Sistemasi: Jurnal Sistem Informasi ISSN:2302-8149
Volume 15, Nomor 10, 2026: 4286-4298 e-ISSN:2540-9719

Ensembles within Ensembles: Nested Boosting and Stacking for Smart-Home IoT Intrusion Detection

[1]Tarfa Hamed*
[1]Department of Networks, College of Computer Science and Mathematics, University of Mosul
Mosul, Iraq
*e-mail: tyaseen@uomosul.edu.iq

(***received*:**14 Octoer 2026, ***revised*:** 29 October 2026,***accepted*:**30 October 2026)

Abstract

Smart home networks connect many low-cost IoT devices that exchange traffic continuously. This is a very attractive target for attackers and a difficult setting for intrusion detection. In this paper we investigate a hierarchical dependent ensemble approach in which one dependent ensemble algorithm is nested inside another. Three algorithms, AdaBoostM1, MultiBoostAB, and Stacking, were combined in all six possible outer-inner configurations and evaluated in WEKA on the DS2OS smart home traffic dataset using stratified 3-fold cross-validation, with the dataset's naturally imbalanced class distribution left intact. Five of the six configurations reached near-perfect results, with accuracy between 99.9964% and 99.9997%, Kappa between 0.9993 and 0.9999, and a false alarm rate of effectively zero. The AdaBoostM1→MultiBoostAB configuration, which relied on OneR rather than J48 as its weak learner, achieved 99.11% accuracy and a Kappa of 0.837, and it failed to detect the Data Type Probing and Wrong Setup classes altogether. Confusion matrix analysis showed that the nesting order of the ensembles had little effect on detection once the base classifier was held constant, whereas the choice of weak learner had a large one. Configurations with Stacking as the outer ensemble were also the fastest to train. Finally, we show that the near-ceiling scores are likely inflated by device-identifier attributes in DS2OS and discuss what this means for how intrusion detection results on this dataset should be reported and interpreted.

Keywords: AdaBoostM1, ensemble learning, Internet of Things (IoT), intrusion detection, MultiBoostAB, stacking.

1 Introduction

Smart homes connect devices such as temperature controllers, lighting systems, and window actuators so that they can be monitored and controlled over a shared network. The same connectivity that makes these homes convenient also exposes them to attack. A compromised device can be used to scan the network, spy on the occupants, or issue malicious control commands, which are the kinds of behavior captured by smart home benchmarks such as DS2OS [1]. Studies that focus directly on the smart home have shown that attacks including host discovery and SYN, ACK, and HTTP flooding can be recognized from device traffic [2], which makes intrusion detection systems (IDSs) a practical line of defense in this environment.

Building an accurate IDS for this setting is not straightforward. Home devices usually have limited computing resources, so a detection model must be efficient as well as accurate [1], [3]. Traffic from different devices is heterogeneous, and attack traffic is usually much rarer than normal traffic, and a model can report a high overall accuracy while missing entire attack classes. Much of the earlier IDS work also treated detection as a binary problem, distinguishing benign from malicious traffic without identifying the type of attack [4], even though knowing the attack type is what allows a meaningful response.

Ensemble learning is now one of the most important methods for meeting these needs. Similar studies found that ensemble classifiers always outperform single models, such as KNN, SVM, and decision trees for IoT intrusion detection [5], and bagging and boosting ensembles enhance multiclass detection [4], including on recent and unbalanced benchmarks such as CICIoT2023 [6]. Boosting has been applied to smart home traffic with XGBoost [2] and with LightGBM supported by majority voting [1]. Stacking has been used to combine feature selection with hyperparameter optimization for lightweight detection [3], to fuse several deep learning models [7], and to merge the outputs of

 ISSN:2302-8149
e-ISSN:2540-9719

multiple feature-selection algorithms [8]. Other studies have built hybrid or multi-stage pipelines that pair ensembles with autoencoders and transformer-based models [9] or separate fast binary screening from detailed attack identification at the IoT edge [10].

Two gaps remain in this body of work. First, nearly all of these models place a single ensemble layer on top of a base classifier. Classic dependent ensembles such as AdaBoostM1 [11], MultiBoostAB [12], and stacked generalization [13] can themselves serve as the base learner of another ensemble, yet little is known about how such nested combinations behave for intrusion detection, or whether the order of nesting matters. Second, most of the studies above are evaluated on general IoT or network datasets rather than smart home traffic [3]–[5], and very high reported accuracy is rarely examined to check whether it reflects genuine attack behavior or artifacts of how the dataset was generated.

This paper addresses both gaps by studying a hierarchical dependent ensemble model for intrusion detection in IoT-based smart homes. AdaBoostM1, MultiBoostAB, and Stacking are arranged in a two-level structure in which one algorithm acts as the outer (meta-level) ensemble and a different algorithm acts as the inner (base-level) ensemble. This produces six outer-inner configurations, each of which is evaluated on the DS2OS smart home dataset in its original, imbalanced form. The main contributions of this paper are as follows:

1) This work presents a two-level dependent ensemble framework that nests AdaBoostM1, MultiBoostAB, and Stacking in all six outer-inner combinations, and evaluates them on smart home traffic from the DS2OS dataset.
2) A class-level evaluation has been carried out for every configuration, using per-class F1-score, recall, ROC area, and confusion matrices. This makes it possible to see how the models actually behave on rare attack classes, something that aggregate accuracy tends to hide.
3) The analysis looked at nesting order and training cost. It shows that the choice of weak learner has a much bigger impact on detection than the order in which the ensembles are nested, and that configurations using Stacking as the outer ensemble train the fastest.
4) The work also offers a critical assessment of the near-perfect results on DS2OS. It points out that these scores are best treated as an upper bound rather than a realistic target, and it outlines what future directions can be pursued.

The rest of this paper is organized as follows. Section 2 reviews related work on ensemble-based intrusion detection for IoT and smart homes. Section 3 describes the three ensemble algorithms used in this study. Section 4 presents the research methodology, the model design, data preparation, and evaluation metrics in DS2OS, and then we report and discuss our results in Section 5. Section 6 ends the paper and sets some future direction for future work.

