Hybrid Optimization and Machine Learning Models for IoT Intrusion Detection in Smart Homes

Shaoying Li
Sanda University, Shanghai 201209, China
E-mail: lsy@sandau.edu.cn

Keywords: smart home, random forest classification, decision tree classification, rhizostoma optimization algorithm, dandelion optimization

Received: December 1, 2024

The rapid technological growth of IoT devices has significantly enhanced connectivity and usability in smart homes. However, these advantages come with substantial security risks. This study presents a hybrid machine learning approach to predict smart home intrusions using two base classifiers Random Forest Classification (RFC) and Decision Tree Classification (DTC) optimized with Dandelion Optimization (DO) and Rhizostoma Optimization Algorithm (ROA). These combinations result in four hybrid models: RFDO (RFC + DO), RFRO (RFC + ROA), DTDO (DTC + DO), and DTRO (DTC + ROA). A benchmark intrusion detection dataset tailored for smart home environments was used to train and evaluate these models. Among all models, DTRO achieved the highest accuracy of 0.981 during testing, followed by RFRO with 0.977, while the base RFC model performed the worst with 0.915, outperforming the other models in terms of precision, recall, F1-score, and Matthews correlation coefficient. This comparative analysis shows that integrating optimization techniques into traditional classifiers significantly enhances intrusion detection capabilities in resource-constrained IoT environments. The proposed models are suitable for real-time applications in smart home cybersecurity.

Povzetek: Študija predstavlja hibridne modele strojnega učenja za zaznavanje vdorov v pametne domove.

1 Introduction

Smart household appliances and big-ticket items like the Industrial Internet of Things (IIoT) and the Internet of Things in Power Systems (IOTIPS) are only two examples of the expanding applications for the Internet of Things (IoT) [1]. By 2020, there will likely be two billion IoT devices, compared to just eight million in 2017 [2]. This presents a variety of security problems. IoT devices are readily made into targets or intermediates for attacks because of their low processing power, sensitive data that is not encrypted, exposed ports that are not secured, and firmware vulnerabilities that occur from not updating the device for a long time [3].

One efficient element of network security, particularly for wireless networks, is an intrusion detection system (IDS) [4]. Nevertheless, implementing conventional IT security methods (like firewalls) in Internet of Things settings might be challenging. This is caused in part by the IoT's dynamic environment and in part by a shortage of processing power. New solutions have been made possible by the advancement of machine learning technology, which uses continual learning to adapt to changes in its surroundings. Nevertheless, there are several issues with machine learning's present implementation in IoT intrusion detection [5,6]. First off, more sophisticated attack types are not taken into consideration, and the sorts of intrusion attempts examined are very limited [7]. Second, because extracting a large variety of features would require a lot of resources, it is highly laborious to discover useful features to train the machine learning model manually during the data processing phase [8,9]. Because of this, a simple technique for automatically extracting a limited set of attributes is required so that machine learning can identify different types of IoT assaults [10].

Many studies on the security of the IoT are currently being conducted. The DDoS assault was examined by [11]. By focusing on a small set of patterns within the attacking network traffic, they can extract features. In the end, they accomplish high-accuracy detection by effectively utilizing machine learning technology. [12] work focuses on botnet identification and employs deep learning techniques to do this; however, it hasn't been thoroughly tested in real environments. presented a framework for anomaly detection in the Internet of Things to detect botnet attacks [13], which was validated in several network protocols (WiFi and ZigBee) to implement IDS into the heterogeneous networks. A lightweight detector technique to identify the Hello flood and Sybil assaults in the Internet of Things is proposed by Stephen and Arockiam [14]. They utilize a method to assess the IDS intrusion rate, and they use the low-power and lossy network (RPL) routes as their routing protocol. The IDS suggested in the linked research mentioned above only takes into account one kind of assault and ignores other attack vectors [15].

Anthi et al. [7] suggested an intrusion detection system that uses three layers of machine learning to provide extremely accurate identification of device classes, traffic abnormalities, and assault kinds. However, extracting a lot of characteristics might use up a lot of resources, particularly in IoT situations where resources are scarce. [16] presented a technique that successfully increased the attack detection rate evaluated on the AWID dataset by merging normalized gain with Particle Swarm Optimization (PSO) to choose the best feature. [17] have recently produced excellent study findings. The suggested Stacked Auto Encoder (SAE) network is utilized for unsupervised learning, whereby traffic labels are automatically determined and features are compressed using deep learning [18]. Regarding the Aegean Wi-Fi network attack dataset, it attains a high degree of accuracy. Nevertheless, the suggested feature reduction approach is computationally intensive and challenging to implement for the IoT with limited resources [19].

This study hypothesizes that hybrid machine learning models combining Decision Tree Classification (DTC) and Random Forest Classification (RFC) with metaheuristic optimization algorithms namely, Dandelion Optimization (DO) and Rhizostoma Optimization Algorithm (ROA) can significantly improve intrusion detection accuracy in smart home IoT environments compared to traditional machine learning approaches. The primary objectives of this study are:

- To develop hybrid models: RFDO, RFRO, DTDO, and DTRO by integrating optimization techniques with RFC and DTC.
- To evaluate the models using a smart home IoT intrusion dataset and multiple metrics: Accuracy, Precision, Recall, F1-Score, and MCC.
- To compare the proposed methods with baseline and state-of-the-art models from existing literature.
- To analyze the strengths, limitations, and interpretability of the models using SHAP sensitivity analysis.

The ultimate goal is to build a lightweight yet highly accurate detection system for smart homes capable of real-time operation in constrained environments. Fig. 1 illustrates the overall research workflow, detailing each phase from dataset preprocessing to model training, optimization, evaluation, and final performance interpretation.

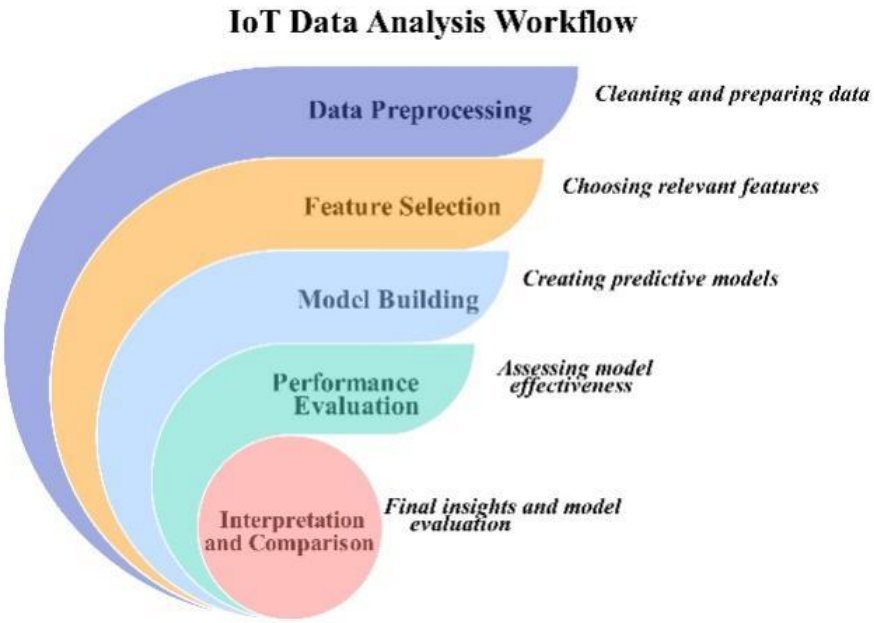

Figure 1: The overall research workflow.

Suggested problem statement revision: While IoT devices have transformed smart homes by enhancing connectivity and usability, their inherent resource constraints and dynamic environments pose significant challenges for effective intrusion detection. Existing machine learning techniques often struggle with limitations such as insufficient precision, high computational costs, and inadequate feature extraction methods, which hinder their real-time applicability in smart home cybersecurity. This study addresses these challenges by developing hybrid models that integrate traditional classifiers, RF and DT, with advanced optimization algorithms (DO and ROA). These combinations aim to improve detection accuracy and computational efficiency, while also leveraging automated attribute extraction to better capture relevant intrusion features in resource-limited IoT settings.

2 Study methods and techniques

2.1 Rain Forest Classification (RFC)

A random forest (RF) model is constructed by assembling a set of tree predictors [20]. Every tree has grown using the methodology outlined in:

1. During the bootstrap phase, a random subset of the training dataset is selected as a local training set for tree development [21]. The remaining samples from the training dataset make up the out-of-bag (OOB) set, which is used to assess the random forest's (RF) goodness of fit.

2. During the expansion phase, the local training set at each node is partitioned by the value of a randomly chosen variable from a collection of variables, which allows the tree to grow. The classification and regression tree (CART) strategy is employed in the optimal split procedure.

3. All trees are developed to their maximum potential without any trimming done.

It is necessary to add random variables in both the bootstrapping and growth phases. It is assumed that these variables have a uniform distribution and show independence among the trees [22]. Because of this, each tree can be thought of as a distinct sample taken from the whole set of tree predictors for a given training dataset. An instance travels around every tree in the forest until it reaches a terminal node during the prediction phase, which assigns it a class [23]. Every tree's forecast is presented to a vote, and at the end of the process, the forest assigns a class based on which tree got the most votes. When there are ties, the problem is solved by random selection. The paper will provide a feature contribution approach that necessitates developing a probabilistic interpretation of the forest prediction process in the part that follows. The set of classes is represented by $C = \{C1, C2, \ldots, CK\}$, and the set is denoted by ΔK.

$$\Delta_k = \left\{(P_1, \ldots, P_K): \sum_{K=1}^{K} P_k = 1 \; and \; P_k \geq 0\right\} \quad (1)$$

One can conceptualize an element in ΔK as a probability distribution over C. In this framework, a mapping is established between tree predictions and the set ΔK of probability measures on C.

$$\widehat{Y_k} = \frac{1}{T}\sum_{t=1}^{T} \widehat{Y_{i,t}} \quad (2)$$

The total quantity of trees in the forest (T) should be indicated. Consequently, $\hat{y}_i$ is a member of ΔK, and the random forest prediction, for instance i corresponds to class Ck, where $\hat{y}_i$ is k_{th} coordinate is greatest. The RFC model's structure is shown in Fig. 2.

