Received: 28 September 2025 | Revised: 4 December 2025 | Accepted: 15 December 2025 | Published online: 15 January 2026

RESEARCH ARTICLE

Artificial Intelligence and Applications
2026, Vol. 00(00) 1–10
DOI: 10.47852/bonviewAIA62027781

Heuristic-Guided Selective Random Forests: Immune-Inspired Heuristic Ensembles for Robust Intrusion Detection in IoT Networks

Saad AL Azzam[1,*], Ghassan ALDharhani[1], and Raenu AL Kolandaisamy[1]

[1] *Institute of Computer Science and Digital Innovation, UCSI University, Malaysia*

Abstract: The increasing adoption of the Internet of Things (IoT) has brought considerable benefits to all aspects of daily life, but the IoT remains vulnerable to a wide range of security threats that compromise device continuity and can lead to significant disruption and losses. Traditional intrusion detection systems (IDSs) often rely on machine learning models, which perform relatively well but struggle when faced with unusual traffic patterns. To address these challenges, this study presents a heuristic-based ensemble method. The proposed method combines decision tree detectors with a biologically inspired filtering mechanism and mimics the behavior of the immune system by retaining only trees that achieve a detection accuracy greater than a specified threshold. This method reduces the memory required to store weak decision trees that can affect the final decision, thus improving classification performance. Experimental results demonstrate an overall accuracy of 93.4%, outperforming baseline algorithms. The research presents a framework that balances detection performance with computational efficiency, a critical requirement for IoT networks with limited resources.

Keywords: Internet of Things, intrusion detection, ensemble learning, random forest, decision tree, negative selection algorithm

1. Introduction

The continuous development of the Internet of Things (IoT) has transformed the way societies operate, integrating billions of devices across diverse sectors, such as smart-healthcare systems and autonomous transportation networks [1]. This interconnected ecosystem is highly vulnerable to a wide range of cyber threats that can be launched against the devices or the network as a whole. This makes intrusion detection systems (IDSs) a critical component of modern IoT security strategies [2].

Machine learning (ML) techniques have been used in developing IDSs due to their ability to detect subtle patterns in network traffic. However, they have significant drawbacks. Many IDSs struggle to handle imbalanced datasets, where the majority of network traffic is benign, while a small portion represents malicious activity [3]. This biases models toward normal categories and misses rare attacks.

Random forests (RFs) have emerged as one of the most popular ensemble ML methods because they achieve good performance by combining predictions from a large number of decision trees (DTs). However, RFs typically retain all generated trees, regardless of their individual quality, which can lead to high computational costs and memory consumption [4–6]. Furthermore, not all trees may contribute effectively to the final decision, and poorly performing trees may even reduce the overall accuracy. This inefficiency is especially problematic in IoT settings, where memory and processing capabilities are limited.

Minority attack classes in IoT datasets typically exhibit sparse and irregular patterns [7], making them difficult for most DTs to learn. In RF, these weak and majority-biased trees participate in the final vote. These trees affect the trees that successfully capture the minority class.

These challenges have motivated the search for alternative ensemble methods that selectively retain only the most effective models and discard models that may consume resources without providing any real benefit in improving prediction accuracy.

In this study, we propose a pairwise DT ensemble model that incorporates a heuristic, immune-inspired filtering mechanism, where the system, instead of storing each generated DT, filters them according to the accuracy that they achieve on self-samples, retaining only those that exceed a predefined threshold. This selectivity reduces the computational burden and ensures that only the strongest detectors contribute to the classification process. In other words, instead of the generated DTs contributing to the classification of different attack types, the trees generated using the proposed method will be specialized for specific attack types, with each set of trees focusing on distinguishing its own attack type from the rest. As a result, the set becomes more sensitive to minority attacks, improving recall without increasing computational complexity.

The motivations behind this work are the following: (1) to address the urgent need for more accurate detection of underrepresented attack classes, which are often overlooked by traditional algorithms, and (2) to build a framework that balances detection performance and computational efficiency, a very important requirement for resource-constrained IoT networks.

[*]**Corresponding author:** Saad AL Azzam, Institute of Computer Science and Digital Innovation, UCSI University, Malaysia. Email: 1002372379@ucsiuniversity.edu.my

The principal contributions of this work are as follows:

1) We introduce a novel ensemble method that integrates heuristic, immune-inspired filtering with DT ensembles to improve intrusion detection accuracy.
2) We design a resource-efficient framework that balances high detection performance with low computational cost, addressing a critical requirement for IoT networks.
3) We conduct extensive evaluations on the UNSW-NB15 dataset, demonstrating that the proposed model achieves superior performance compared to standard RF and other classic ML benchmarks.

The remainder of this paper is organized as follows: Section 2 reviews related work on intrusion detection for IoT. Section 3 details the proposed methodology and dataset. Section 4 presents and discusses the experimental results. Finally, Section 5 concludes the paper and outlines directions for future research.

2. Literature Review

The development of effective IDSs for IoT networks has been a focal point of cybersecurity research, leading to a wide array of ML and deep learning (DL) approaches. This review organizes related work into key thematic categories to contextualize the contributions of this paper.

2.1. Traditional ML approaches

Early research in IDSs has explored conventional ML algorithms. These studies have evaluated a range of classifiers on benchmark datasets such as NSL-KDD and UNSW-NB15 [8]. Support vector machines (SVMs), in particular, have been a popular choice. For example, one study focused specifically on denial-of-service (DoS) attack detection by analyzing packet arrival patterns [9]. To enhance SVM performance and efficiency, hybrid feature selection methods have been proposed, such as combining genetic algorithms (GA) and gray wolf optimization (GWO) to create a lightweight IDS [10]. These conventional models are often computationally inexpensive, but they frequently struggle with the complex, nonlinear relationships in network traffic and are highly susceptible to performance degradation on imbalanced datasets.