2 Literature Review

Ensemble learning has become a key strategy for intrusion detection in IoT and smart home environments. It can improve classification performance, reduce false alarms, and handle heterogeneous traffic more effectively than single models. This technique matters especially in smart homes, where device traffic is diverse, attack patterns vary widely, and computational resources are often limited. Recent works have focused on stacked, boosted, and hybrid ensemble pipelines as well as feature-selection-driven designs [2], and recent surveys of IoT anomaly detection also confirm that ensemble learning improves detection accuracy, and identify DS2OS as one of the commonly used benchmark datasets [14].

The study in [1] focused on the smart home domain and evaluated several machine learning classifiers on the DS2OS dataset. The authors benchmarked Logistic Regression, Random Forest, *XGBoost*, and LightGBM and then proposed an LGB-IDS model that is based on ensemble techniques and majority voting. The strength of this work is its direct relevance to IoT-enabled smart homes and its focus on accuracy and time efficiency.

The authors in [2] presented a smart-home IDS based on *XGBoosting* and evaluated it on the IoT-NI dataset. The authors reported that the model can classify a variety of different attack types such as host discovery, SYN, ACK, and HTTP flooding and has good precision when cross-validating. Its strength lies in being explicitly tailored to smart-home devices and attack traffic [2].

 ISSN:2302-8149
e-ISSN:2540-9719

The article in [3] proposed a stacked ensemble IDS for IoT traffic and combined feature selection with hyperparameter optimization to reduce computational overhead. The reported results on NB-AIoT and UNSW-NB15 showed very high accuracy, which makes the work relevant for lightweight security models in constrained IoT settings. Its strength is the balance between performance and efficiency; while its limitation is that the study is not centered on smart-home traffic and does not directly address household deployment conditions [3].

The work presented in [7] introduced DIS-IoT, a stacking ensemble of deep learning models evaluated on ToN_IoT and CICIDS2017 in both binary and multi-class settings. The method was significant since it demonstrated that deep stacked ensembles can achieve low false positive rates and strong multi-class performance. Its main strength is the ability to model complex traffic patterns, while the limitation is the increased complexity, which may make real-world deployment harder in resource-constrained smart homes [7].

In [9], the authors built a hybrid framework that brings together stacked autoencoders, CatBoost, and an optimized transformer *CNN-LSTM* ensemble. Its main strength is that one pipeline handles feature extraction, feature selection, and ensemble classification, and it achieved high accuracy on three well-known datasets: NSL-KDD, UNSW-NB15, and AWID. However, the framework was designed for intrusion detection in general rather than for smart homes. Its results would need to be adapted and validated before they could be applied to domestic IoT environments with any confidence [9].

In [8], the authors set out to improve feature selection by stacking several feature-selection algorithms together, with logistic regression serving as the meta-learner. The features chosen this way were then fed into a *CNN*-based IDS, further strengthened with *BiLSTM* layers and attention mechanisms. The study does well to address noisy features and detection performance together. Its gap, however, is that it was tested on general IoT intrusion datasets rather than on traffic from smart-home settings [8].

The study in [4] looked at how ensemble machine-learning methods perform in multiclass intrusion detection for IoT environments. The authors tested bagging and boosting decision-tree techniques on IoT telemetry datasets. Their goal went beyond separating benign from malicious traffic to telling different attack categories apart. The work's main contribution is showing that ensemble classifiers can improve multiclass detection. This addresses a weakness in many earlier IDS studies, which mostly dealt with binary classification. Still, the work covers IoT security broadly rather than smart homes, so its direct relevance to domestic IoT traffic is limited, and further validation in smart-home settings is needed [4].

In [6], the authors addressed multiclass intrusion detection for IoT traffic using the CICIoT2023 benchmark, a dataset that captures both the variety of modern IoT attacks and the class imbalance found in real traffic. They tested several ensemble-based machine-learning methods and found that these outperformed simple single-model approaches at detecting attacks across multiple categories. A key strength of the study is its use of a recent and demanding dataset, which makes the findings more relevant to today's IoT security challenges. Even so, the work targets general IoT networks rather than smart homes, so more validation is needed before it can be put into practice in domestic IoT environments [6].

The authors in paper [10] proposed a two-stage intrusion detection framework for IoT-edge environments, where the first stage used an Extra Tree classifier for binary detection and the second stage applied an ensemble of Extra Tree, deep neural network, and random forest models for more detailed attack identification. The main contribution of the work is its dual-level design that is well suited for edge-based IoT security because it separates fast initial screening from more refined classification. Its strength lies in demonstrating improved detection performance across multiple benchmark datasets, including Bot-IoT, CICIDS2018, NSL-KDD, and IoTID20, but its limitation is that the study is oriented toward general IoT-edge systems rather than smart-home networks, so further validation in domestic IoT settings is still needed [10].

The paper [5] compared several machine learning classifiers for IoT intrusion detection and showed that ensemble methods consistently outperform single models. Its main contribution is the systematic benchmarking of *KNN*, Gradient Boosting, *XGBoost*, SVM, Random Forest, Decision Tree and Extremely Randomized Trees (ERT), where ensemble methods have the best overall performance in terms of accuracy, precision, recall and *F1-score*. The strongest result reported is that ERT

 ISSN:2302-8149
e-ISSN:2540-9719

performs especially well for real-time attack detection, making it useful for IoT security benchmarking, but its limitation is that it is a comparative evaluation rather than a new smart-home-specific IDS design, so it still leaves a gap in direct domestic IoT deployment validation [5].

In [15], the authors proposed a majority-voting ensemble that combines five machine learning classifiers to detect DDoS attacks in IoT networks. They tested it on CICDDoS2019 and UNSW-NB15, and reported detection accuracies ranging from 99.87% to 100%. The main strength of this work is that it demonstrates how combining diverse classifiers can overcome the weaknesses of individual models. However, it focuses only on DDoS traffic in general IoT networks, and does not cover the broader range of attacks that appear in smart home environments.

Overall, the published studies show that ensemble learning is effective for IoT intrusion detection, especially when used in combination with feature selection, stacking, or boosting. There is a vast research gap to develop **lightweight, robust, and smart-home-specific IDS models** that validate on the benchmark datasets and can be effectively used in a realistic home environment.