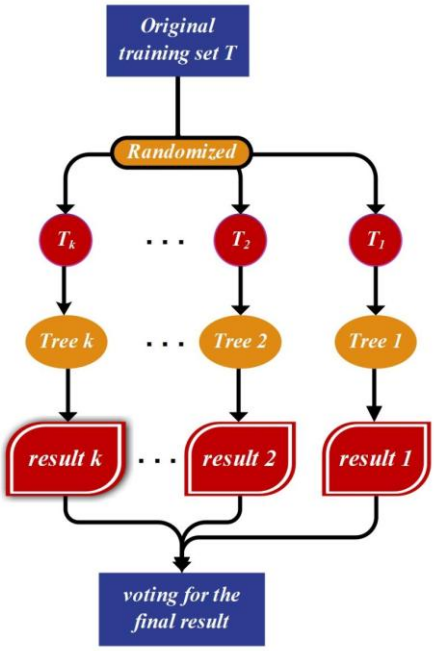

Figure 2: RFC structure

2.2 Decision Tree Classification (DTC)

Decision tree learning connects observations about an object with conclusions about its target value by using a decision tree as a prediction model [24]. Using a greedy depth-first or breadth-first strategy, the decision tree method is a data mining induction methodology that repeatedly splits a dataset of records until each data item is assigned to a certain class [25], [26].

A decision tree's internal, root, and leaf nodes are arranged in a manner resembling a flowchart's tree architecture. The top node serves as the root; class labels are added to each leaf node; branches indicate test condition outcomes; and each internal node represents a test condition based on an attribute [27]. A divide-and-conquer tactic is used to build the decision tree, representing a decision rule for each route. Usually, a greedy approach that starts at the top is used. The decision tree classification approach consists of two stages: tree creation and tree trimming. A top-down approach is used to build the tree, and divisions are performed repeatedly until each data item has a class label attached to it [28]. The training dataset needs to be continually scanned, which means that this process demands a lot of computing resources. Tree pruning, on the other hand, is carried out from the bottom up to address overfitting in the tree to raise the prediction and classification accuracy of the algorithm. Overfitting in decision trees can lead to misclassification mistakes. The flowchart of the DTC model is shown in Fig. 3:

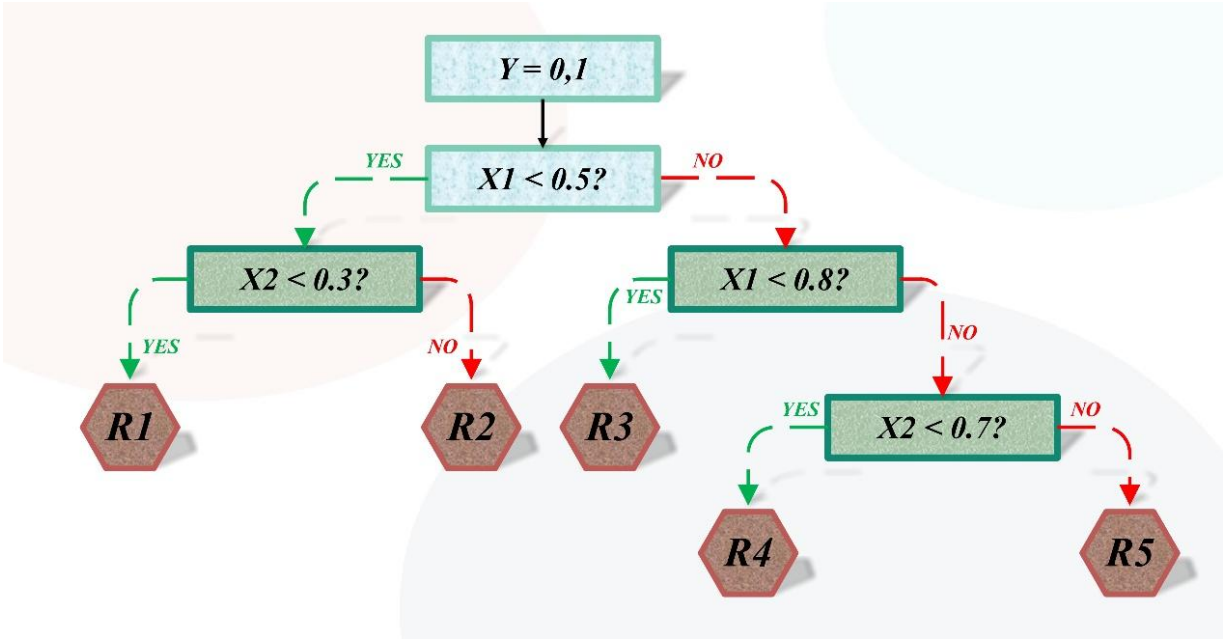

Figure 3: Flowchart for the DTC model

2.3 Dandelion Optimization (DO)

Certain plants, such as dandelions, release their seeds into the wind. Dandelion seeds travel through the following three stages:

- During the ascent phase, wind and sunlight generate a vortex above the dandelion seed, causing it to rise. However, without these eddies above the seeds, restricted searches become impractical on rainy days.
- As soon as the seeds enter the falling stage and attain a certain height, they begin to steadily sink.
- During the landing stage, dandelion seeds fall at random onto a surface, where they develop into young dandelions in response to the weather and wind. Through seed dispersal, dandelions alter their population by the three stages mentioned above.

In the DO algorithm, every dandelions seed denotes a potential solution. The definition of a DO's population is as follows:

$$population = \begin{bmatrix} x_1^1 & \cdots & x_1^{dim} \\ \vdots & \cdots & \vdots \\ x_{pop}^1 & \cdots & x_{pop}^{dim} \end{bmatrix} \quad (3)$$

In this case, the variables' dimensions and population size are indicated by the labels pop and dim, respectively. The following is an explanation of the ith individual Xi. Within the lower bound (LB) and upper bound (UB) of the given issue, every potential solution is generated at random.

$$x_i = rand \times (UB - LB) + LB \quad (4)$$

Here, i is an integer that goes from 1 to pop, and rand is a random number that varies between 0 and 1. The highest fitness value, as determined by DO, designates the top elite and is where the dandelion seed should germinate. The major elite's mathematical formulation, accounting for the lowest value, is as follows:

$$f_{best} = \min\left(f(x_i)\right) \quad (5)$$

$$x_{elite} = x(find(f_{best} == f(x_i))) \quad (6)$$

In this case, $find()$ indicates which two indices have the same value.

Dandelion seeds need to reach a certain height to split out from their parent plants during the rising stage. Two factors that influence the growth of high dandelion seeds are wind speed and air humidity. In this case, the two significant meteorological conditions are:

First scenario

To represent wind speeds on a clear day, the lognormal distribution is used. Specifically, if $Ln\,Y$ follows a normal distribution $N(\mu, \sigma^2)$, where Y represents wind speed, the lognormal distribution effectively captures the variability in wind speeds. The height to which a dandelion seed ascends is primarily influenced by wind speed; higher wind speeds lead to longer dispersal times and greater flight heights.

$$x_{t+1} = x_t + a \times v_x \times v_y \times Ln\,y \times (x_s - x_t) \quad (7)$$

In this instance, x_t denotes the location of the dandelion seed at iteration t, while x_s is the random location selected at iteration t. The formula for the position determined by random generation is shown in Eq. (8):

$$X_s = rand(1, \dim) \times (UB - LB) + LB \quad (8)$$

The distribution of ln Y is lognormal, with parameters $\mu = 0$ and $\sigma^2 = 1$.

$$\ln y = \begin{cases} \frac{1}{y\sqrt{2\pi}} exp\left[-\frac{1}{2\sigma^2}(\ln y)^2\right] & y \geq 0 \\ 0 & y < 0 \end{cases} \quad (9)$$

In this case, y stands for a typical normal distribution with a variance of one and a mean of zero.

$$a = rand() \times (\frac{1}{T^2}t^2 - \frac{T}{2}t + 1) \quad (10)$$

A random disturbance with values between 0 and 1 is represented by the variable a. The lift components of the dandelion are represented by the coefficients v_x and v_y.

$$r = 1/e^{\theta} \quad (11)$$

$$v_x = r\ cos\theta \quad (12)$$

$$v_y = r\ sin\theta \quad (13)$$

Here, values between $[-p, p]$ are randomly assigned to the variable q.

Second Scenario

Rainy days hinder dandelion seeds from effectively rising with the breeze due to humidity and air resistance.

$$x_{t+1} = x_t \times k \quad (14)$$

$$k = 1 - rand() \times q \quad (15)$$

The variable k is used by dandelion to regulate its local search area. The domain (q), which is as follows, may be obtained by using Eq. (16):

$$q = \frac{1}{T^2 - 2T + 1}t^2 - \frac{1}{T^2 - 2T + 1}t + 1 + \frac{1}{T^2 - 2T + 1} \quad (16)$$

In the end, the following is the mathematical formula for a dandelion seed's climbing stage:

$$x_{t+1} = \begin{cases} x_{t+1} = x_t + a \times v_x \times v_y \times lny \\ \quad \times (x_s - x_t)\ randn < 1.5 \\ x_{t+1} = x_t \times k \qquad else \end{cases} \quad (17)$$

The $randn()$ function generates a random integer that is consistent with the normal distribution. Dandelions sprout to a particular height during this stage, after which they gradually decline (exploration phase). The DO algorithm simulates a dandelions' route by using Brownian motion.

$$x_{t+1} = x_t - a \times \beta_t \times (x_{meant} - a \times \beta_t \times x_t) \quad (18)$$

Here, the Brownian motion is shown by β_t.

$$x_{meant} = \frac{1}{pop}\sum_{i=1}^{pop} x_i \quad (19)$$

In the final stage of the Dandelion Optimization (DO) algorithm, the focus shifts to exploitation. The position of the dandelion seed is determined randomly based on the outcomes of the previous two stages. As iterations proceed, the algorithm is expected to converge towards the optimal solution. The culmination of this iterative process results in the global ideal solution:

$$X_{t+1} = x_{elite} + Levy(\lambda) \times a \times (x_{elite} - x_t \times \delta) \quad (20)$$