2.2. Ensemble and hybrid methods

To overcome the limitations of single classifiers, researchers have turned to ensemble methods, which aggregate multiple models to improve robustness and accuracy. RF is frequently employed due to its inherent ensemble nature such as one that combined K-nearest neighbors (KNNs), RF, and XGBoost, using principal component analysis (PCA) for dimensionality reduction to minimize CPU consumption [11]. Beyond traditional ensembles, more complex hybrid frameworks have emerged. These include FIDChain, which integrates blockchain with a lightweight neural network for securing healthcare IoT [12], and systems that combine federated learning with K-means clustering to identify attacks while reducing feature dimensions [13]. These ensemble and hybrid methods generally achieve higher accuracy than single models, but they often require more resource demands; for example, standard RF stores all generated trees regardless of their individual performance, which can lead to bloated memory usage and the inclusion of weak learners that negatively affect the collective decision.

This research [14] used the synthetic minority over-sampling technique (SMOTE) to address data imbalance. The authors used a hybrid approach that combines long short-term memory (LSTM) with convolutional neural networks (CNNs). Furthermore, the use of the CNN + LSTM approach is computationally expensive and consumes a significant amount of resources, which is very important in IoT networks that contain terminal nodes with limited resources.

This research [15] used DL techniques for the detection of DDoS attacks in IoT. The authors evaluated many DL models, which were latent autoencoders, CNN, and LSTM. They proposed a hybrid approach that combines CNN with LSTM. The researchers acknowledged the limitations of the proposed model in detecting different types of attacks and the biases arising from the imbalance of the dataset. Furthermore, the model was computationally expensive and unsuitable for resource-constrained environments such as IoT.

2.3. Lightweight and resource-aware models

Given the resource constraints of IoT devices, a significant research direction involves developing a lightweight IDS. A common strategy is to apply dimensionality reduction techniques such as PCA, as shown in study [16], which used the SAM-KNN algorithm. Other approaches focus on model deployment in constrained environments, such as implementing a J48 DT on a Raspberry Pi to balance accuracy with processing speed [17]. While these methods successfully reduce computational load, this often comes at the cost of detection capability, particularly for complex or rare attacks.

The research [18] proposed a method to select high representative subset samples from a dataset based on entropy. The goal is to reduce the model size and obtain a lightweight model for IoT intrusion detection. The model reduces the memory usage by 18% compared with other approaches. Selecting fewer samples from the dataset may result in the loss of important information or important data patterns that the artificial intelligence model could benefit from.

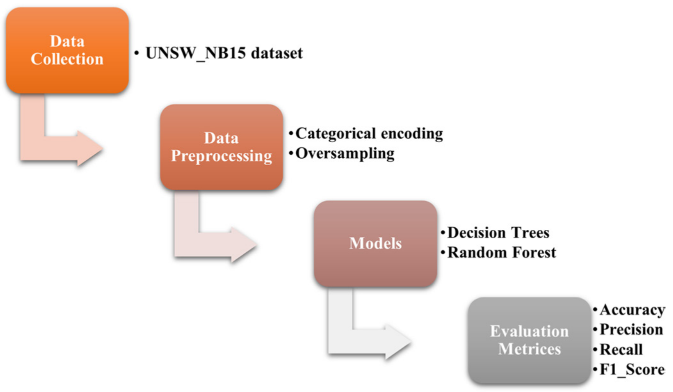
2.4. Research gap and contribution

Conventional and lightweight models often lack high-accuracy detection across all attack types, especially minority classes, while more powerful ensemble methods can be computationally expensive for IoT environments. There is a clear need for an IDS that does not only balance but also optimize both detection performance and resource efficiency. Our proposed model directly addresses this gap by enhancing powerful RF ensembles by introducing a biologically inspired, heuristic filtering step that selectively stores only high-performing DTs. This novel approach reduces the model's memory size and improves the overall accuracy by deleting weak detectors.

3. Research Methodology

In this section, we describe the classification methodology implemented using a Python code. The proposed method generates a set of DT detectors for each pair of classes, transforming the classification process from multi-class to multi-binary to improve performance and accurately capture the relationships between each pair of classes in the dataset. We then simulate the mechanism of the body's immune system by verifying the performance of the detectors on Class (1) samples, which represent self-cells (normal body cells). Detectors that achieve accuracy greater than a specified threshold are retained. Through this strategy, we simulate the RF construction process, but a new, pseudo-random method that relies on heuristics to select DTs that guarantee high classification performance is used. The final predictions are generated through a voting system, where the majority of the votes are taken for the selected DTs. Figure 1 shows the general steps of the proposed methodology.

Figure 1
The general steps of the proposed methodology



3.1. Dataset description and preprocessing

The UNSW-NB15 dataset is used, which has 42 features, 82,333 records from several types of attacks (Fuzzers, Analysis, Backdoors, DoS, Exploit, Generic, Reconnaissance, Shellcode, and Worms), and normal records [19]. The UNSW-NB15 dataset suffers from a severe imbalance in categories [20]. As shown in Table 1, some categories are represented by a large number of samples and others by a very small number of samples. Table 1 shows the sample count for each attack in the dataset.

Table 1
The sample count for each attack in the dataset

Attack	Count
Normal	37000
Generic	18871
Exploits	11132
Fuzzers	6062
DoS	4089
Reconnaissance	3496
Analysis	677
Backdoor	583
Shellcode	378
Worms	44

Dataset *D* can be described as shown in Equation (1):

$$D = (x_n, y_n) \; where \; n \; \in \; [1, N], \tag{1}$$

where

$x_n \; \in \; R^d$: input sample (feature matrix)

$y_n \; \in \; \Upsilon = \{1, 2, 3, \ldots, C\}$: the label

C: class set.

Here, *N* is the total number of records in dataset *D*, and x_n represents the *n*-th feature vector. The label y_n belongs to the set of possible classes *C*, where *C* = 10 attack types, corresponding to the nine attack types and the normal traffic class.