3 Learning Algorithms

3.1 AdaBoostM1

AdaBoost.M1, introduced by Freund and Schapire, is a boosting algorithm that constructs a strong classifier by iteratively training a weak learner on reweighted versions of the training data. At each iteration t, the algorithm maintains a distribution D_t over the training instances, trains the weak learner to obtain a hypothesis h_t, computes a weight α_t proportional to the accuracy of h_t, and then increases the weights of misclassified instances so that subsequent learners focus more on previously hard examples. The final prediction is obtained by a weighted majority vote of all weak hypotheses [11]:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \quad (1)$$

where x is an input instance (feature vector), $h_t(x)$ is the prediction of the weak learner at iteration t (typically ±1 for binary classification), α_t is the confidence weight assigned to h_t based on its training error, and $H(x)$ is the final strong classifier obtained by taking the sign of the weighted sum of all weak hypotheses.

This procedure reduces both bias and variance and has been shown to significantly improve classification performance compared with single classifiers and bagging, particularly when combined with simple base learners [11].

3.2 MultiBoostAB

MultiBoosting, introduced by Webb, is an ensemble approach to AdaBoost combined with wagging (a weighted version of bagging) to overcome bias and variance. The algorithm constructs multiple AdaBoost sub-committees by applying AdaBoost to multiple weighted samples of the training data generated through wagging. In each sub-committee, we train the base learners consistently with instance reweighting as in AdaBoost and assign each learner a confidence weight based on the weighted error. The final prediction is obtained by aggregating the outputs of all sub-committees (typically weighted majority voting). This approach is able to make the most out of the strong bias-reduction capability of AdaBoost and variance reduction of wagging and therefore yields a generalization error that is lower than either AdaBoost or wagging alone [12].

3.3 Stacking

Stacked generalization, introduced by Wolpert, is an ensemble learning scheme that combines multiple base learners (level-0 models) by training a meta-learner (level-1 model) on their predictions. The method is based on splitting the training data (e.g., cross-validation) by training each of the base learners on some subset of the data and then taking the held-out predictions to the meta-learner and using those as inputs to the true labels. The meta-learner then learns how to combine the outputs of the base models to minimize the generalization error. This two-level system allows stacked generalization to leverage the strengths of different learners and can often be more accurate than simple voting or averaging schemes [13].

 ISSN:2302-8149
e-ISSN:2540-9719

4 Research Methodology

This section outlines the overall approach used to evaluate the proposed two-level dependent ensemble architecture. First of all, how the model is designed and configured in the six outer-inner combinations (Subsection 4.1), then the benchmark dataset used to evaluate it (Subsection 4.2), the steps taken to prepare the data prior to training (Subsection 4.3) and finally the metrics used to assess the classification performance (Subsection 4.4).

4.1 Model Designing

The proposed model follows the nested (two-level) ensemble architecture, in which one dependent ensemble algorithm acts as the outer, meta-level ensemble while a different algorithm acts as the inner, base-level ensemble. Given the three dependent ensemble algorithms considered in this study: AdaBoostM1, MultiBoostAB, and Stacking, and the constraint that the outer and inner algorithm must not be the same, six distinct outer-inner configurations are formed and evaluated. Each configuration is trained and tested using stratified *k-fold* cross-validation in WEKA, and its performance is recorded for later comparison. Algorithm 1 formalizes this process.

Algorithm 1 Two-Level Dependent Ensemble Intrusion Detection Model with 6 Ensemble Combinations

Input:
D: benchmark IoT intrusion detection dataset (DS2OS)
E: set of dependent ensemble algorithms, E = {AdaBoostM1, MultiBoostAB, Stacking}
B: base classifier used as the underlying weak learner
k: number of cross-validation folds
Output:
P: performance results (Accuracy, Precision, Recall, *F1-score*, *FAR*) for each of the 6 outer-inner ensemble combinations

```
Initialize P ← ∅
for each outer algorithm O ∈ E do
    for each inner algorithm I ∈ E, where I ≠ O do
        Model_I ← Build inner ensemble using I with base classifier B
        Model_O ← Build outer ensemble using O with Model_I as its base learner
        Partition D into k stratified folds
        for fold = 1 to k do
            Train Model_O on the k − 1 training folds
            Test Model_O on the held-out fold
            Record predictions
        end for
        Aggregate predictions across all k folds
        Compute Accuracy, Precision, Recall, F1-score, and FAR for the (O, I) combination
        P ← P ∪ {(O, I) → computed metrics}
    end for
end for
return P
```

The proposed dependent hierarchical ensemble learning framework is shown in Fig. 1.