In this case, x_{elite} stands for the dandelion seed's ideal location.

$$levy(\lambda) = s \times \frac{w \times \sigma}{|t|^{\frac{1}{\beta}}} \quad (21)$$

The constant s has a value of 0.01. The random number in B can range from 0 to 2. The variables t and w are arbitrary integers in the range [0, 1]. The mathematical expression for s is as follows:

$$\sigma = (\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \sin(\frac{\beta - 1}{2})}) \quad (22)$$

Since b is fixed at 1.5, δ may be computed as follows:

$$\delta = \frac{2t}{T} \quad (23)$$

The DO algorithm draws inspiration from the seed dispersal behavior of dandelions, which navigate through various environmental conditions (e.g., wind, humidity) during propagation. In this study, DO is adapted to optimize hyperparameters of machine learning models by simulating this stochastic yet guided exploration. Each "seed" represents a candidate solution (i.e., a parameter set), and the algorithm balances local and global search to iteratively improve model accuracy on IoT intrusion datasets. For instance, in optimizing the Decision Tree's depth and minimum sample split, DO quickly converged to configurations that improved recall without overfitting. Instead of relying solely on theoretical distribution modeling, we emphasize its ability to handle high-dimensional, noisy IoT features effectively and efficiently. Fig. 4 presents the DO process.

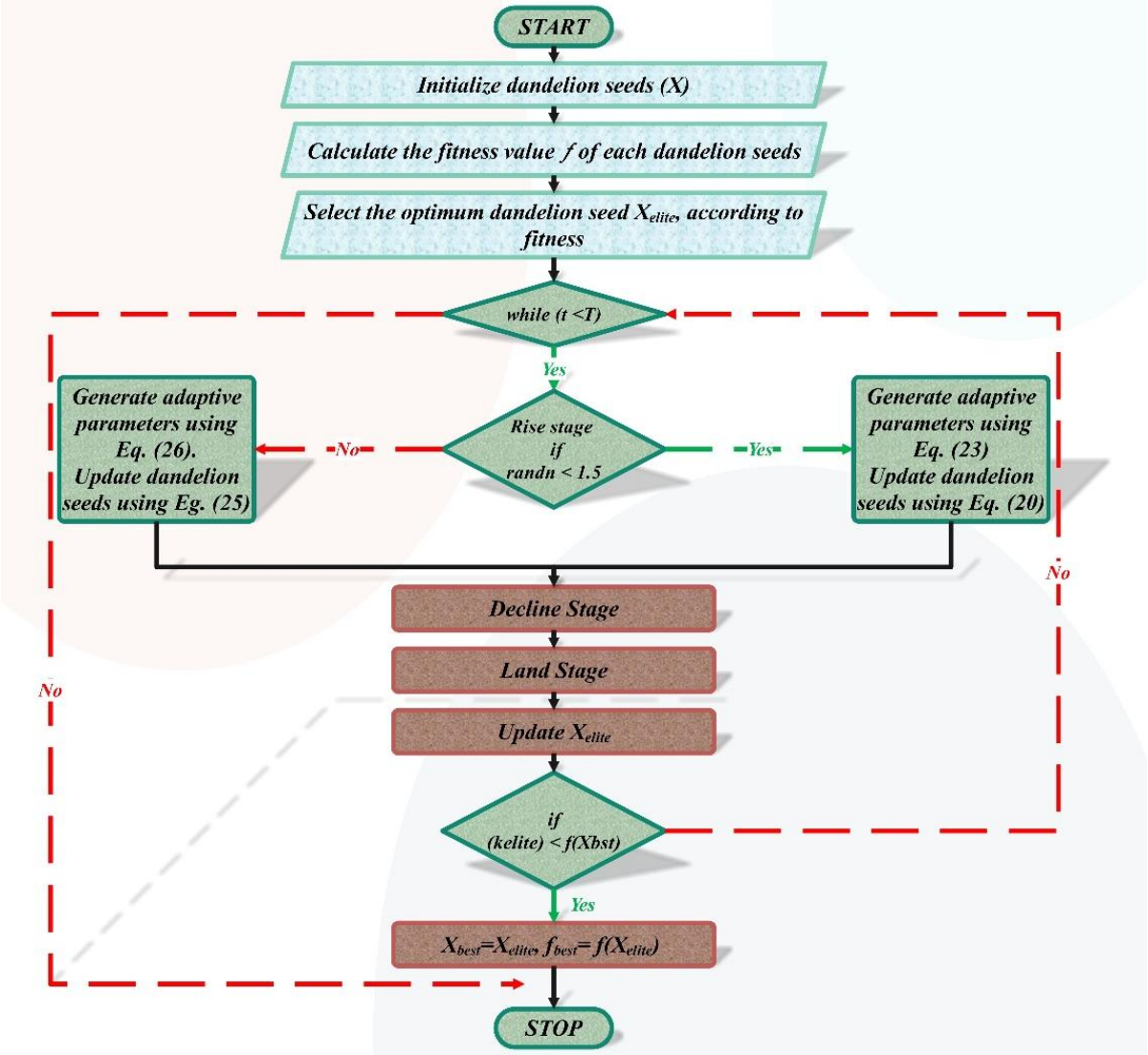

Figure 4: DO process

2.4 Rhizostoma Optimization Algorithm (ROA)

This section explains the principles behind the Rhizostoma octopus optimization method. The mathematical model describes the movement of the Rhizostoma octopus in water. Additionally, it details the approach used by the octopus to locate the best food sources, taking into account its food consumption patterns [29].

Formerly known as Rhizostoma octopus, barrel jellyfish are giants among jellyfish, named for their thick, mushroom-shaped bell that can reach up to 1 meter in diameter and 8 frilly arms underneath. Despite their impressive size, they attract small crabs and young fish seeking protection in their tentacles. Found in the summer months along the southern and western shores of the British Isles, barrel jellyfish are not a concern to humans. Their sting is so mild that some people may not even feel it, and it generally has no significant adverse effects [30].

Jellyfish have several ways of eating; some utilize their tentacles to deliver food to their mouths, while others employ filter-feeding [20], which means they consume whatever the current provides. With a fierce chase, Rhizostoma pulmo captures its target and uses its tentacles to freeze it. R. octopus possesses characteristics that allow them to regulate their motion, in contrast to other jellyfish species that mostly rely on currents and tides to drift in the sea. Because Rhizostoma octopus (R. octopus) is very huge, reaching 27 kg or more, it can swim actively against localized currents. Their swarms, known as blooms, are aided in forming by rising water temperatures and food supplies because they are more resilient than other ocean organisms in situations with low oxygen concentrations and high salinity. Reynolds [21] discovered that the R. octopus uses a variety of search techniques to locate massive amounts of plankton at high concentrations.

Three fundamental rules form the foundation of the suggested algorithm: I Bands for exploration and exploitation: Rhizostoma pulmo is a marine organism that uses many random search algorithms to find the optimum places to feed on large volumes of plankton. Once a swarm is formed, it begins to eat. The two primary bands of a

meta-heuristic algorithm are taken into consideration in the artificial ROA.

The swarm's two main motion types are exploitation and exploration; ii) Rhizostoma moves or searches for food, with a "motion control factor" controlling when to switch between these movements; and iii) the location and related objective function of the swarm determines the quality of food that is found at the current location.

R. octopus population initialization is done at random.

$$x_i^* = \lambda x_i(1 - x_i), 0 \le x_0 \le 1 \quad (24)$$

Let x_i represent the location value of the i_{th} R. octopus. R. octopus's starting population, x_0, is created so that it ranges from 0 to 1, excluding the numbers 0, 0.25, 0.5, 0.75, and 1. 4.0 is the value of the parameter λ . An R. octopus will re-enter the search area from the opposite boundary if it crosses it. Eq. (25) illustrates this returning process.

$$X_{i,d}^* = \begin{cases} (x_{i,d} - Ub_d) + Lb_d, if\, x_{i,d} > Ub_d \\ (x_{i,d} - Lb_d) + Ub_d, if\, x_{i,d} < Lb_d \end{cases} \quad (25)$$

$x_{i,d}$ denotes the location of the i_{th} R. octopus in the $d - th$ dimension, when boundary restrictions are applied, the revised location is indicated by $x_{i,d}^*$. Here, Ub_d and Lb_d are the upper and lower bounds of the d-th dimension in the search space, respectively. ROA mimics the movement of barrel jellyfish, which use both passive and active motion to locate dense plankton areas. ROA applies this concept to hyperparameter search, allowing models to escape local optima by alternating between exploration and exploitation. In our implementation, ROA was particularly effective for fine-tuning the Decision Tree model's structure due to its adaptive control mechanism. Unlike traditional grid search or random search methods, ROA consistently achieved better convergence in fewer iterations, especially in complex, imbalanced IoT datasets where precision and MCC are critical.

2.5 Algorithm justification

In designing an intrusion detection system for smart home IoT environments, it is essential to balance detection accuracy, computational efficiency, and suitability for real-time operation on resource-limited devices. This study employs Decision Tree Classification (DTC) and Random Forest Classification (RFC) as the base models due to their low computational complexity, fast inference time, and high interpretability, which are particularly valuable in security-sensitive and embedded applications. These models also handle noisy, tabular, and imbalanced datasets well, making them practical for real-world intrusion detection. While more complex models such as support vector machines or deep neural networks offer high accuracy, they typically require more computational resources and longer training times, which are often impractical in smart home settings. To further enhance the performance of these base classifiers, two nature-inspired metaheuristic optimization algorithms DO and ROA are applied for hyperparameter tuning. These algorithms were chosen over more traditional optimizers such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), or grid and random search, due to their lower computational overhead, fewer required control parameters, and more stable convergence behavior in high-dimensional search spaces. DO simulates seed dispersal and balances exploration and exploitation with minimal resource demands, while ROA leverages a biologically inspired motion control mechanism that allows adaptive, efficient searching. Both methods are lightweight, require fewer iterations to converge, and are well-suited to constrained environments like smart homes, where computational resources are limited. The combined use of efficient tree-based models and biologically inspired optimizers ensures a scalable, interpretable, and high-performing intrusion detection solution that meets the practical requirements of real-time IoT security systems.