Data preprocessing involves the following:

1) Categorical encoding: categorical columns in the dataset are mapped to integer codes via the label encoding function.
2) A stratified train–test split is performed with a proportion (train: 80%, test: 20%), and the output contains two datasets (D_{train}, D_{test}).
3) To reduce class imbalance, SMOTE is applied on the training set to generate synthetic samples for minority classes, where this technique synthesizes new samples by interpolation between a minority sample and one of its KNNs (we adjusted $k = 3$ in this research) in feature space [21].
4) Assume that the class count in the dataset is donated n_c, so SMOTE outputs a balanced training set D_{res} with n'_c count for each class. The new sample is calculated as shown in Equation (2).

$$x_{new} = x + \lambda(x_{nn} - x) \; where \; \lambda \; \in [0, 1]. \tag{2}$$

3.2. The proposed model

RF constructs an ensemble by generating a large number of DTs through bootstrap aggregation and random feature selection, retaining all trees for the final majority vote. While powerful, this approach is agnostic to the individual quality of each tree, often resulting in a model bloated with weak or redundant learners that consume memory and can degrade the overall performance.

The heuristic-guided selective RF (HGS-RF) model proposed in this work introduces a fundamental shift from this inclusive paradigm. The core innovation lies in the integration of a heuristic filtering mechanism, inspired by the negative selection of the immune system, which acts as a quality gate after tree generation but before ensemble formation. This process rigorously evaluates and selectively retains only the most competent DT "detectors," thereby creating a more accurate and more efficient ensemble.

The model draws conceptual inspiration from the negative selection mechanism of the biological immune system. However, the algorithm itself operates purely on ML principles. The analogy is used only to motivate the idea of filtering detectors, not to define algorithmic steps. Table 2 illustrates the biological terms and their meanings in ML.

3.2.1. Immune inspiration and core concept

The human immune system employs a "negative selection" process where T-cells that react strongly to "self" proteins are eliminated, ensuring that only cells capable of recognizing "non-self" pathogens mature. Inspired by this mechanism, the proposed model treats benign network traffic as "self" and malicious activities as "non-self." Instead of retaining all DTs such as a standard RF, the model introduces a heuristic filter that retains only high-performing tree "detectors" based on their accuracy on "self" (benign) samples. This selective process enhances the ensemble's overall robustness and efficiency.

3.2.2. Generation of candidate detectors

The model begins by generating a diverse pool of candidate DT detectors. To capture nuanced decision boundaries, the multi-class problem is decomposed into multiple binary classification tasks.

For each pair of classes (c_i, c_j), a subset $D_{i,j}$ of the original dataset is generated, containing the samples for each class.

$$D_{i,j} = \{(x, y) \in D | y \in \{c_i, c_j\}\}. \tag{3}$$

To train specialized detectors and capture fine-grained decision boundaries, HGS-RF decomposes the multi-class problem

Table 2
The biological terms and their meanings in ML

Biological term	ML term
Self-cells	Benign network traffic
Non-self	Malicious network traffic
T-cells, cells, detectors	Decision trees
Candidate detectors	Decision trees that are candidates to be specialized for the specific type of attack
Valid detector	Candidate decision trees that are suitable for specializing in a specific type of attack
Negative selection	Selecting decision trees that accurately identify malicious network behavior (negative behavior) for a specific attack type

into binary tasks. For every unique pair of classes (c_i, c_j), where $i > j$, we construct a dedicated binary dataset $D_{i,j}$. This dataset contains only the samples belonging to either class c_i or class c_j, effectively isolating the discrimination task between those two specific traffic types. This subset contains only examples from these two classes.

DTs are trained on a subset of the dataset features. These features are indexed by set $F \subseteq \{1,\ldots,d\}$. DT predictions T_F for input samples x are indicated by $T_F(x_F)$, where x_F is the feature subvector restricted to indices in F.

Therefore, it can be said that each pair of classes in the dataset generates a set of *DT*s where each candidate tree is produced by the following:

1) Random subspace selection: pick a subset of feature indices, F, where the size of F is determined as shown in Equation (4):

$$F_{size} = \alpha d, \tag{4}$$

where

$\alpha \in [0,1]$: percentage of features to be selected

d: number of features in dataset D.

Each candidate DT is trained on a random subset of features to ensure diversity. Let F be a randomly selected set of feature indices, where the size of F is determined by a sampling ratio α. A tree T_F is then trained using only the features indexed by F.

2) Bootstrap sampling [22]: at this stage, we draw a bootstrap sample $\mathcal{S}$ of the pair-wise data $D_{i,j}$.
3) Training each of generated DTs (T_F) on $D_{i,j}$. This stage returns a set of candidate detectors $\mathcal{T}_{i,j}$ shown in Equation (5).

$$\mathcal{T}_{i,j} = \{(T_{F1}, F_1), \ldots, (T_{FM}, F_M)\}, \tag{5}$$

where m is the number of trees for each pair ($M = 2$ in this research).

3.2.3. Heuristic filtering process

This stage constitutes the principal contribution of the HGS-RF model. Unlike standard *RF*, which retains all trees in $\mathcal{T}_{i,j}$, HGS-RF subjects each candidate detector (T_F, F) to a performance-based filtering process.

The filter evaluates whether a tree is a specialized detector for either of the two classes that it was trained to distinguish. For a given candidate tree (T_F, F) trained on the pair (c_i, c_j), we assess its performance on each class separately.

The candidate detectors are then filtered so that a detector is retained only if it reliably recognizes at least one of the two classes (i.e., if it has a high true positive rate in the "self" examples). The procedure is implemented as follows:

For a detector (T_F, F) and class $c \in \{c_i, c_j\}$, the self set $\mathcal{S}_c$ is defined as shown in Equation (6). $\mathcal{S}_c$ consists of all samples in $D_{i,j}$ that truly belong to class c:

$$\mathcal{S}_c = \{(x, y) \in D_{i,j} \mid y = c. \tag{6}$$

Compute the fraction of self-samples predicted as class c (Accuracy) as shown in Equation (7).

$$Acc_{self}(T_F, c) = \frac{\sum_{(x,y)\in \mathcal{S}} [T_F(X_F)=c]}{\text{Size of } \mathcal{S}_c}. \tag{7}$$

A tree is considered a valid specialized detector for class c if its accuracy for that class meets or exceeds a predefined performance threshold τ. The core heuristic retention rule is the following: a tree is kept for the final ensemble if it qualifies as a valid detector for at least one of the two classes in its pair.