 ISSN:2302-8149
e-ISSN:2540-9719

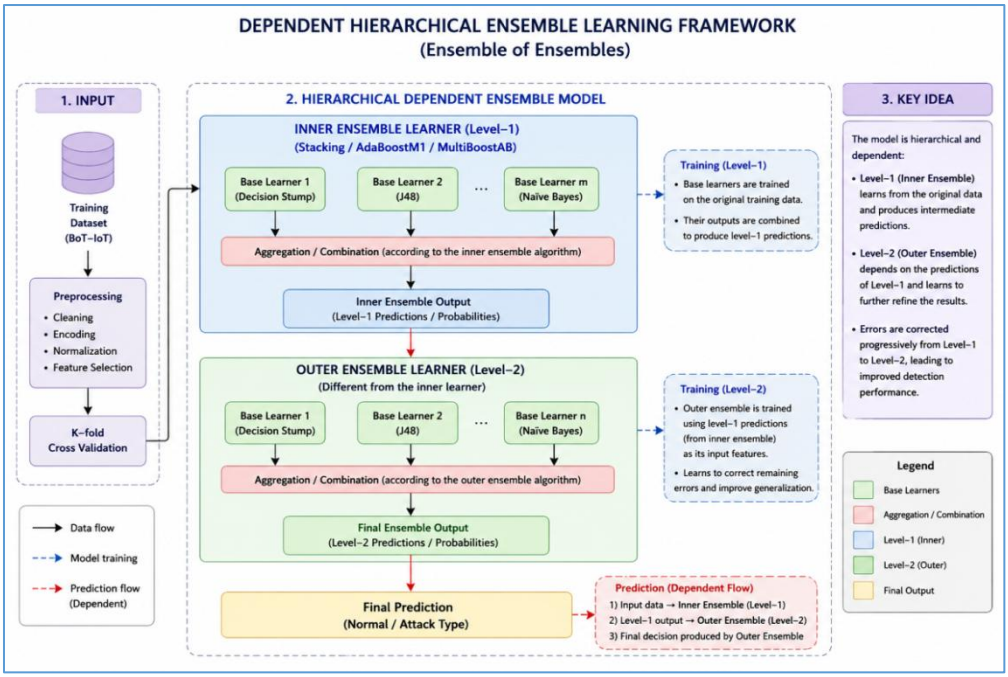

Figure 1 The dependent hierarchical ensemble learning framework

4.2 Dataset Description

The proposed model is evaluated using the **DS2OS (Distributed Smart Space Orchestration System) traffic traces dataset**, a benchmark dataset designed for intrusion detection research in IoT smart home environments. The dataset was generated by simulating a virtual IoT environment containing multiple smart-home services, such as temperature control, light control, and window control, with all communication between users and services captured and stored. It consists of 357,952 instances described by 13 features, and is labeled with eight classes: one *Normal* class and seven attack/anomaly classes, namely *Denial of Service* (*DoS*), *Data Type Probing*, *Malicious Control*, *Malicious Operation*, *Scan*, *Spying*, and *Wrong Setup*. Of the total instances, 347,935 belong to the Normal class while the remaining 10,017 instances represent anomalous activity, reflecting the natural, highly imbalanced nature of real-world intrusion traffic. The dataset's 13 attributes are: sourceID, sourceAddress, sourceType, sourceLocation, destinationServiceAddress, destinationServiceType, destinationLocation, accessedNodeAddress, accessedNodeType, operation, value, timestamp, and the class attribute, normality.

4.3 Data Preparation

Prior to training, the “DS2OS” dataset has been prepared for use in WEKA as follows. The raw CSV file is converted into WEKA's native Attribute-Relation File Format (ARFF), and the *normality* attribute is set as the target class. We have removed the cases with missing or malformed values to be consistent across attributes. We have also discarded the identifiers with no discriminative information in the data (e.g., unique source/destination identifiers). This is because it would cause the model to overfit the identifiers to the instances and not to general attacks. We have not used any resampling (oversampling or undersampling) to rebalance our classes, but in the original, naturally imbalanced form. The prepared dataset has been then compared with stratified *k-fold* cross-validation with each fold keeping the class proportions of the entire dataset.

4.4 Evaluation Metrics

To evaluate how well each of the six two-level ensemble setups performs, we use several metrics. These are all calculated from the confusion matrix - specifically the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

1. **Accuracy**: represents how often the model gets it right. It's simply the number of correct predictions divided by the total number of predictions as shown in Equation (2) [1]:

 ISSN:2302-8149
e-ISSN:2540-9719

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

2. **Precision**: the proportion of instances the model labels as positive that are actually positive. In other words, it shows how reliable the model's positive predictions are: The precision formula is shown in Equation (3) [1]:

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

3. **Recall**: the proportion of actual positive instances that the model correctly identifies. In other words, it shows how many of the true positives the model manages to catch. Equation (4) below is for calculating Recall [1]:

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

4. ***F1-score*:** a single score that brings precision and recall together into one balanced average. It leans toward whichever of the two is lower, so a model can't get a high *F1-score* by doing well on just one of them. That makes it a handy measure when catching real positives matters just as much as avoiding false alarms. The formula of calculating *F1-score* is presented in Equation (5) [1]:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (5)$$

5. **False Alarm Rate (*FAR*),** also called the **False Positive Rate (FPR)**: shows how often the model mistakes normal traffic for an attack. It's calculated by dividing the number of normal instances wrongly labeled as attacks by the total number of actual normal instances. Put simply, it tells us how often the system raises an alarm over legitimate traffic. The formula of calculating *FAR* is show in Equation (6) [4]:

$$FAR = \frac{FP}{FP+TN} \quad (6)$$

This metric is especially important for intrusion detection systems. A low False Alarm Rate means the model seldom misclassifies normal traffic as an attack. This is critical for real-world deployment, because a high *FAR* would flood analysts with false alerts and erode trust in the system.

5 Results and Discussion

The six outer-inner ensemble configurations have been evaluated using 3-fold stratified cross-validation on the DS2OS dataset. The results are examined below across eight figures, covering overall performance, per-class behavior, error patterns, and computational cost.

5.1 Overall Performance Comparison

The first metric to measure the performance of the proposed model is the overall performance. In Fig. 2, we compare the Accuracy, weighted *F1-score*, Kappa statistic and weighted False Alarm Rate (*FAR*) in all six configurations. Five of the six configurations showed near-identical and near-perfect performance. These configurations included every pairing that involved Stacking and MultiBoostAB (outer) + AdaBoostM1 (inner) pairing. Accuracy ranged from 99.9964% to 99.9997%, and Kappa ranged from 0.9993 to 0.9999. The sixth configuration, AdaBoostM1(outer)→MultiBoostAB(inner), stood apart with a comparatively lower accuracy of 99.11% and a Kappa of 0.837, along with a markedly higher *FAR* of 0.159 against a *FAR* of effectively 0.000 for the other five. Rather than indicating a genuinely weaker model, this gap is best explained by a difference in the underlying weak learner. The AdaBoostM1→MultiBoostAB configuration used *OneR*, which is restricted to splitting on a single attribute. The remaining five configurations used J48, which can exploit multiple attributes simultaneously.

 ISSN:2302-8149
e-ISSN:2540-9719

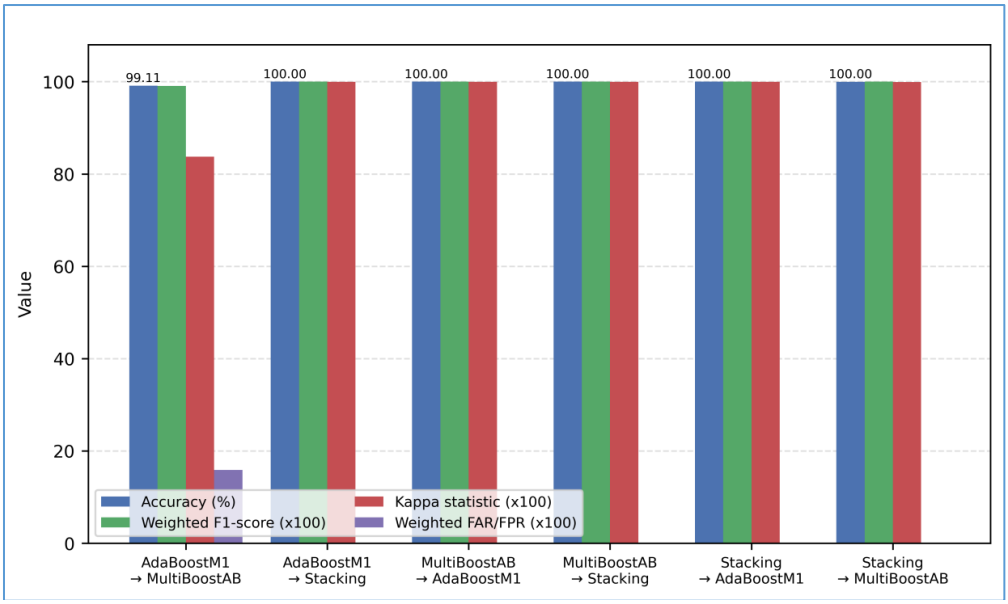

Figure 2 Overall performance comparison across the six outer→inner ensemble configurations