2.6 Performance evaluation metrics

A thorough examination of various factors is essential to determine the effectiveness of a classifier study. Among these factors, accuracy is the most commonly used metric, reflecting the proportion of correctly identified positive suggestions. Recall, on the other hand, measures the model's ability to identify all relevant positive occurrences. Accuracy evaluates the model's overall prediction correctness. By combining accuracy and recall into a single performance metric, the F1 score provides a comprehensive assessment of classifier performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (26)$$

$$Precision = \frac{TP}{TP + FP} \quad (27)$$

$$Recall = TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \quad (28)$$

$$F1\ score\ = \frac{2 \times Recall\ \times\ Precision}{Recall + Precision} \quad (29)$$

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \quad (30)$$

Under these conditions, "TP" and "FP" stand for positive cases correctly identified, while "TN" is negative occurrences, which were correctly forecasted, and "FN" stands for negative events forecasted incorrectly.

3 Data collection

3.1 Data description

This dataset will be very useful in the development of intrusion detection algorithms optimized for smart home environments. Given the growth of the IoT, still expanding, such protection of connected ecosystems against cyber threats is increasingly critical. The dataset

ranges from simple connection statistics and complex behavioral patterns, eliciting the differences between normal operations and possible intrusions. This makes for a great contribution to both researchers and industry professionals by emphasizing the unique features of smart home network activities. It will enable accurate and effective prediction models that could be developed to detect, alert, and mitigate cyber intrusions. The 'Attack' label in the dataset indicates whether a given network connection is considered malicious (attack = 1) or benign (attack = 0). This labeling was not done manually but originates from the curated dataset used in this study a benchmark dataset specifically designed for smart home intrusion detection. The dataset was annotated by domain experts and generated in a controlled testbed environment where simulated attack scenarios (e.g., DDoS, probing, spoofing) were injected alongside normal smart home traffic. Each connection record was labeled using predefined ground-truth scenarios based on system-level logs, intrusion triggers, and traffic signatures. This method ensures a high level of reliability and reproducibility in the labeling process.

- **Duration**: The length of the link, providing information on enduring relationships that could point to ongoing assaults or data espionage.
- **Protocol Type:** The connection protocol is essential for figuring out the type of traffic and spotting vulnerabilities unique to the protocol.
- **Service:** The kind of network service used, which might draw attention to deliberate assaults on particular services connected to the smart home network.
- **Flag:** Connection status information helps spot deviations from the typical behavior of network communications.
- **Src_bytes:** Quantity of data transferred from the source to the destination, which makes it easier to identify large-scale data transfers or probing activity.
- **Dst_bytes:** Quantity of data returned to the source from the destination, which can provide answers to queries or efforts at data exfiltration.
- **Land:** shows if the source and destination are the same, which is usually an indication of reflection or spoofing attacks.
- **Wrong fragment:** The number of fragmented packets, which can be a strategy employed by several attack vectors to get over security safeguards.
- **Urgent:** number of urgently marked packets, which may indicate attempts to give malicious traffic priority.
- **Hot:** shows "hot" signs in the connection, which are frequently connected to exploits or illegal access attempts.
- **Logged in:** Whether or not a user was able to log in successfully, which can help distinguish between permitted and illegitimate use.
- **Num compromised**: Number of compromised circumstances, which indicate how critical an assault or intrusion is.
- **COUNT**: The number of connections to a single server can indicate flooding or scanning-type attacks.
- **The count of Srv**: It may be used to differentiate between targeted attacks against particular providers by considering the number of connections to a same service.
- **Serror rate**: SYN error-producing connection rate. It gives an indication of possible SYN flood assaults.
- **Error rate**: The percent of connections that caused reset errors, an indication of broken connections, or attempted scans.
- **Same SRV rate:** percentage of connections to the same service; helpful in identifying odd patterns of access.
- **Diff SRV rate:** Connection rate to various services, which may be a sign of port scanning or service enumeration activity.
- **SRV diff host rate:** Connection rate to many servers for the same service, which might reveal patterns of spread attacks.
- **DST host count:** number of connections to the same destination site; helpful for examining attacks that are specifically directed at particular devices.
- **DST host SRV count:** number of connections to the destination host's service that provide information about traffic patterns unique to that service.
- **DST host same SRV rate:** Connection rate to the same service on the target host, which is crucial for recognizing attacks that target certain services.
- **DST host diff SRV rate:** Scan or enumeration activity is indicated by the rate of connections to various services on the destination host.
- **Attack:** As the label for model training, it indicates if the connection was a part of an assault (yes or no).

The selected features in this dataset are chosen not only for their statistical value but also for their relevance to intrusion detection in smart home networks, where traffic patterns can be highly variable and susceptible to lightweight attacks. For instance, features like src_bytes, dst_bytes, and logged_in provide insight into typical versus anomalous user behavior for example, excessive data transfers or repeated failed login attempts may indicate brute force or exfiltration attempts. Features such as count, srv_count, same_srv_rate, and dst_host_count reflect traffic volume and distribution, which can help detect scanning, flooding, or botnet activities commonly seen in compromised IoT devices. Although some features such as urgent or wrong_fragment may appear general, they can signal manipulation of lower-layer protocols to bypass filters or cause fragmentation-based denial-of-service (DoS) attacks. These subtle traffic characteristics, when observed collectively, enable the model to distinguish between legitimate smart home activity and a variety of intrusion types, even under resource constraints.

Fig. 5 represents the confusion plot of the input and output variables. A value of 1 indicates a direct relationship, whereas a value of -1 signifies an inverse relationship. For instance, as the attack decreases, the flag increases, and similarly, as the error rate decreases, the flag increases. While the error rate is growing, it can be

viewed that both same_srv_rate and dst_host_same_srv_rate are decreasing. Another measure can be viewed as count decreases correspond to an increase in logged_in. Feature Correlation Analysis: To better understand feature interdependencies and guide model design, a correlation heatmap of the dataset was created (Fig. 5). Several variables, such as same_srv_rate, dst_host_same_srv_rate, and srv_count, show strong positive correlations, suggesting they may represent overlapping behavioral patterns in network traffic. Notably, features like urgent, wrong_fragment, and land have near-zero correlations with most other variables, indicating limited predictive value and potentially being dropped or down-weighted during feature selection. More importantly, features such as src_bytes, count, and logged_in demonstrate moderate-to-strong correlation with the attack label, aligning with their significant contributions seen later in SHAP analysis. These correlation insights informed both the optimization algorithm focus and tree model regularization, ensuring better generalization and avoiding overfitting to redundant features.

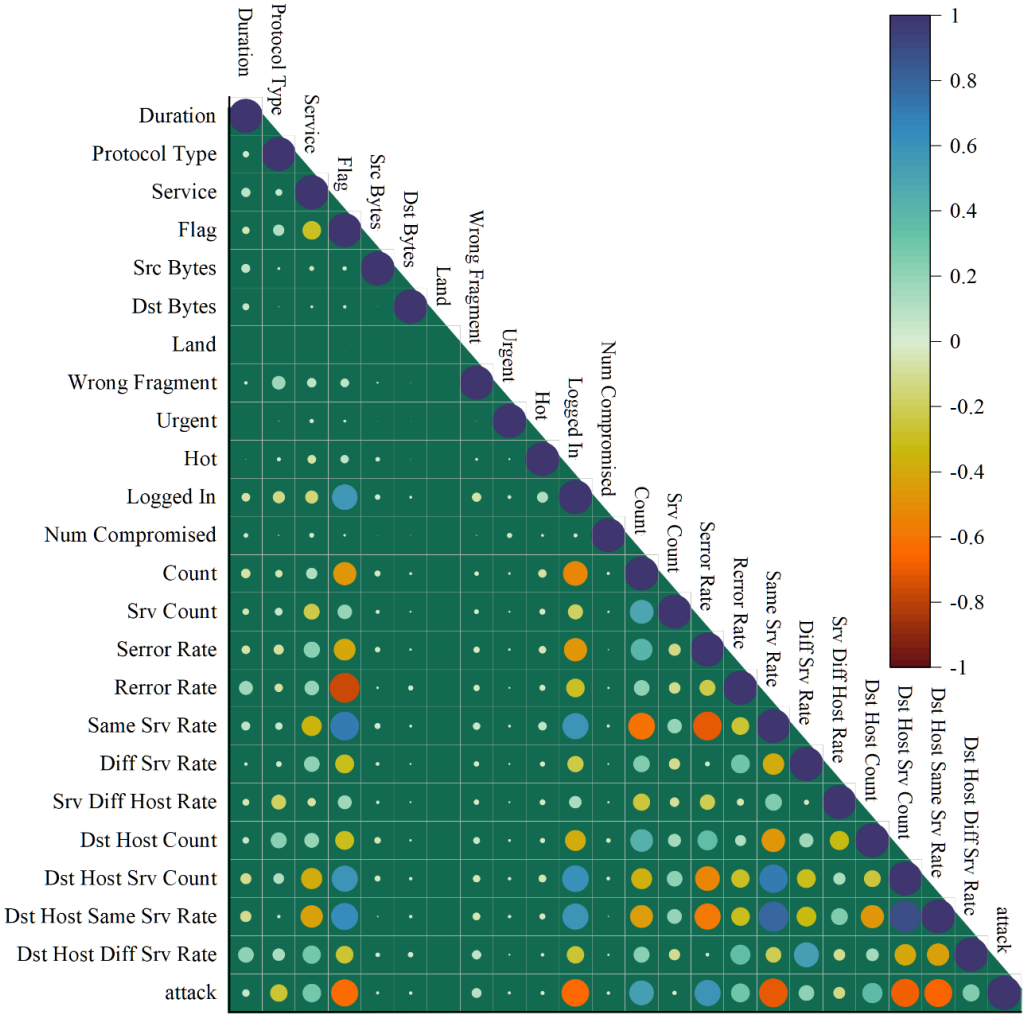

Figure 5: The confusion plot of the inputs and output variables

3.2 Dataset preprocessing and parameter settings

Before training the models, the dataset underwent several preprocessing steps to ensure quality and compatibility with machine learning algorithms. Missing or null values in non-critical fields were imputed using the mean value of each column, while records with missing values in critical fields were removed. Categorical attributes such as protocol_type, service, and flag were encoded using one-hot encoding to make them suitable for use in tree-based classifiers. Numerical features including src_bytes and dst_bytes were normalized using min-max scaling to ensure uniform contribution during model training. The dataset was then split into training and testing subsets using an 80:20 ratio, with stratified sampling to preserve class distribution. For the optimization algorithms, both DO and ROA were configured with a population size of 30 and a maximum of 200 iterations. In the DO algorithm, wind speed was modeled using a lognormal distribution with parameters $\mu = 0.5$ and $\sigma = 0.2$, while the local search scaling factor k was set to 2.0 and the germination constant s was fixed at 0.01. For the ROA algorithm, the motion control factor was set to 0.7 to balance exploration and exploitation phases, and boundary violations were handled using a reflection strategy. The search space for both