Then, the valid detector is determined, i.e., c if Acc_{self} $(T_F, c) \geq \tau$, where τ is a threshold (in this research, $\tau = 0.95$). The detector is considered valid if it is valid for at least one of the two classes in the pair. This can be expressed mathematically as shown in Equation (8).

$$Keep\ detector\ (T_F, F) \Leftrightarrow \exists c \in \{c_i, c_j\} : Acc_{self}(T_F, c) \geq \tau. \tag{8}$$

This process results in a set of filtered detectors $\tilde{\mathcal{T}}_{i,j} \in \mathcal{T}_{i,j}$ that achieve high accuracy in detecting one or more pairs of classes. This filtering method reduces spurious detectors that do not reliably recognize their intended class (self), and this prioritizes a high true positive coverage for at least one class.

This heuristic filter ensures that the final ensemble is composed exclusively of trees that have proven high proficiency in recognizing at least one specific class of traffic, effectively eliminating weak, non-specialized, or noisy learners. All filtered detectors are aggregated into a final, global detector bank: $\mathcal{B} = \cup_{i<j} \mathcal{T}_{i,j}$.

3.2.4. Ensemble predictions

Each tree votes $V(c,x)$ for one of the two classes in its pair. The final prediction is the class with the maximum votes, as shown in Equation (9).

The final prediction is determined by a majority vote. For a sample x, each tree in the global detector bank $\mathcal{B}$ casts a vote for one of the two classes that it was specialized to distinguish. Let $V(c,x)$ denote the total votes received by class c. The predicted class $\hat{y}(x)$ is the one that receives the most votes:

$$\hat{y}(x) = argmax_{c \in C}\ V(c,x), \tag{9}$$

where $V(c, x)$ is the total number of votes received by class c for sample x.

During inference, for a new sample x, each tree in bank $\mathcal{B}$ casts a vote for one of the two classes that it was specialized to distinguish.

The final predicted class $\hat{y}(x)$ is determined by a majority vote across all trees.

If multiple classes share the highest vote count, the tie is resolved by prioritizing the “Normal” class to minimize disruption to legitimate IoT network operations. If the “Normal” class is not among the tied classes, the tie is broken by selecting the class with the lower index c.

3.2.5. Workflow and implementation

Figure 2 shows the pseudocode for the HGS-RF training algorithm.

Figure 3 shows the workflow diagram of the proposed methodology.

3.3. Evaluation metrics

We report the confusion matrix (CM) and the per-class and aggregated metrics used by classification_report [23]. By analyzing this matrix, we can pinpoint the types of errors that the model is making, refine them accordingly, and calculate the metrics shown in Equations (10), (11), (12), and (13) [24]. Figure 4 shows the confusion matrix in general.

Figure 2
The methodology pseudocode

```
Algorithm 1: HGS-RF Training
Input: Dataset D, Number of classes C, Feature sampling ratio α, Performance threshold τ, Trees per pair M
Output: Global Detector Bank $\mathcal{B}$
Initialize empty global detector bank $\mathcal{B} \leftarrow \emptyset$
for each unique class pair $(c_i, c_j)$ where i < j do
    // 1. Create binary dataset for the pair
    $D_{i,j} \leftarrow \{(x, y) \in D \mid y = c_i$ or $y = c_j\}$
    Initialize empty candidate set $T_{i,j} \leftarrow \emptyset$

    // 2. Generate M candidate trees for this pair
    for m = 1 to M do
        F ← RandomSubset(Features, size = α x total_features)
        S ← BootstrapSample($D_{i,j}$) // Sample with replacement
        $T_F$ ← TrainDecisionTree(S using features F)
        Add ($T_F$, F) to $T_{i,j}$
    end for

    // 3. Heuristic Filtering: Keep only high-performing trees
    Initialize empty filtered set $\tilde{\mathcal{T}}_{i,j} \leftarrow \emptyset$
    for each candidate tree ($T_F$, F) in $T_{i,j}$ do
        for each class c in $\{c_i, c_j\}$ do
            $\mathcal{S}_c \leftarrow \{(x, y) \in D_{i,j} \mid y = c\}$
            $Acc_{self}(T_F, c)$ ← CountCorrectPredictions
            if $Acc_{self}(T_F, c) \geq \tau$ then
                Add ($T_F$, F) to $\tilde{\mathcal{T}}_{i,j}$

            end if
        end for
    end for

    // 4. Add filtered trees to the global bank
    $\mathcal{B} \leftarrow \mathcal{B} \cup \tilde{\mathcal{T}}_{i,j}$
end for
return $\mathcal{B}$
```