5.2 Per-Class *F1-Score*

The per-class *F1-Score* is another metric that has been measured for the proposed model. Fig. 3 presents a class-level view that a single aggregate metric cannot capture. Across the five J48-based configurations, every class (including the two rarest, *Data Type Probing* and *Wrong Setup* achieved an *F1-score* at or above 0.999). The *OneR*-based AdaBoostM1→MultiBoostAB configuration told a very different story: it failed to detect *Data Type Probing* and *Wrong Setup* entirely (*F1* = 0.000 for both), and showed only moderate discrimination for *Malicious Operation* (*F1* = 0.498) and *Malicious Control* (*F1* = 0.584), while still performing reasonably on *Scan* (0.824) and *Spying* (0.756). This is probably the most important figure in the whole diagnostic story. It shows that the near-perfect scores in Fig. 2 for the other five configurations aren't just an artifact of the majority "Normal" class pulling the average up. Those high scores hold up even for classes that make up a tiny fraction of a percent of the dataset. That kind of pattern points much more toward memorization at the attribute level than toward real behavioral generalization.

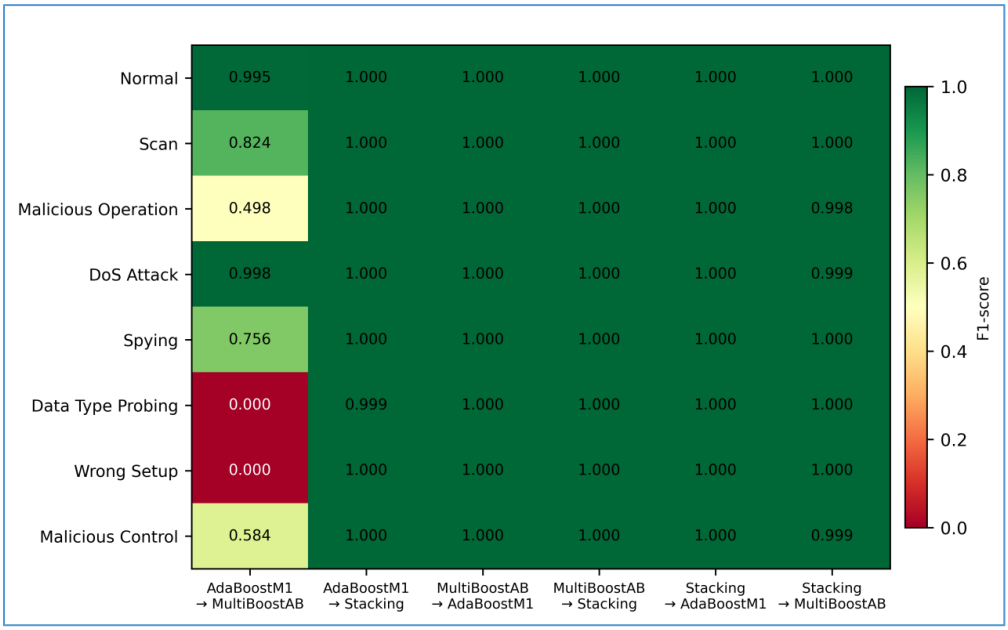

Figure 3 Per-Class *F1-score* across the six outer→inner ensemble configurations

 ISSN:2302-8149
e-ISSN:2540-9719

5.3 Per-Class Recall / Detection Rate

Fig. 4 looks at recall on its own, which is arguably the metric that matters most for an intrusion detection system in practice. It tells us how many real attacks were actually caught, rather than how well the model avoids false alarms. The results follow almost the same pattern as Fig. 3. The five J48-based configurations detect every attack class with virtually no exceptions, whereas the *OneR*-based configuration misses most of the *Data Type Probing* and *Wrong Setup* instances. For an IDS in real-world use, missing these two types of attack is arguably more serious than missing a *Scan*, because they often point to misconfiguration or malformed traffic that may come before a more serious attack. That recall for these two classes drops to zero under *OneR* is therefore an important finding in its own right.

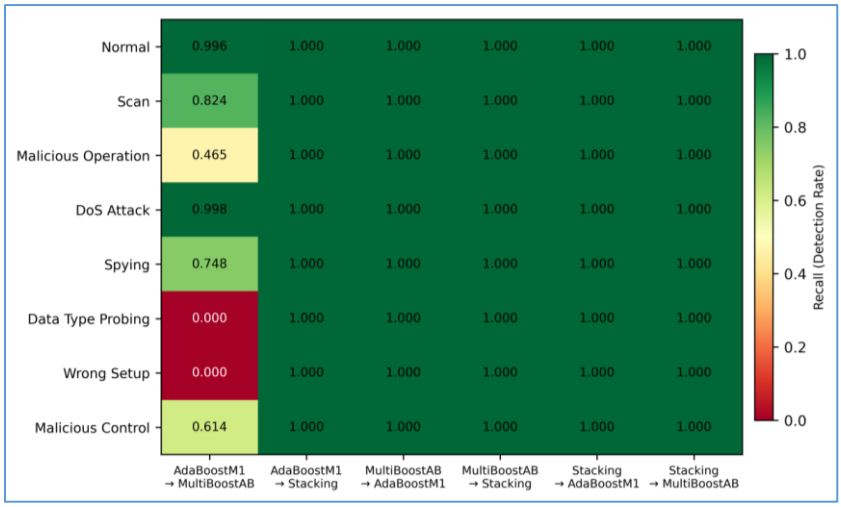

Figure 4 Per-class recall (detection rate) across the six configurations

5.4 Per-Class ROC Area

The ROC Area (AUC) per class is another metric measured in this study. Fig. 5 presents it and offers a threshold-independent look at separability. As in Figs. 3 and 4, the five J48-based configurations score at or extremely close to 1.0 for every class. The *OneR*-based configuration stands out with visibly lower separability for *Malicious Operation* and *Malicious Control*. Notably, these are the same two classes that showed the weakest *F1* and recall in the previous figures. The convergence of all three class-level metrics (*F1*, recall, AUC) on the same two problem classes strengthens the case that these classes are genuinely harder to separate using *timestamp* alone, rather than the poor scores being an artifact of a single metric's sensitivity to class imbalance.