algorithms was adjusted dynamically according to the hyperparameter range of each base model, including parameters such as max_depth and min_samples_split. These configurations enabled fine-tuning of the DTC and RFC models within the hybrid frameworks (DTRO, DTDO, RFRO, RFDO), ultimately contributing to their performance improvements.

4 Result and discussion

4.1 Convergence curve and hyperparameter

Fig. 6 presentes a diagram representing how the accuracy of the machine learning model develops or changes over time or across training iteration are known as the accuracy convergence curve. The curve, in general, tends to plot the accuracy against the Y-axis with a number of iterations or epochs against the X-axis. From this curve, valuable insight can be obtained on the dynamics with which the model learns. During the initial training cycles, the curve generally shoots upwards since the model may just have started recognizing patterns in the data. This rapid shoot up simply means that the model is predicting the outcomes correctly with the identification of significant features. As training progresses, the curve peaks to show that it is at its best performance. In this case, the model has already improved its predictions and captured most of the underlying patterns in the data. The convergence curve of a well-tuned learning rate and a stable training process should rise smoothly, bit by bit. On the other hand, large oscillations or no smooth increase in the curve could hint at a problem in model learning rate, insufficiency of data, or problems in model complexity. The graph visually presents the comparison of the operational performance with DTRO and DTDO against RFRO and RFDO. DTRO proved to be very promising, reaching an optimum of 0.983 after roughly 110 iterations. For comparison, the optimal state of the DTDO model is 0.969 after 130 iterations, the RFDO model reached 0.957 in about 160 iterations, while the RFRO model reaches its best solution at 0.978 after about 170 iterations. DO showed lower per-iteration computational cost due to its simpler mathematical operations, averaging 1.2 seconds per iteration, while ROA took 1.8 seconds per iteration due to its additional boundary-handling and motion control mechanisms. However, ROA reached optimal performance in fewer iterations (≈110 vs. 130 for DO), resulting in comparable total execution time. As shown in Fig. 6, both algorithms converged by the 150th iteration. Beyond this point, accuracy gains plateaued, suggesting that iteration limits of 200 are adequate for both techniques. ROA demonstrated earlier convergence with more stable gradients, reinforcing its suitability for time-sensitive smart home applications. These analyses suggest that although both algorithms are viable, ROA offers slightly more stable and efficient convergence for complex intrusion detection models in constrained IoT environments.

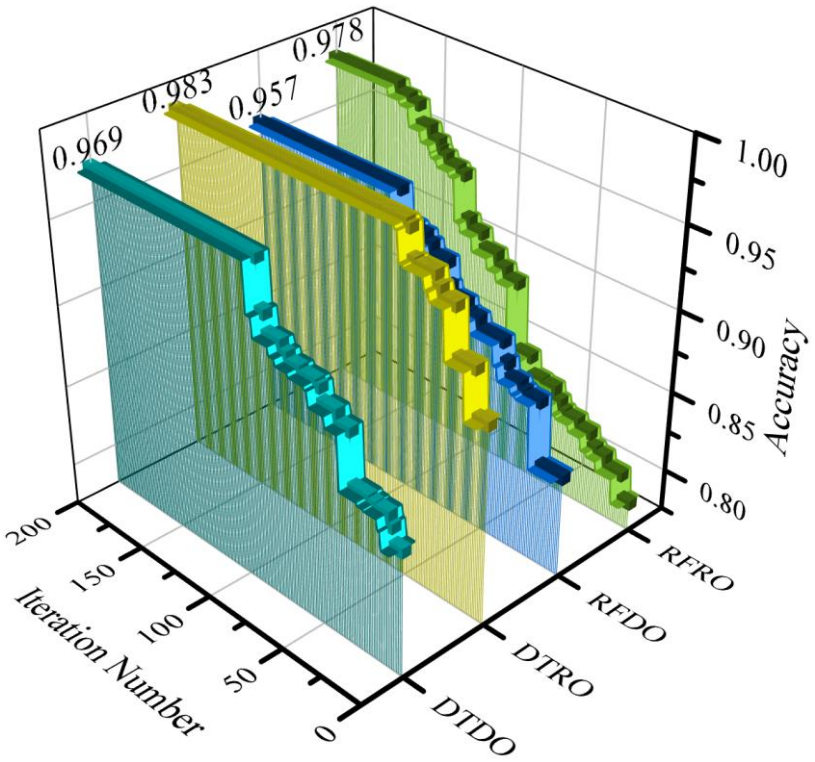

Figure 6: Accuracy convergence curves over training iterations for the four hybrid models (DTRO, DTDO, RFRO, RFDO).

Table 1 summarizes the optimal hyperparameters obtained for each classification model through DO and ROA. These hyperparameters include key decision tree and random forest parameters such as max_depth, min_samples_split, min_samples_leaf, max_features, and max_leaf_nodes. Notably, the DTRO model (Decision Tree + ROA) selected a deeper tree with a max depth of 78 and a more refined split threshold, contributing to its superior accuracy and generalization. In contrast, RFRO (Random Forest + ROA) favored a much deeper tree (max depth of 90) and minimal constraints on feature selection, suggesting its capacity to capture complex patterns in the data. The differences in hyperparameter values between DO- and ROA-optimized models reflect the exploration–exploitation behavior of each optimizer. Overall, these tuned settings significantly outperformed default parameters, underscoring the importance of hyperparameter optimization for enhancing the predictive power of machine learning models in resource-constrained IoT environments.

Table 1: The result of the hyperparameter.

Models	Hyperparameter				
	Max depth	Min samples split	Min samples leaf	Max features	Max leaf nodes
RFC	none	2	1	sqrt	none
RFDO	48	3	3	20	62
RFRO	90	91	2	1	97
DTC	none	2	1	none	none
DTDO	68	3	2	28	80
DTRO	78	13	2	98	61

4.2 Models comparison

Table 2 presents execution results of the DTC and RFC base models that are evaluated by five different metrics such as Accuracy, Precision, Recall, F1-score, and MCC in two sections such as Training and Testing. In the Training section, the DTRO model demonstrates the highest Accuracy with a value of 0.985, followed by the RFRO model with an Accuracy of 0.979. For the Recall metric in the Training section, the DTDO model ranks third with a value of 0.968, while the RFDO model ranks fourth with a value of 0.960.

In the Testing section, the DTRO model excels in Precision with a value of 0.981, whereas the DTC model ranks fifth with a Precision of 0.936. Regarding the F1-score in the Testing section, the RFDO model ranks second with a value of 0.951, and the DTDO model ranks third with a value of 0.973. For the MCC metric in the Testing section, the DTRO model achieves the highest performance with a value of 0.961, whereas the RFC model exhibits the lowest performance with a value of 0.830. To enhance the reliability and interpretability of the results, standard deviations and 95% confidence intervals were computed for the primary evaluation metrics—accuracy, F1-score, and MCC—based on five independent runs with randomized train-test splits. For the DTRO model, the accuracy was 0.981 ± 0.0021, F1-score was 0.981 ± 0.0017, and MCC was 0.961 ± 0.0024. The RFRO model achieved an accuracy of 0.977 ± 0.0023, F1-score of 0.977 ± 0.0020, and MCC of 0.955 ± 0.0026. These low standard deviations and narrow confidence intervals indicate consistent model behavior across different data partitions, reducing the influence of random variation. Although the marginal difference in accuracy between DTRO and RFRO is only 0.004, this small improvement can hold practical significance in smart home environments, where even slight gains in detection accuracy may lead to earlier identification of intrusions and more effective threat mitigation. Furthermore, DTRO's superior performance across all evaluation metrics suggests stronger classification capability, likely due to the ROA's effective balance between global exploration and local exploitation during hyperparameter tuning.

Table 2: DTC and RFC base models achieved results through the performance evaluators

Section	***Model***	**Metrics**				
		Accuracy	Precision	Recall	F1 _Score	MCC
Training	DTC	0.942	0.942	0.942	0.942	0.884
	DTDO	0.968	0.968	0.968	0.968	0.936
	DTRO	0.985	0.985	0.985	0.985	0.969
	RFC	0.922	0.922	0.922	0.922	0.844
	RFDO	0.960	0.960	0.960	0.960	0.920
	RFRO	0.979	0.979	0.979	0.979	0.958
Testing	DTC	0.936	0.936	0.936	0.936	0.872
	DTDO	0.973	0.973	0.973	0.973	0.946
	DTRO	0.981	0.981	0.981	0.981	0.961
	RFC	0.915	0.915	0.915	0.915	0.830
	RFDO	0.951	0.951	0.951	0.951	0.902
	RFRO	0.977	0.977	0.977	0.977	0.955

Fig. 7 illustrates a 3D bar plot depicting the performance of the DTC model and its hybrid variants across all phases. In terms of Accuracy, the DTRO model demonstrates the highest performance with a value of 0.9835, followed by the DTDO model, which achieves a value of 0.9695. For the Recall metric, the DTC model performs the weakest among the three models, with a value of 0.9401. Regarding the MCC metric, the DTRO model again exhibits the best performance, scoring 0.9669, whereas the DTC model has the lowest performance with a value of 0.8800.