Figure 3
The workflow diagram of the proposed methodology

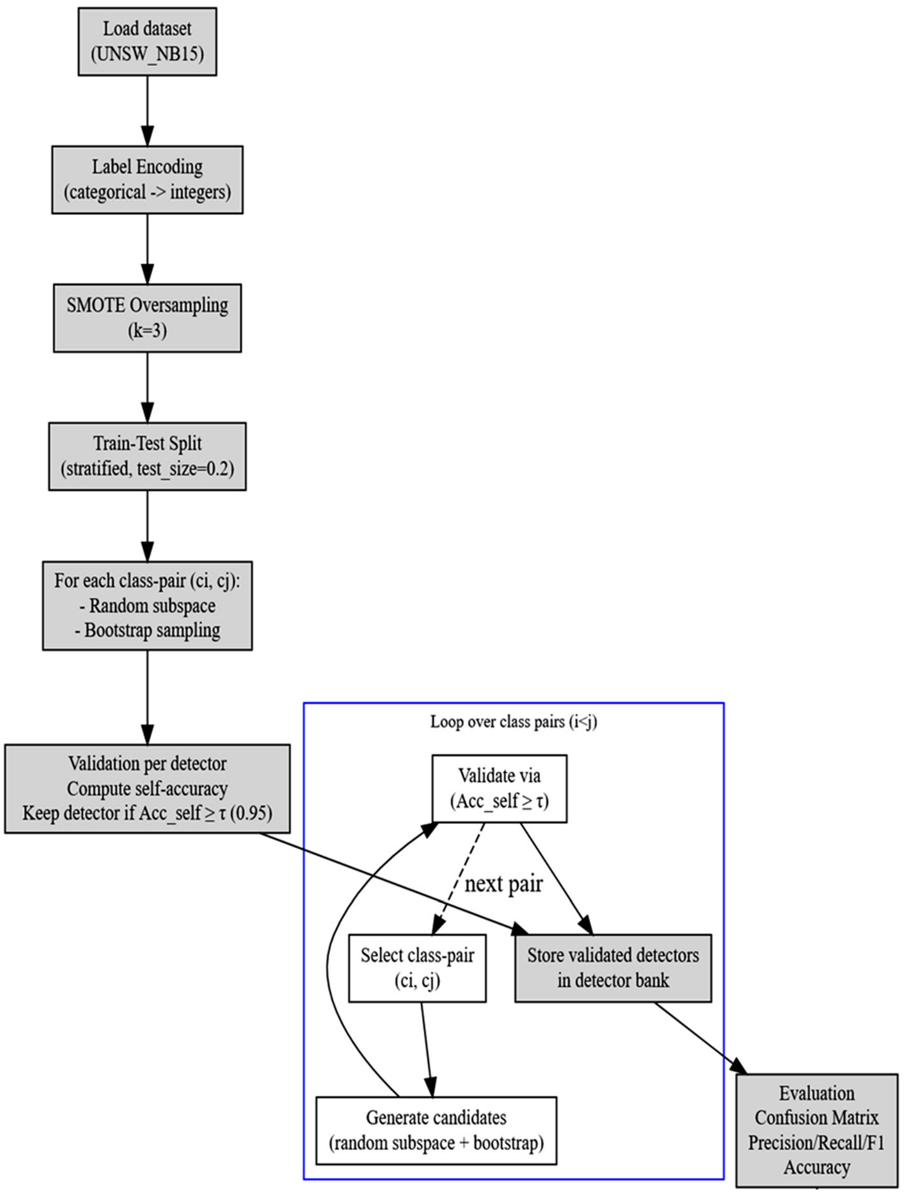

Load dataset (UNSW_NB15)
Label Encoding (categorical -> integers)
SMOTE Oversampling (k=3)
Train-Test Split (stratified, test_size=0.2)
For each class-pair (ci, cj): - Random subspace - Bootstrap sampling
Validation per detector Compute self-accuracy Keep detector if Acc_self ≥ τ (0.95)
Loop over class pairs (i<j)
Validate via (Acc_self ≥ τ)
next pair
Select class-pair (ci, cj)
Store validated detectors in detector bank
Generate candidates (random subspace + bootstrap)
Evaluation Confusion Matrix Precision/Recall/F1 Accuracy

Figure 4
The confusion matrix in general

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

$$Precision = \frac{TP}{TP+FP}. \quad (10)$$

$$Recall = \frac{TP}{TP+FN}. \quad (11)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad (12)$$

$$F1_{Score} = 2 \times \frac{Precision \times Recall}{Precision+Recall}. \quad (13)$$

4. Results and Discussions

4.1. Experimental setup and overall performance

The proposed model and all baseline algorithms were implemented in Python. The experiments were conducted on Google Colab platform. The overall performance measured by accuracy, precision, recall, and F1-score is summarized in Table 3. The proposed model achieved the highest accuracy of 93.4% in the representative single run detailed in Table 3. A comprehensive statistical analysis across 30 independent runs confirms the robustness of this result, showing a mean accuracy of 93.28% ± 0.72%, which statistically significantly outperforms the baseline *RF* (90.12% ± 0.85%). The overall F1-score of 0.7688 indicates a robust balance between precision and recall across all classes. This good F1-score value is considered a significant challenge in imbalanced intrusion detection tasks.

An overall comparison with classic ML models, as shown in Table 3, shows the superiority of the ensemble approach. Models such as Naïve Bayes, logistic regression, and SVM performed poorly, with accuracies below 65%. Therefore, these models are not suitable for complex and nonlinear patterns in IoT traffic. The DT performed better (88% accuracy).

Table 3
A comparison of the proposed model with ML algorithms

Algorithm	Precision	Recall	F1-score	Accuracy
Decision tree	0.564	0.543	0.550	88%
Naïve Bayes	0.252	0.212	0.158	41%
Logistic regression	0.120	0.184	0.144	61%
SVM	0.232	0.188	0.152	63%
RF	0.658	0.552	0.6	90%
Proposed model	0.741	0.7988	0.7688	93.4%

Our model consistently outperformed the baseline *RF* (90% accuracy and 0.6 F1-score) across all metrics. This performance gain is achieved by the intelligent selection of trees, not by using more trees. The heuristic filtering process effectively generates a "group of experts," leading to more accurate and reliable collective decisions.

4.2. Analysis of class-wise performance

Figure 5 shows the proposed methodology for CM.

The class-wise performance shown in Table 4 presents the strengths in detecting a wide range of attacks. The model achieved exceptional capability (F1 > 0.9) for well-defined classes such as Normal, Generic, Fuzzers, and Reconnaissance and substantial improvement in detecting complex attack classes compared to the baseline RF, as shown in Table 5.

We note from Table 4 and Table 5 that the recall value of Backdoor improved from 0.01 (RF) to 0.701 (proposed model) and that for Worms improved from 0.12 to 0.889. This high improvement is attributed to the heuristic filter, which selectively stored trees that are highly sensitive to these subtle attacks. While (Analysis and DoS) classes remained challenging (F1 of 0.366 and 0.479, respectively), our model still achieved a marked improvement over RF, which achieved (F1 of 0.05 and 0.37, respectively). The lower performances on these classes show that their patterns are highly variable or overlap significantly with other traffic.

