst. 2020;14(11):4246–4267.

[14] Man Z, Li J, Di X, et al. Double image encryption algorithm based on neural network and chaos. Chaos, Solitons Fractals. 2021;152:111318. doi:10.1016/j.chaos.2021.111318

[15] Mahmood MT, Ahmed SRA, Ahmed MRA. Using machine learning to secure IOT systems. In: 2020 4th international symposium on multidisciplinary studies and innovative technologies (ISMSIT). Istanbul: IEEE; 2020 Oct. p. 1–7.

[16] Nimbalkar P, Kshirsagar D. Feature selection for intrusion detection system in internet-of-things (IoT). ICT Express. 2021;7(2):177–181. doi:10.1016/j.icte.2021.04.012

[17] Otoum Y, Liu D, Nayak A. DL-IDS: a deep learning–based intrusion detection framework for securing IoT. Trans Emerg Telecommun Technol. 2022;33(3):e3803. doi:10.1002/ett.3803

[18] Roopak M, Tian GY, Chambers J. Deep learning models for cyber security in IoT networks. In: 2019 IEEE 9th annual computing and communication workshop and conference (CCWC). Las Vegas, NV: IEEE; 2019 Jan. p. 0452–0457.

[19] Ravindranath V, Ramasamy S, Somula R, et al. Swarm intelligence-based feature selection for intrusion and detection system in cloud infrastructure. In: 2020 IEEE congress on evolutionary computation (CEC). Glasgow: IEEE; 2020 July. p. 1–6.

[20] Saheed YK, Abiodun AI, Misra S, et al. A machine learning-based intrusion detection for detecting internet of things network attacks. Alexandria Eng J. 2022;61(12):9395–9409. doi:10.1016/j.aej.2022.02.063

[21] Sahu AK, Sharma S, Tanveer M, et al. Internet of things attack detection using hybrid deep learning model. Comput Commun. 2021;176:146–154. doi:10.1016/j.comcom.2021.05.024

[22] Smys S, Basar A, Wang H. Hybrid intrusion detection system for internet of things (IoT). J ISMAC. 2020;2(04):190–199. doi:10.36548/jismac.2020.4.002

[23] Valdiviezo-Diaz P, Ortega F, Cobos E, et al. A collaborative filtering approach based on Naïve Bayes classifier. IEEE Access. 2019;7:108581–108592. doi:10.1109/ACCESS.2019.2933048

[24] Verma A, Ranga V. Machine learning based intrusion detection systems for IoT applications. Wirel Pers Commun. 2020;111:2287–2310. doi:10.1007/s11277-019-06986-8

[25] Wang W, Dumont F, Niu N, et al. Detecting software security vulnerabilities via requirements dependency analysis. IEEE Trans Software Eng. 2020;48(5):1665–1675. doi:10.1109/TSE.2020.3030745

[26] Khan RA, Khan SU, Alzahrani M, et al. Security assurance model of software development for global software development vendors. IEEE Access. 2022;10:58458–58487. doi:10.1109/ACCESS.2022.3178301

[27] Kumar P, Gupta GP, Tripathi R. Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for IoT networks. Arab J Sci Eng. 2021;46:3749–3778. doi:10.1007/s13369-020-05181-3

[28] Zhang Y, Ni M, Zhang C, et al. Research and application of AdaBoost algorithm based on SVM. In: 2019 IEEE 8th joint international information technology and artificial intelligence conference (ITAIC). Chongqing: IEEE; 2019 May. p. 662–666. 10.1109/ITAIC.2019.8785556.

[29] Zhaofeng M, Jialin M, Jihui W, et al. Blockchain-based decentralized authentication modeling scheme in edge and IoT environment. IEEE Internet Things J. 2020;8(4):2116–2123. doi:10.1109/JIOT.2020.3037733

[30] Zou X, Hu Y, Tian Z, et al. Logistic regression model optimization and case analysis. In: 2019 IEEE 7th international conference on computer science and network technology (ICCSNT). Dalian: IEEE; 2019 Oct. p. 135–139. 10.1109/ICCSNT47585.2019.8962457.