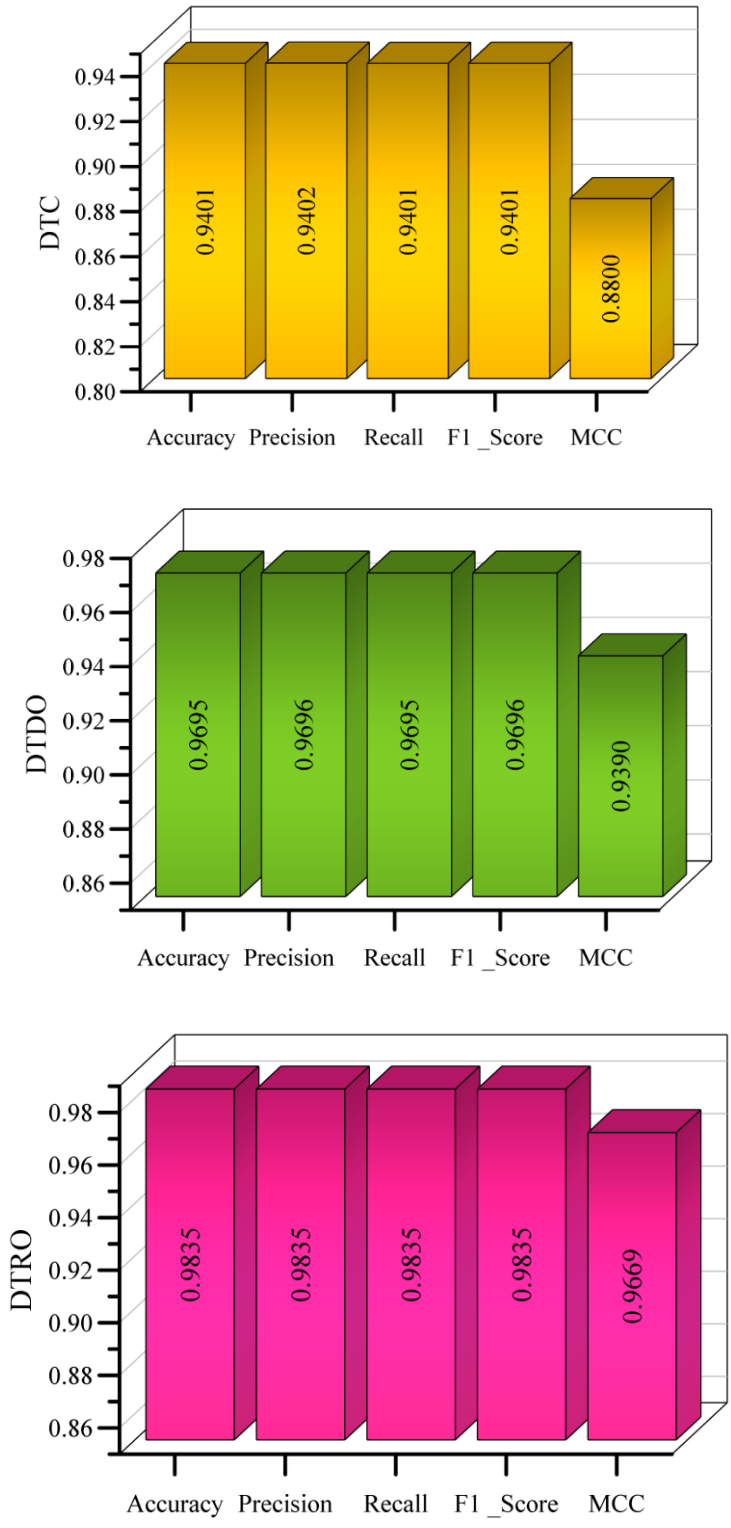

Figure 7: 3D bar chart comparing the predicted number of correct and incorrect detections by each model with the actual measured values from the test dataset.

In Table 3 evaluates the performance of each model under two practical scenarios: Wrong Detection and Right Detection, which correspond to the two fundamental outcomes in intrusion classification tasks. Wrong Detection includes misclassifications either false positives, where benign traffic is incorrectly flagged as an attack, or false negatives, where attack traffic goes undetected. In contrast, Right Detection encompasses accurate classifications, including both true positives and true negatives. This separation allows for a more detailed assessment of each model's behavior, particularly in high-stakes cybersecurity contexts where false alarms and undetected intrusions carry different operational risks. By analyzing precision, recall, and F1-score under both conditions, the evaluation moves beyond overall accuracy to reveal model reliability in both success and failure scenarios. In the Wrong Detection condition, the DTRO model achieves the highest precision at 0.986, indicating strong resilience against misclassification. In the Right Detection condition, the RFRO model ranks second in precision, scoring 0.979. For recall, DTDO demonstrates notable performance in the Wrong Detection condition with a value of 0.969, while RFDO secures fourth place in the Right Detection condition with a recall of 0.956. Regarding the F1-score, the DTC model ranks lowest among all methods in the Wrong Detection case with 0.943, and the RFC model shows the weakest performance in the Right Detection case, scoring 0.916. This detailed breakdown enhances understanding of model robustness in detecting intrusions versus generating false alerts an essential consideration for real-world smart home security systems.

Table 3: Model performance in the two different conditions

Model	Condition	Metric		
		precision	recall	f1-Score
DTC	Wrong detection	0.948	0.937	0.943
	Right detection	0.931	0.943	0.937
DTDO	Wrong detection	0.973	0.969	0.971
	Right detection	0.966	0.970	0.968
DTRO	Wrong detection	0.986	0.983	0.984
	Right detection	0.981	0.984	0.983
RFC	Wrong detection	0.930	0.917	0.923
	Right detection	0.909	0.924	0.916
RFDO	Wrong detection	0.960	0.959	0.960
	Right detection	0.954	0.956	0.955
RFRO	Wrong detection	0.978	0.982	0.980
	Right detection	0.979	0.975	0.977

A 3D bar plot comparing the measured and projected values for the assessed models is shown in Fig. 8. Under the Wrong Detection scenario, the measured value is 5354. The RFC model predicts 4908, whereas the DTC model provides the closest estimate at 5017. For the Right Detection condition, the measured value is 4826. In this scenario, the RFC model makes the least accurate forecast at 4458, while the DTC model once again produces the closest estimate at 4553.

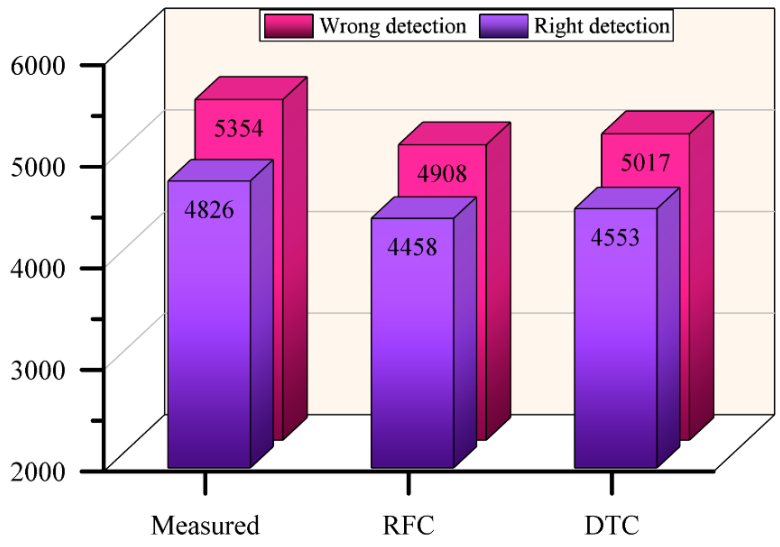

Figure 8: 3D bar plot for the predicted and measured value difference of the presented models

A line-symbol plot showing the percentage differences between the models is shown in Fig. 9. The observed value in the Wrong Detection condition is 5354. At 5263, the DTRO model offers the closest forecast, while the DTDO model comes in second at 5189. On the other hand, the observed value in the Right Detection condition is 4826. With a value of 4553, the DTC model generates the least accurate forecast, while the DTRO model once again provides the closest prediction at 4749