Figure 5
The proposed methodology CM

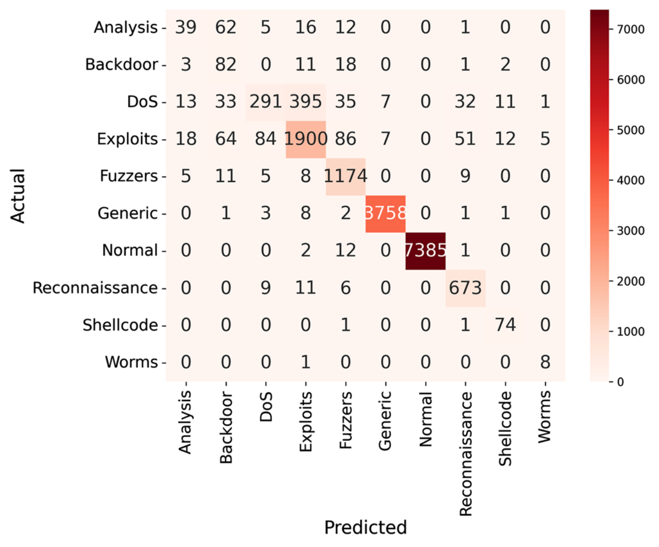


Table 4
The performance metrics for each class of the proposed methodology

Attack type	Precision	Recall	F1-score
Analysis_Att	0.5	0.289	0.366
Backdoor_Att	0.324	0.701	0.443
DOS_Att	0.733	0.356	0.479
Exploits_Att	0.808	0.853	0.830
Fuzzers_Att	0.872	0.969	0.918
Generic_Att	0.996	0.996	0.996
Normal_Att	1.00	0.998	0.999
Reconnaissance_Att	0.874	0.963	0.916
Shellcode_Att	0.740	0.974	0.841
Worms_Att	0.571	0.889	0.696

Table 5
The performance metrics for each class of the RF algorithm

Attack type	Precision	Recall	F1-score
Analysis_Att	0.07	0.04	0.05
Backdoor_Att	0.01	0.01	0.01
DOS_Att	0.40	0.35	0.37
Exploits_Att	0.70	0.78	0.74
Fuzzers_Att	0.87	0.85	0.86
Generic_Att	1.00	0.97	0.99
Normal_Att	1.00	1.00	1.00
Reconnaissance_Att	0.82	0.81	0.82
Shellcode_Att	0.71	0.59	0.64
Worms_Att	1.00	0.12	0.22

Figure 6 shows a comparison between the proposed model and the RF in terms of F1-score for each class.

Figure 6
A comparison between the proposed model and the RF in terms of F1-score for each class

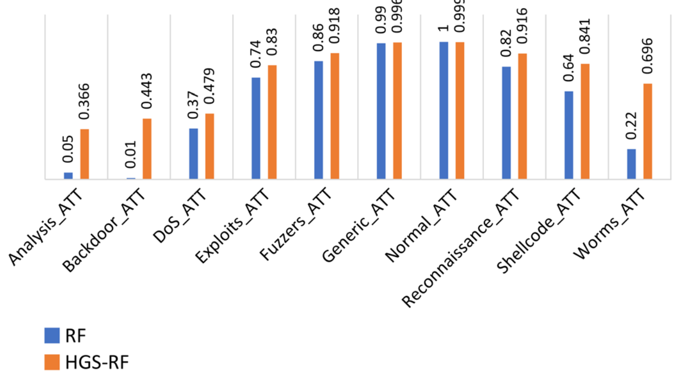


Table 6
A comparison of the proposed model with the RF in terms of hyperparameters

Hyperparameter	Description	Baseline RF	Proposed model
n_estimators	Number of decision trees generated in the forest	100	85
Criterion	The function used to measure the quality of a split	gini	gini
min_samples_split	The minimum sample count required to split a node	2	2
min_samples_leaf	The minimum sample count required to be at a leaf node	1	1
Bootstrap	Whether bootstrap samples are used when building trees. If false, the whole dataset is used to build each tree	True	True
max_features	Feature count that is considered for the best split	80%	80%
max_depth	The max-depth for each tree. None refers to the node expanded until all leaves are pure or until all leaves contain fewer than min_samples_split samples	None	Random (20,50)
min_weight_fraction_leaf	The minimum weighted fraction of the sum total of weights (of all input samples) required to be at a leaf node. Samples have equal weights when sample_weight is not provided	0	0
max_features	The number of features to consider when looking for the best split	0.8	0.8
max_leaf_nodes	Grow trees with max_leaf_nodes in best-first fashion	None	None
min_impurity_decrease	A node will be split if this split induces a decrease of the impurity greater than or equal to this value	0	0
class_weight	Weights associated with classes in the form {class_label: weight}	None	None
Parameters after training			
Hyperparameter	**Description**	**Baseline RF**	**Proposed model**
Min depth result	Minimum tree depth among the resulting trees	43	20
Max depth result	Maximum tree depth among the resulting trees	86	50
Number of saved trees	Number of final decision trees in the forest	100	85

The performance gains for underrepresented attacks were better. As shown in Figure 6, the proposed model achieved better improvements in F1 for minority classes. For instance, Backdoor improved from 0.01 (RF) to 0.443, and Worms improved from 0.22 to 0.696. This shows the efficacy of the heuristic filtering mechanism in selecting and retaining trees that are sensitive to these subtle and sparse attack patterns.

4.3. The impact of heuristic filtering

The core contribution of the heuristic filter is validated by analyzing the model characteristics (see Table 6). While the baseline RF generated and stored 100 trees, our model applied the filtering process, which resulted in a more efficient ensemble of only 85 trees. This 15% reduction in model size directly translates to a lower memory consumption for storage, which is considered a critical advantage for IoT devices with limited resources.