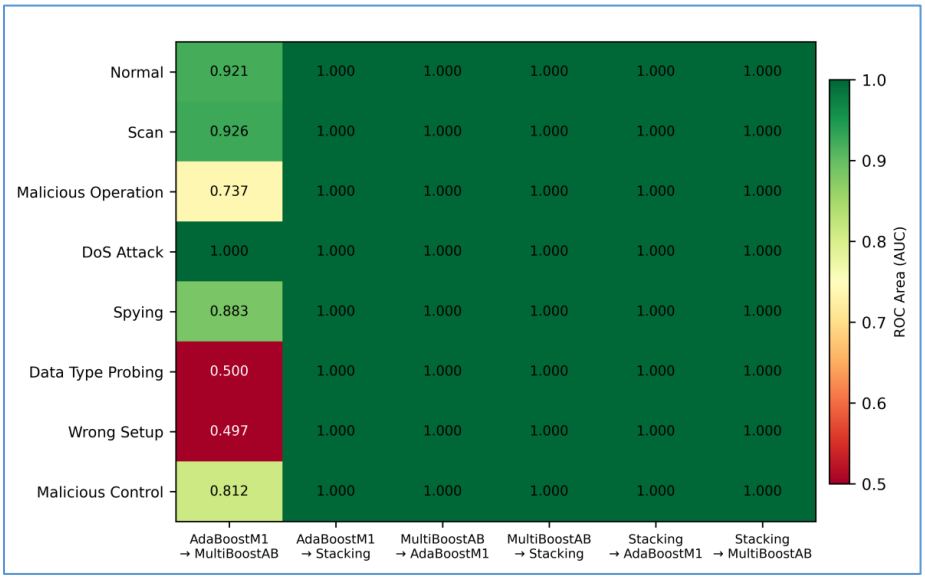

Figure 5 Per-Class ROC Area (AUC) across the six configurations

 ISSN:2302-8149
e-ISSN:2540-9719

5.5 Confusion Matrix Pairs: Outer/Inner Order Effects

The confusion matrices for the outer/inner configurations can reveal crucial information. Figs. 6a–6c examine each pair of algorithms in both outer/inner orderings, isolating the effect of nesting order from the effect of algorithm choice.

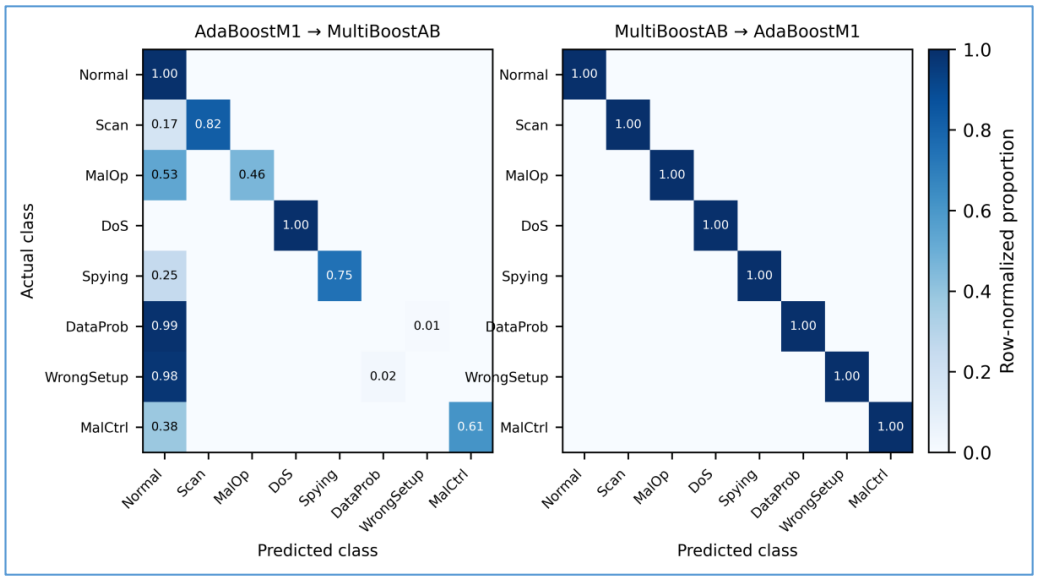

Figure 6a Confusion matrices (row-normalized): AdaBoostM1 ↔ MultiBoostAB configuration

Fig. 6a (AdaBoostM1 ↔ MultiBoostAB) is the only pair that shows any asymmetry: AdaBoostM1(outer)→MultiBoostAB(inner) exhibits the confusion pattern already discussed above, while reversing the order to MultiBoostAB(outer)→AdaBoostM1(inner) produces a perfect diagonal matrix. Figs. 6b (AdaBoostM1 ↔ Stacking) and 6c (MultiBoostAB ↔ Stacking), by contrast, show perfect diagonal matrices in both directions, with no visible difference between orderings. Taken together, these three figures indicate that outer/inner order alone does not materially affect performance once the underlying base classifier is held constant across both directions of a pair. The asymmetry visible in Fig. 6a is attributable to the change in base classifier (*OneR* versus J48) between that specific pair of runs, rather than to the nesting order of the two ensemble algorithms per se.

This is a useful negative result: it suggests that, for practitioners choosing between outer/inner arrangements of these three algorithms, the choice of underlying weak learner is likely to matter more than which ensemble sits on the outside.