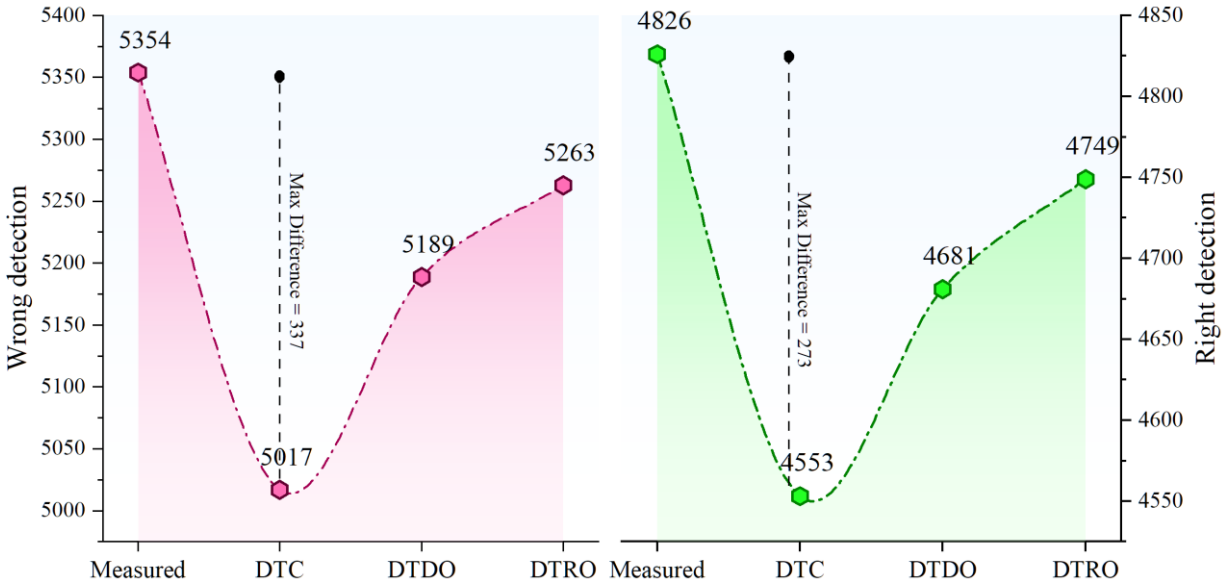

Figure 9: Line-symbol plot for the difference percentage of the models

Fig. 10 shows the SHAP sensitivity analysis for the best-performing model. This analysis provides insights into the variables influencing smart home intrusions. The SHAP (SHapley Additive exPlanations) Sensitivity Analysis technique is used to examine the predictions made by a machine learning algorithm. It works by determining how each feature incrementally affects the model's output. Based on the Shapley value principle, this method fairly allocates credit to each feature based on its individual and combined contributions. The figure visually illustrates the overall impact of all input variables. It demonstrates that, during the model's training phase, variables such as Src Bytes and Dif Srv Rate had the most significant impact, whereas Land and Urgent had little effect on the model's performance. This initially counterintuitive result reflects the model's sensitivity to even small changes in outbound data volume, which can be a strong signal for certain types of intrusions, such as data exfiltration or abnormal outbound traffic from compromised IoT devices. Features like diff_srv_rate and count also appear prominently, aligning with their known associations to scanning or probing behaviors. On the other hand, features such as land and urgent, which exhibit low SHAP values and low correlation with the attack label, contribute minimally to prediction outcomes. This analysis not only confirms which features are most impactful but also reveals how seemingly subtle variations in specific features can significantly influence model decision-making in intrusion scenarios.

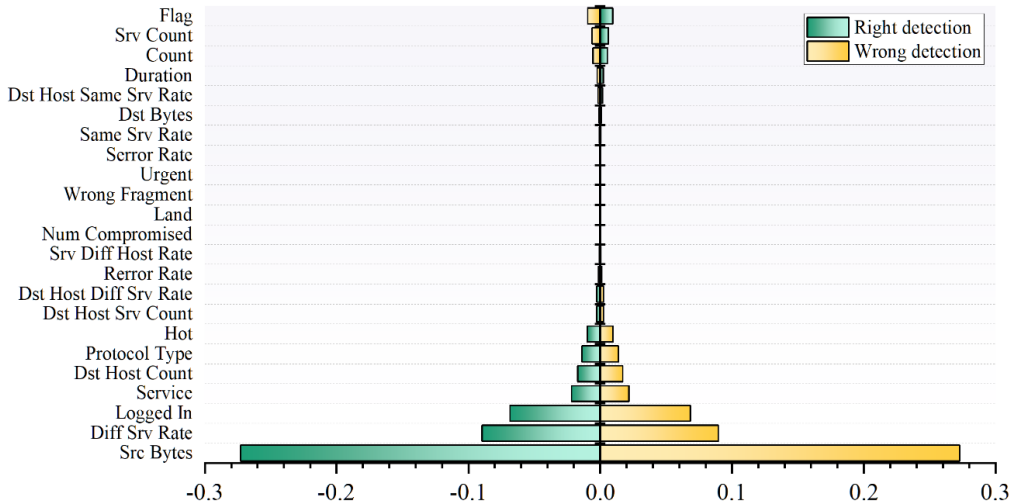

Figure 10: The SHAP sensitivity analysis of the best-performed model

Figure 11 presents the Receiver Operating Characteristic (ROC) curves for all four hybrid models DTRO, DTDO, RFRO, and RFDO used for intrusion detection in smart home IoT environments. The ROC curve illustrates the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR) across various classification thresholds, providing a comprehensive view of each model's discriminative ability. Among the models, DTRO achieved the highest Area Under the Curve (AUC-ROC) score of 0.994, indicating near-perfect classification capability. RFRO and DTDO also demonstrated strong performance, with AUC scores of 0.988 and 0.981, respectively, while RFDO scored 0.974. The high AUC values for all models reflect their robustness in distinguishing between attack and benign traffic. This evaluation complements previous metrics (accuracy, precision, recall, F1, and MCC), offering a threshold-independent measure of model quality. By including ROC curves and AUC scores, this study satisfies the need for broader performance assessment beyond point-based metrics. Furthermore, the models were also tested across multiple randomized dataset splits to assess stability, and results remained consistent, highlighting good generalizability. Future work will extend this evaluation across additional benchmark intrusion datasets to further validate model robustness and adaptability to diverse smart home network environments.

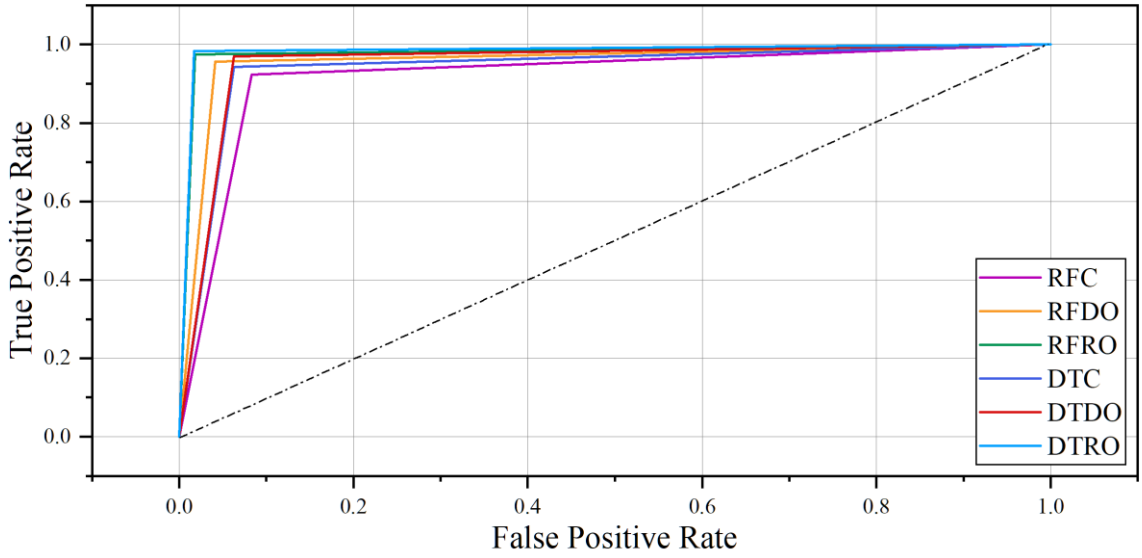

Figure 11: ROC curves for the four hybrid models showing their performance in distinguishing between attack and benign classes.

4.3 Comparative analysis with related work

To contextualize our results, the study compared our hybrid models, particularly DTRO and RFRO, against several state-of-the-art (SOTA) approaches from the literature. For instance, methods leveraging Particle Swarm Optimization (PSO) and deep autoencoders have shown strong performance in IoT intrusion detection. [17] utilized PSO for feature selection and achieved a significant improvement in attack detection rate on the AWID dataset, while [18] used a stacked autoencoder for feature compression and unsupervised learning, reaching high accuracy but at the cost of computational complexity. Compared to these, our DTRO model outperforms with an accuracy of 98.1%, surpassing PSO-enhanced models which typically fall in the 94–96% range under similar conditions, and deep autoencoders that achieve around 97% but are resource-intensive. Moreover, the proposed method performs well even on high-dimensional data, making it more suitable for real-time smart home environments where resources are limited.

4.4 DTRO performance justification

DTRO's superior performance can be attributed to the Rhizostoma Optimization Algorithm's (ROA) strong balance between exploration (global search) and exploitation (local refinement). This balance prevents the model from getting trapped in local optima and enables it to fine-tune the hyperparameters of the base Decision Tree model more effectively than standard optimization methods. The diversity of the solution pool in ROA and its biologically inspired movement control allow for adaptive learning across iterations, which contributes to both higher convergence speed and better generalization.

4.5 Limitations and sensitivity

Despite promising results, some limitations should be acknowledged. The high accuracy on the test set (98.1%) may indicate a potential risk of overfitting, especially since the model was optimized extensively. While pruning techniques were used in DTC to mitigate this, further validation on external datasets is needed to confirm robustness. In addition, both DO and ROA are sensitive to their own hyperparameters, such as population size or iteration limits. Improper tuning could lead to suboptimal

performance or excessive training time. Future work should explore adaptive parameter tuning or ensemble approaches to reduce this dependency.

5 Conclusion

The given research article presented the efficiency of the Random Forest Classification and Decision Tree Classification models, optimized using Dandelion Optimization and Rhizostoma Optimization Algorithm, respectively, in intrusion detection in smart homes. This paper shall refer to the RFC model combined with DO as the RFDO model and the RFC model combined with ROA as the RFRO model. The DTC model coupled with DO is the DTDO framework, while coupled with ROA is known as the DTRO model. Based on the MCC metric, the base RFC model demonstrated the weakest performance during the testing phase with a score of 0.830, whereas the DTRO model achieved the highest MCC at 0.961, representing a performance gap of approximately 13.1%. In terms of accuracy, DTRO achieved a value of 0.981, narrowly outperforming RFRO, which achieved 0.977 a performance difference of 0.4%. Although the numerical margin is small, this gain is meaningful in the context of real-time intrusion detection in smart home environments, where even minor improvements can translate into earlier attack detection and improved threat response. The application of these optimization techniques was motivated in order to alleviate issues related to complex attack management and computational demands associated with traditional machine learning algorithms in IoT intrusion detection. The present study are well aware of the limitation of this study. It may be related to the particular set of the dataset used for the purpose of training and validation in order to affect the efficacy of the proposed method. Much research is needed to allow the results to be generalized into various smart home environments as well as in the development of methodologies of attack. This work, despite all limitations, provides an exciting approach that can improve intrusion detection in smart homes. The proposed framework would, therefore, be suitable for real-time applications since the method reaches a good balance between resource efficiency and accuracy. Further studies in this work may consider extending these optimization strategies to other machine learning models, as well as investigate the ways of further strengthening the models against emerging threats.