Furthermore, the depth of the trees in our model was controlled (random between 20 and 50), so the final model has a maximum depth of 50 compared to 86 in RF. This demonstrates that the proposed model achieved higher accuracy not with more complex trees but with a more intelligent ensemble strategy that discards underperforming trees, resulting in a more generalized and efficient model.

While the basic RF model generated and stored 100 trees, the filtering process in the HGS-RF model resulted in a more efficient set of only 85 trees. This 15% reduction in model size is significant in resource-constrained IoT environments.

Table 7 shows a quantitative comparison of efficiency metrics between baseline RF and the proposed HGS-RF.

Table 7
A quantitative comparison of efficiency metrics between baseline RF and the proposed HGS-RF

Metric	Baseline RF	HGS-RF	Improvement
Number of trees	100	85	15% reduction
Model storage size	4.0 MB	3.1 MB	22.5% reduction
Avg. inference time per sample	1.53 ms	1.25 ms	18.3% reduction

4.4. Statistical robustness analysis

We evaluated the proposed HGS-RF model and the baseline RF across 30 independent runs. In each run, a new stratified train–test split (80%–20%) was generated with a different random seed. This procedure accounts for variability arising from data sampling and model initialization. Table 8 presents the aggregated results, reporting the mean ± standard deviation of the overall accuracy, precision, recall, and F1-score.

4.5. Discussions

The results demonstrate that the proposed methodology offers clear improvements in both class-level and overall performances. As shown in Table 4, the method achieved high precision and recall for critical attack categories. In contrast, more challenging classes, such

Table 8
Statistical comparison of overall performance across 30 independent runs

Algorithm	Accuracy (%)	Precision	Recall	F1-score
Random forest	90.12 ± 0.85	0.651 ± 0.012	0.548 ± 0.015	0.595 ± 0.011
Proposed HGS-RF	93.28 ± 0.72	0.737 ± 0.010	0.795 ± 0.011	0.765 ± 0.008

as Analysis and DoS, recorded lower recall values, suggesting that the complex variability of these attacks still poses difficulties even for the enhanced model.

When comparing the proposed model directly to RF in Table 5, the proposed approach showed notable gains, especially in minority classes. For instance, the Backdoor and Worms categories, which were poorly recognized by the baseline RF, benefited significantly from the detector selection strategy, leading to better recall without sacrificing precision. Table 6 further highlights that the new method achieves better performance while retaining fewer trees than the baseline, which reflects an efficiency advantage in terms of storage and computation.

The comparison in Table 3 confirms these findings, where the proposed method achieved the highest overall accuracy (93.4%) while classic models such as Naïve Bayes, logistic regression, and SVM struggled to generalize. These results highlight that integrating a biologically inspired filtering process with DT ensembles can create a more robust classifier. Nevertheless, the relatively modest performance on certain classes indicates that future refinements should explore adaptive thresholding or hybrid feature extraction methods to strengthen detection in difficult categories further.

The results collectively show that the proposed model successfully achieved its dual objective: enhancing detection performance and improving computational efficiency. The immune-inspired heuristic directly addresses the key weakness of baseline RF, which is the inclusion of weak learners, by implementing a quality-over-quantity approach.

The performance on classes such as Analysis and DoS indicates a potential direction for future work. These classes may require specialized feature engineering or a dynamically adjusted threshold (τ) to further improve their detection rates without compromising the model's overall efficiency.

Furthermore, the statistical robustness analysis (Section 4.4) shows that the performance gains of HGS-RF are consistent across multiple random data splits with a low standard deviation.

5. Conclusion

This study introduced a heuristic-based ensemble method that builds upon the principles of DT detectors and biologically inspired filtering to improve intrusion detection performance. The model achieves a balance between prediction power and computational efficiency by focusing on detectors that achieve accuracy above a certain threshold and excluding weak detectors with detection accuracy below the threshold value, which can affect the final decision and consume memory space for storage. The results showed that the proposed approach outperformed several traditional ML algorithms and the standard baseline RF with an accuracy of 93.4% with a single run. After 30 runs, the model achieved an overall accuracy of 93.28% ± 0.72%, a statistically significant improvement over the baseline RF (90.12% ± 0.85%). The most important results are the improvements in detecting minority attacks, increasing recall for Backdoor from 0.01 to 0.701 and for Worms from 0.12 to 0.889 while simultaneously reducing model memory usage by 22.5% and inference time by 18.3%.

This study also revealed limitations where certain classes, such as Analysis and DoS, remained challenging. Therefore, future work will explore several promising directions:

1) Dynamic thresholding: implementing adaptive threshold (τ) strategies, such as thresholds that adjust based on class prevalence or tree depth.
2) Designing a pipeline where a lightweight deep feature extractor (e.g., a small CNN or autoencoder) processes raw traffic or sequence data to generate enriched features, which are then fed into the HGS-RF ensemble for final classification.
3) Testing the trained HGS-RF model on real IoT hardware (e.g., Raspberry Pi and ESP32) to validate its performance and energy consumption in a true constrained environment.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in the UNSW-NB15 dataset at https://doi.org/10.1109/MilCIS.2015.7348942.

Author Contribution Statement

Saad AL Azzam: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Ghassan ALDharhani:** Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Raenu AL Kolandaisamy:** Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Supervision, Project administration.