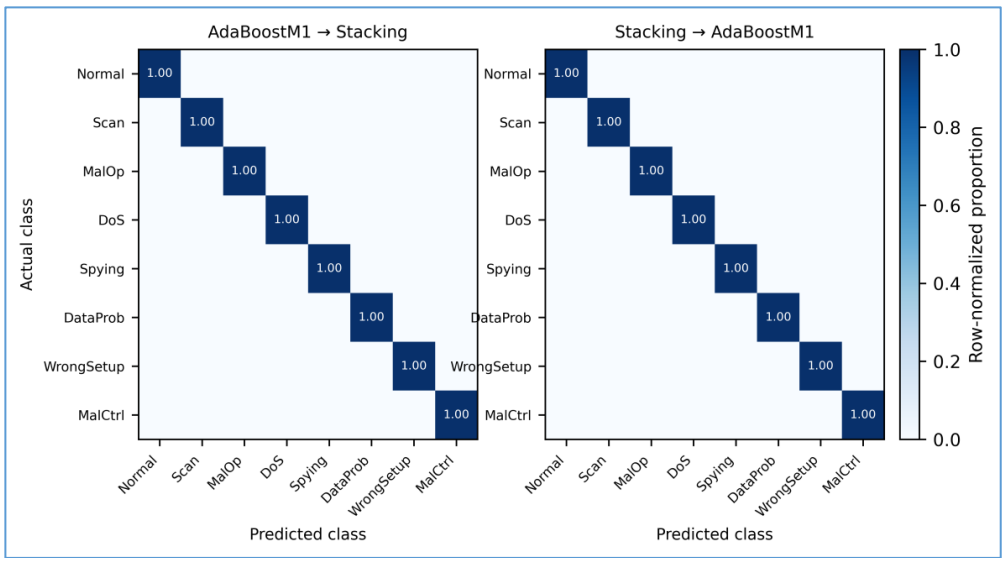

Figure 6b Confusion matrices (row-normalized): AdaBoostM1 ↔ stacking configuration

 ISSN:2302-8149
e-ISSN:2540-9719

5.6 Model Build Time

Another metric that can be used to compare the proposed configurations is the model building time. Fig. 7 compares computational cost across the six configurations. Build times ranged from 49.4 seconds (Stacking→AdaBoostM1) to 179.1 seconds (AdaBoostM1→Stacking), with the two configurations using Stacking as the outer ensemble consistently faster to train than those using AdaBoostM1 or MultiBoostAB as the outer ensemble. This is consistent with the structural properties discussed in Section 3. MultiBoostAB's periodic, wagging-based resampling reduces the strictly sequential dependency present in pure boosting. Stacking's single-pass combination of pre-trained base learners is inherently less iterative than repeatedly reweighting a training set.

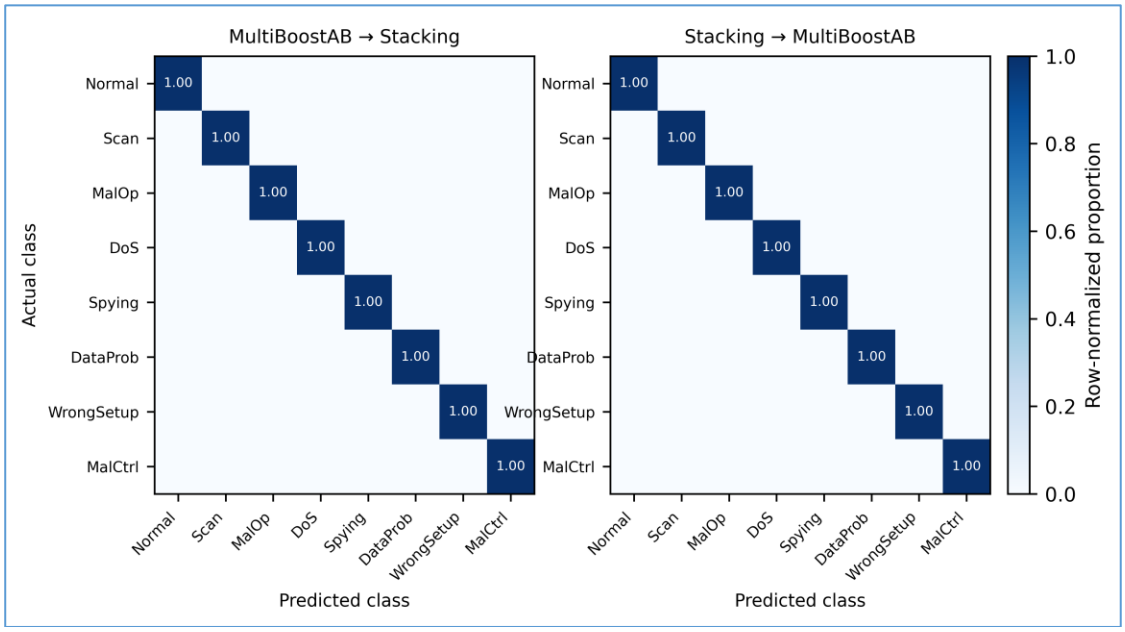

Figure 6c Confusion matrices (row-normalized): MultiBoostAB ↔ stacking configuration

While build time is a secondary consideration relative to detection accuracy, it is a practical factor for any real-world IoT deployment where model retraining may need to occur frequently as network conditions evolve.