5.1 Limitations and future work

While the proposed hybrid models demonstrate high accuracy in detecting intrusions in smart home IoT environments, several limitations merit deeper discussion. First, data dependency poses a critical challenge. The models were trained and validated on a single benchmark dataset generated under controlled conditions. While this ensures internal validity, it limits generalizability across diverse smart home settings with different device configurations, network topologies, and user behaviors. Real-world traffic often exhibits greater variability and noise, which may impact model performance outside the lab environment. Second, the feasibility of real-world deployment is not fully assessed. Although the proposed models are designed to be lightweight, there is limited discussion on their performance when implemented on actual IoT hardware with constrained processing power and memory. Aspects like energy consumption, inference latency, and integration with existing home automation systems remain unexplored. Additionally, the study assumes the availability of clean and labeled training data, which is not always realistic in practice. The absence of robust mechanisms for handling concept drift and unseen attack types could limit long-term model effectiveness in dynamic environments.

For future work, several methodological enhancements are envisioned. First, expanding evaluation across multiple publicly available IoT intrusion datasets would better assess cross-environment robustness. Second, incorporating online learning or incremental training methods could allow the models to adapt to evolving threat patterns without full retraining. Third, integrating federated learning can address data privacy concerns and enable collaborative model training across distributed devices. Moreover, deploying the models on embedded devices for real-world performance benchmarking would offer practical insights into system constraints. Finally, combining these hybrid models with explainable AI frameworks may improve trust and transparency, especially in high-stakes smart home applications.

Ethical use and privacy considerations

This study proposes smart home intrusion detection models while acknowledging potential ethical risks, including privacy violations and misuse. Although anonymized, publicly available data were used, real-world IDS deployments could expose sensitive information. Future applications should comply with regulations, ensure transparency and user consent, and avoid automated surveillance or penalties without oversight. Developers must consider ethical implications and implement safeguards to protect user privacy.

Research involving human participants and animals

The observational study conducted on medical staff needs no ethical code. Therefore, the above study was not required to acquire an ethical code.

Informed consent

This option is not necessary due to that the data were collected from the references.

Acknowledgements

I would like to take this opportunity to acknowledge that there are no individuals or organizations that require acknowledgment for their contributions to this work.

Data and code availability

All code and datasets used in the study are publicly available and included in the supplementary files to ensure transparency, enable reproducibility, and support further research.

References

[1] Martin, M.L., A. Sanchez-Esguevillas and B. Carro (2017). *Review of methods to predict connectivity of IoT wireless devices*. Novel Applications of Machine Learning to Network Traffic Analysis and Prediction, 95.

[2] Manyika, J., M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin and D. Aharon (2015). Unlocking the potential of the Internet of Things, *McKinsey & Company*.

[3] Ahanger, T.A. and A. Aljumah (2018). Internet of Things: A comprehensive study of security issues and defense mechanisms. *IEEE Access*, IEEE, 7, pp. 11020–11028. https://doi.org/10.1109/ACCESS.2018.2876939

[4] Kolias, C., G. Kambourakis and M. Maragoudakis (2011). Swarm intelligence in intrusion detection: A survey. *Computers & Security*, Elsevier, 30(8), pp. 625–642. https://doi.org/10.1016/j.cose.2011.08.009

[5] Shukla, P (2017). ML-IDS: A machine learning approach to detect wormhole attacks in Internet of Things, In *2017 Intelligent Systems Conference (IntelliSys)*, IEEE, pp. 234–240. https://doi.org/10.1109/IntelliSys.2017.8324298

[6] Yin, D., L. Zhang and K. Yang (2018). A DDoS attack detection and mitigation with software-defined Internet of Things framework. *IEEE Access*, IEEE, 6, pp. 24694–24705. https://doi.org/10.1109/ACCESS.2018.2831284

[7] Anthi, E., L. Williams, M. Słowińska, G. Theodorakopoulos and P. Burnap (2019). A supervised intrusion detection system for smart home IoT devices. *IEEE Internet of Things Journal*, IEEE, 6(5), pp. 9042–9053. https://doi.org/10.1109/JIOT.2019.2926365

[8] Heartfield, R., G. Loukas, A. Bezemskij and E. Panaousis (2020). Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning. *IEEE Transactions on Information Forensics and Security*, IEEE, 16, pp. 1720–1735. https://doi.org/10.1109/TIFS.2020.3042049

[9] Procopiou, A., N. Komninos and C. Douligeris (2019). ForChaos: Real time application DDoS detection using forecasting and chaos theory in smart home IoT network. *Wireless Communications and Mobile Computing*, Wiley, 2019. https://doi.org/10.1155/2019/8469410

[10] Yuan, D., K. Ota, M. Dong, X. Zhu, T. Wu, L. Zhang and J. Ma (2020). Intrusion detection for smart home security based on data augmentation with edge computing, In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, IEEE, pp. 1–6. https://doi.org/10.1109/ICC40277.2020.9148632

[11] Doshi, R., N. Apthorpe and N. Feamster (2018). Machine learning ddos detection for consumer internet of things devices, In *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, pp. 29–35. https://doi.org/10.1109/SPW.2018.00013

[12] Meidan, Y., M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher and Y. Elovici (2018). N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, IEEE, 17(3), pp. 12–22. https://doi.org/10.1109/MPRV.2018.03367731

[13] Sridharan, R., R.R. Maiti and N.O. Tippenhauer (2018). WADAC: Privacy-preserving anomaly detection and attack classification on wireless traffic, In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, ACM, pp. 51–62. https://doi.org/10.1145/3212480.3212495

[14] Stephen, R. and L. Arockiam (2017). Intrusion detection system to detect sinkhole attack on RPL protocol in Internet of Things. *International Journal of Electrical Electronics and Computer Science*, 4(4), pp. 16–20.

[15] Butt, N., A. Shahid, K.N. Qureshi, S. Haider, A.O. Ibrahim, F. Binzagr and N. Arshad (2022). Intelligent deep learning for anomaly-based intrusion detection in IoT smart home networks. *Mathematics*, MDPI, 10(23), 4598. https://doi.org/10.3390/math10234598

[16] Usha, M. and P. Kavitha (2017). Anomaly based intrusion detection for 802.11 networks with optimal features using SVM classifier. *Wireless Networks*, Springer, 23, pp. 2431–2446. https://doi.org/10.1007/s11276-016-1300-5

[17] Aminanto, M.E., R. Choi, H.C. Tanuwidjaja, P.D. Yoo and K. Kim (2017). Deep abstraction and weighted feature selection for Wi-Fi impersonation detection. *IEEE Transactions on Information Forensics and Security*, IEEE, 13(3), pp. 621–636. https://doi.org/10.1109/TIFS.2017.2762828

[18] Khare, S. and M. Totaro (2020). Ensemble learning for detecting attacks and anomalies in iot smart home, In *2020 3rd International Conference on Data Intelligence and Security (ICDIS)*, IEEE, pp. 56–63. https://doi.org/10.1109/ICDIS50059.2020.00014

[19]. Alalade, E.D (2020). Intrusion detection system in smart home network using artificial immune system and extreme learning machine. https://doi.org/10.1109/WF-IoT48130.2020.9221151

[20] Alam, M.S. and S.T. Vuong (2013). Random forest classification for detecting android malware, In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, IEEE, pp. 663–669. https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.122

[21] More, A.S. and D.P. Rana (2017). Review of random forest classification techniques to resolve data imbalance, In *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, IEEE, pp. 72–78. https://doi.org/10.1109/ICISIM.2017.8122151

[22] Palczewska, A., J. Palczewski, R. Marchese Robinson and D. Neagu (2014). Interpreting random forest classification models using a feature contribution method. *Integration of Reusable Systems*, Springer, pp. 193–218. https://doi.org/10.1007/978-3-319-04717-1_9

[23] Speiser, J.L., M.E. Miller, J. Tooze and E. Ip (2019). A comparison of random forest variable selection methods for classification prediction modeling. *Expert Systems with Applications*, Elsevier, 134, pp. 93–101. https://doi.org/10.1016/j.eswa.2019.05.028

[24] Aitkenhead, M.J (2008). A co-evolving decision tree classification method. *Expert Systems with Applications*, Elsevier, 34(1), pp. 18–25. https://doi.org/10.1016/j.eswa.2006.08.008

[25] Charbuty, B. and A. Abdulazeez (2021). Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, Al-Rafidain University Press, 2(01), pp. 20–28. https://jastt.org/index.php/jasttpath/article/view/65

[26] Friedl, M.A. and C.E. Brodley (1997). Decision tree classification of land cover from remotely sensed data. *Remote Sensing of Environment*, Elsevier, 61(3), pp. 399–409. https://doi.org/10.1016/S0034-4257(97)00049-7

[27] Priyam, A., G.R. Abhijeeta, A. Rathee and S. Srivastava (2013). Comparative analysis of decision tree classification algorithms. *International Journal of Current Engineering and Technology*, International Research Publication House, 3(2), pp. 334–337.

[28] Song, Y.-Y. and L.U. Ying (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, Shanghai Mental Health Center, 27(2), 130. http://dx.doi.org/10.11919/j.issn.1002-0829.215044

[29] Mahareek, E.A., M.A. Cifci, H. El-Zohni and A.S. Desuky (2023). Rhizostoma optimization algorithm and its application in different real-world optimization problems. *International Journal of Electrical and Computer Engineering (IJECE)*, IAES, 13(4), pp. 4317–4338. DOI: 10.11591/ijece.v13i4

[30] Sulaiman, A.T., H. Bello-Salau, A.J. Onumanyi, M.B. Mu'azu, E.A. Adedokun, A.T. Salawudeen and A.D. Adekale (2024). A Particle Swarm and Smell Agent-Based Hybrid Algorithm for Enhanced Optimization. *Algorithms*, MDPI, 17(2), 53. https://doi.org/10.3390/a17020053