References

[1] Mishra, P., & Singh, G. (2023). Internet of medical things healthcare for sustainable smart cities: Current status and future prospects. *Applied Sciences*, *13*(15), 8869. https://doi.org/10.3390/app13158869

[2] Isong, B., Kgote, O., & Abu-Mahfouz, A. (2024). Insights into modern intrusion detection strategies for Internet of Things

ecosystems. *Electronics*, *13*(12), 2370. https://doi.org/10.3390/electronics13122370

[3] Pavithra, S., & Vikas, K. V. (2024). Detecting unbalanced network traffic intrusions with deep learning. *IEEE Access*, *12*, 74096–74107. https://doi.org/10.1109/ACCESS.2024.3405187

[4] Slimani, C., Wu, C. F., Rubini, S., Chang, Y. H., & Boukhobza, J. (2022). Accelerating random forest on memory-constrained devices through data storage optimization. *IEEE Transactions on Computers*, *72*(6), 1595–1609. https://doi.org/10.1109/TC.2022.3215898

[5] Noura, H. N., Allal, Z., Salman, O., & Chahine, K. (2025). An optimized tree-based model with feature selection for efficient fault detection and diagnosis in diesel engine systems. *Results in Engineering*, *27*, 106619. https://doi.org/10.1016/j.rineng.2025.106619

[6] Beaurivage, J., Ouameur, M. A., & Domingue, F. (2025). A memory representation of random forests optimized for resource-limited embedded devices. *IEEE Embedded Systems Letters*. https://doi.org/10.1109/LES.2025.3574563

[7] Maoudj, S. E., & Belghiat, A. (2025). A deep learning-based approach with two-step minority classes prediction for intrusion detection in Internet of Things networks. *Knowledge-Based Systems*, *312*, 113143. https://doi.org/10.1016/j.knosys.2025.113143

[8] Mohamed, T. S., & Aydin, S. (2022). IoT-based intrusion detection systems: A review. *Smart Science*, *10*(4), 265–282. https://doi.org/10.1080/23080477.2021.1972914

[9] Bahaa, A., Abdelaziz, A., Sayed, A., Elfangary, L., & Fahmy, H. (2021). Monitoring real time security attacks for IoT systems using DevSecOps: A systematic literature review. *Information*, *12*(4), 154. https://doi.org/10.3390/info12040154

[10] Tabassum, A., Erbad, A., Mohamed, A., & Guizani, M. (2021). Privacy-preserving distributed IDS using incremental learning for IoT health systems. *IEEE Access*, *9*, 14271–14283. https://doi.org/10.1109/ACCESS.2021.3051530

[11] Shakhov, V., Jan, S. U., Ahmed, S., & Koo, I. (2019). On lightweight method for intrusions detection in the Internet of Things. In *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 1–5. https://doi.org/10.1109/BlackSeaCom.2019.8812813

[12] Jan, S. U., Ahmed, S., Shakhov, V., & Koo, I. (2019). Toward a lightweight intrusion detection system for the Internet of Things. *IEEE Access*, *7*, 42450–42471. https://doi.org/10.1109/ACCESS.2019.2907965

[13] Roy, S., Li, J., Choi, B. J., & Bai, Y. (2022). A lightweight supervised intrusion detection mechanism for IoT networks. *Future Generation Computer Systems*, *127*, 276–285. https://doi.org/10.1016/j.future.2021.09.027

[14] Susilo, B., Muis, A., & Sari, R. F. (2025). Intelligent intrusion detection system against various attacks based on a hybrid deep learning algorithm. *Sensors*, *25*(2), 580. https://doi.org/10.3390/s25020580

[15] Ain, N. U., Sardaraz, M., Tahir, M., Abo Elsoud, M. W., & Alourani, A. (2025). Securing IoT networks against DDoS attacks: A hybrid deep learning approach. *Sensors*, *25*(5), 1346. https://doi.org/10.3390/s25051346

[16] Soe, Y. N., Feng, Y., Santosa, P. I., Hartanto, R., & Sakurai, K. (2019). Implementing lightweight IoT-IDS on Raspberry Pi using correlation-based feature selection and its performance evaluation. In *International Conference on Advanced Information Networking and Applications*, *926*, 458–469. https://doi.org/10.1007/978-3-030-15032-7_39

[17] Fenanir, S., Semchedine, F., & Baadache, A. (2019). A machine learning-based lightweight intrusion detection system for the Internet of Things. *Revue d'Intelligence Artificielle*, *33*(3). https://doi.org/10.18280/ria.330306

[18] Almalawi, A. (2025). A lightweight intrusion detection system for Internet of Things: Clustering and Monte Carlo cross-entropy approach. *Sensors*, *25*(7), 2235. https://doi.org/10.3390/s25072235

[19] Vibhute, A. D., Khan, M., Patil, C. H., Gaikwad, S. V., Mane, A. V., & Patel, K. K. (2024). Network anomaly detection and performance evaluation of convolutional neural networks on UNSW-NB15 dataset. *Procedia Computer Science*, *235*, 2227–2236. https://doi.org/10.1016/j.procs.2024.04.211

[20] Hassouneh, N., & Al-sharaeh, S. (2025). Intrusion detection in IoT networks using LSTM deep learning models with the UNSW-NB15 dataset. In *2025 International Conference on New Trends in Computing Sciences*, 263–269. https://doi.org/10.1109/ICTCS65341.2025.10989358

[21] Amirruddin, A. D., Muharam, F. M., Ismail, M. H., Tan, N. P., & Ismail, M. F. (2022). Synthetic minority over-sampling technique (SMOTE) and logistic model tree (LMT)-adaptive boosting algorithms for classifying imbalanced datasets of nutrient and chlorophyll sufficiency levels of oil palm (*Elaeis guineensis*) using spectroradiometers and unmanned aerial vehicles. *Computers and Electronics in Agriculture*, *193*, 106646. https://doi.org/10.1016/j.compag.2021.106646

[22] Schonlau, M. (2023). Applied statistical learning. *Statistics and Computing*, 143–160. https://doi.org/10.1007/978-3-031-33390-3

[23] Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: Multi-label confusion matrix. *IEEE Access*, *10*, 19083–19095. https://doi.org/10.1109/ACCESS.2022.3151048

[24] Takahashi, K., Yamamoto, K., Kuchiba, A., & Koyama, T. (2022). Confidence interval for micro-averaged F_1 and macro-averaged F_1 scores. *Applied Intelligence*, *52*(5), 4961–4972. https://doi.org/10.1007/s10489-021-02635-5

How to Cite: AL Azzam, S., ALDharhani, G., & AL Kolandaisamy, R. (2026). Heuristic-Guided Selective Random Forests: Immune-Inspired Heuristic Ensembles for Robust Intrusion Detection in IoT Networks. *Artificial Intelligence and Applications*. https://doi.org/10.47852/bonviewAIA62027781