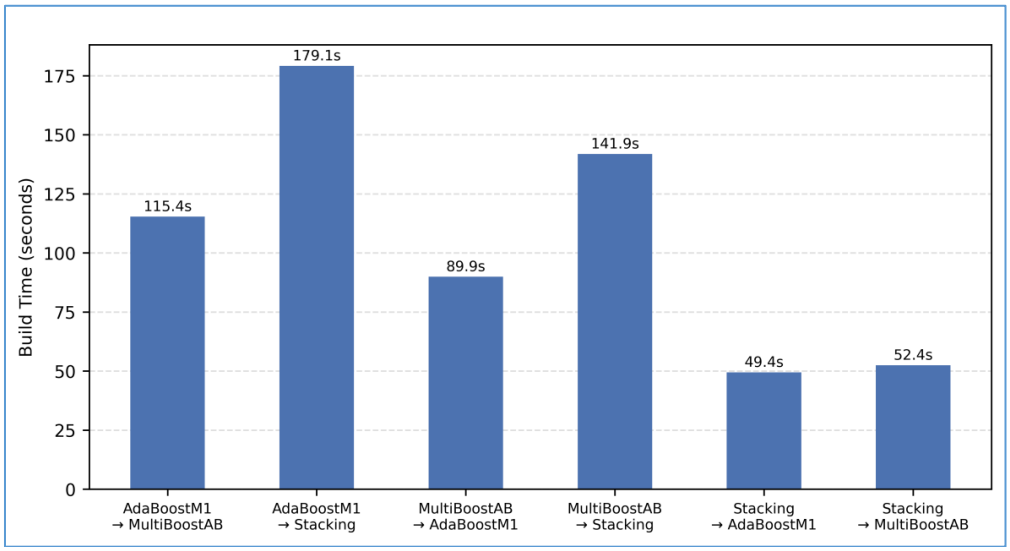

Figure 7 Model build time across the six outer→inner ensemble configurations

 ISSN:2302-8149
e-ISSN:2540-9719

6 Conclusion

This paper studied a hierarchical dependent ensemble approach to intrusion detection in IoT-based smart homes. AdaBoostM1, MultiBoostAB, and Stacking were nested in six outer–inner configurations and tested on the DS2OS dataset using stratified 3-fold cross-validation, with the natural class imbalance of the dataset left untouched. Five of the six configurations achieved near-perfect detection, with accuracy between 99.9964% and 99.9997%, Kappa values of at least 0.9993, *FAR* of effectively zero, and an *F1-score* of at least 0.999 for every class, including the rarest ones. The AdaBoostM1→MultiBoostAB configuration, which used *OneR* as its weak learner, reached 99.11% accuracy but missed *Data Type Probing* and *Wrong Setup* entirely and struggled with *Malicious Operation* and *Malicious Control*. Comparing each pair of algorithms in both orders showed no visible effect of nesting order on detection when the base classifier stayed the same. In terms of training cost, configurations with Stacking as the outer ensemble were the fastest, with build times as low as 49.4 s compared with 179.1 s for the slowest configuration. However, these results should be read with care. The near-perfect scores of the J48-based configurations are best treated as an upper bound, because J48 was able to split on device-identifier attributes that tie specific simulated devices to specific attack labels in DS2OS. The *OneR*-based configuration, although weaker, probably gives a more honest picture of how difficult the dataset really is.

References

[1] A. Kadam and P. S. Lokhande, "*Design of an Intrusion Detection Model for IoT-Enabled Smart Home*," *SSRN Electronic Journal*, 2024, DOI: 10.2139/ssrn.4833390.

[2] M. Amru *et al.*, "*Network Intrusion Detection System by Applying Ensemble Model for Smart Home*," *International Journal of Electrical and Computer Engineering*, Vol. 14, No. 3, pp. 3485–3494, 2024, DOI: 10.11591/ijece.v14i3.pp3485-3494.

[3] Y. Cao, Z. Wang, H. Ding, J. Zhang, and B. Li, "*An Intrusion Detection System based on Stacked Ensemble Learning for IoT Network*," *Computers and Electrical Engineering*, Vol. 110, Art. No. 108836, 2023, DOI: 10.1016/j.compeleceng.2023.108836.

[4] D. Rani, N. S. Gill, P. Gulia, and J. M. Chatterjee, "*An Ensemble-based Multiclass Classifier for Intrusion Detection using Internet of Things*," *Computational Intelligence and Neuroscience*, Vol. 2022, No. 1, Art. No. 1668676, 2022, DOI: 10.1155/2022/1668676.

[5] N. U. Sama, S. Ullah, S. M. A. Kazmi, and M. Mazzara, "*Cutting-Edge Intrusion Detection in IoT Networks: A Focus on Ensemble Models*," *IEEE Access*, Vol. 13, pp. 8375–8392, 2025, DOI: 10.1109/ACCESS.2024.3491831.

[6] D. V. Premalatha and S. Ramanujam, "*Ensemble-based Intrusion Detection for IoT Networks using the CICIoT2023 Dataset*," *Journal of Information Systems Engineering and Management*, Vol. 10, No. 21s, pp. 743–755, 2025, DOI: 10.52783/jisem.v10i21s.3407.

[7] R. Lazzarini, H. Tianfield, and V. Charissis, "*A Stacking Ensemble of Deep Learning Models for IoT Intrusion Detection*," *Knowledge-based Systems*, Vol. 279, Art. No. 110941, 2023, DOI: 10.1016/j.knosys.2023.110941.

[8] Z. Zheng, W. Gai, P. Zhang, and M. Zhou, "*Enhancing Feature Selection in IoT Intrusion Detection using the Ensemble Stacking*," in *2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2024, pp. 2151–2156, DOI: 10.1109/ISPA63168.2024.00293.

[9] M. Tawfik, "*Optimized Intrusion Detection in IoT and Fog Computing using Ensemble Learning and Advanced Feature Selection*," *PLoS ONE*, Vol. 19, No. 8, Art. No. e0304082, 2024, DOI: 10.1371/journal.pone.0304082.

[10] A. Aldaej, I. Ullah, T. A. Ahanger, and M. Atiquzzaman, "*Ensemble Technique of Intrusion Detection for IoT-Edge Platform*," *Scientific Reports*, Vol. 14, No. 1, Art. No. 11703, 2024, DOI: 10.1038/s41598-024-62435-y.

ISSN:2302-8149
e-ISSN:2540-9719

[11] Y. Freund and R. E. Schapire, "*Experiments with a New Boosting Algorithm*," in *Proceedings of the 13th International Conference on Machine Learning (ICML)*, Bari, Italy, 1996, pp. 148–156, DOI: 10.5555/3091696.3091715.

[12] G. I. Webb, "*MultiBoosting: A Technique for Combining Boosting and Wagg*ing," *Machine Learning*, Vol. 40, No. 2, pp. 159–196, 2000, DOI: 10.1023/A:1007659514849.

[13] D. H. Wolpert, "Stacked generalization," *Neural Networks*, Vol. 5, No. 2, pp. 241–259, 1992, DOI: 10.1016/S0893-6080(05)80023-1.

[14] S. H. Rafique, A. Abdallah, N. S. Musa, and T. Murugan, "*Machine Learning and Deep Learning Techniques for Internet of Things Network Anomaly Detection-Current Research Trends*," *Sensors*, Vol. 24, No. 6, Art. No. 1968, 2024, DOI: 10.3390/s24061968.

[15] S. Cheng and X. Feng, "*Ensemble-based Detection of Distributed Denial-of-Service Attacks in IoT Networks using Majority Decision Mechanisms*," *Scientific Reports*, Vol. 16, Art. No. 10886, 2026, DOI: 10.1038/s41598-025-23638-z.