ENHANCING IOT CYBERATTACK DETECTION VIA HYPERPARAMETER OPTIMIZATION TECHNIQUES IN ENSEMBLE MODELS

Otshepeng Kgote [1], Bassey Isong [1] and Tsapang Mashego [2]

[1] Department of Computer Science, North-West University, Mafikeng, South Africa
[2] Department of Statistical Sciences, University of Cape Town, Rondebosch, South Africa

ABSTRACT

In the face of rapidly growing security threats on the Internet of Things (IoT) networks, machine learning (ML) integration shows promise in identifying cyberattacks. However, while the traditional ML models are effective in certain areas, they often fail to detect complex patterns and unusual behaviour in IoT data due to their difficulty adapting or generalizing. Ensemble learning models utilize the strengths of multiple base models to provide a promising solution but are largely influenced by the choice and proper setting of hyperparameters. This paper explores the impact of hyperparameter tuning on ensemble-based ML models for detecting IoT-related cyberattacks. We conducted a series of experiments utilizing the imbalanced and balanced CICToNIoT datasets, with a focus on binary and multi-class classification. The study assessed the Random Forest (RF) and Extreme Gradient Boosting (XGBoost) models with default hyperparameters after applying three tuning methods: Bayesian Optimization with Tree-structured Parzen Estimators, Grid Search (HGS), and Random Search. Our results reveal that HGS significantly enhanced performance, with XGBoost achieving an accuracy of 99.34% and an F1-score of 99.34% in binary classification, and RF achieving an accuracy of 93.76% and an F1-score of 93.73% in multi-class classification. RF demonstrated strong detection capabilities across various attack types, though it struggled with distinguishing certain attacks. These findings highlight the importance of hyperparameter tuning in enhancing the effectiveness of ML models for IoT cybersecurity.

KEYWORDS

IoT Security, Intrusion Detection, Ensemble Learning, Hyperparameter Tuning.

1. INTRODUCTION

In recent years, the Internet of Things (IoT) has been transforming industries such as healthcare, smart cities, agriculture, and manufacturing. With billions of connected devices, it boosts efficiency, enhances productivity, and enables smarter decision-making [1, 2]. However, the proliferation of these smart devices and sensors connected to the internet has led to significant security challenges [1-3]. Several IoT devices have limited resources, lack strong built-in security, and operate in diverse environments, making them vulnerable to cyberattacks [2, 3]. In most cases, these vulnerabilities result in data leaks, unauthorized access, and even large-scale network disruptions [1-3]. As IoT networks expand exponentially, effective intrusion detection and prevention systems are critical to protect sensitive data and maintain system reliability [3, 4]. This is crucial because traditional security measures, such as firewalls and encryption, are often inadequate to address the dynamic and evolving threats in IoT ecosystems [2].

Intrusion detection systems (IDS) have become a vital component of IoT security, helping to monitor network traffic and identify malicious activities [1, 5]. However, IoT environments present unique challenges for developing IDS due to factors such as high data volume, device heterogeneity, and real-time processing demands [5]. To address these challenges, advanced techniques such as machine learning (ML), deep learning (DL), blockchain, etc., have been integrated into IoT systems to enhance security and adapt to evolving threats [1, 3, 4]. This is evident in several studies [4-7], showing that ML techniques such as ensemble learning models that merge multiple algorithms have gained prominence in IoT security due to their improved detection accuracy and robustness [8]. While traditional ML models are effective in certain areas, they often struggle to handle the complexity of intrusion detection in IoT environments [4, 6]. These models often fail to detect complex patterns and unusual behaviour in IoT data because they struggle to adapt or generalize effectively [5]. The constantly evolving IoT environments, with new attack types and a wide range of devices used in various ways, make intrusion detection even more difficult [5-7]. Therefore, ensemble learning models, which leverage the strengths of multiple base models, offer a promising solution to these challenges.

Despite their potential, the performance of ensemble learning models heavily relies on the proper setting of hyperparameters, which guide how the models learn and behave [5, 9]. Hyperparameter tuning plays a vital role in ML. Without properly adjusting the parameters, the models struggle to learn from the data or make accurate predictions, hindering their ability to generalize effectively [7, 9]. Currently, the two approaches for performing hyperparameter fine-tuning or hyperparameter optimization (HPO) are: traditional or manual and automated tunings [5, 10]. While traditionally fine-tuning the hyperparameters of the models is time-consuming and subjective, automating the HPO process is considered a viable alternative approach [5, 7, 10]. This is because it both saves time and reduces human intervention [5, 7, 9, 10]. In addition, it can be achieved through black-box optimization and multi-fidelity optimization (MF-HPO) approaches, which include different techniques such as hyperband (HB), grid-search (GS), random search (RS), Bayesian optimization (BO), etc. [7, 10].

To maximize and enhance the performance of the ML-based intrusion detection schemes, HPO ought to be studied [5, 7, 9]. In the context of ensemble learning, several HPO techniques exist to improve their detection accuracy and robustness [7, 10]. However, deciding which technique to use in which application in the IoT systems is challenging. Thus, this paper empirically analyses the effectiveness of various black-box HPO techniques on the performance of tree-based ensemble classifiers for detecting malicious attacks in IoT networks. The main contributions of this paper include:

1) Proposes a novel approach to hyperparameter tuning specifically designed for ensemble learning models in IoT intrusion detection based on the CICTONIoT dataset [11] and addresses high dimensionality and imbalanced class distributions using the Synthetic Minority Over-Sampling Technique (SMOTE).
2) Evaluation of tree-based ensemble techniques for binary and multi-classification. This illustrates how optimized ensemble models can significantly improve intrusion detection accuracy to surpass traditional ML methods.
3) Provides practical guidance on the selection and implementation of hyperparameter tuning techniques such as halving grid search (HGS), RS, and BO techniques. This provides a valuable resource for researchers and practitioners working on IoT security.

The remaining parts of this paper are organized as follows: Section 2 provides a literature review on IoT security, ensemble learning methods, and HPO techniques. Section 3 outlines the study methodology, Section 4 discusses the experimental setup, Section 5 presents the results, Section 6 discusses the findings and comparisons, and Section 7 concludes the paper.

2. Literature Review

This section presents an overview of IoT architecture, intrusion detection in IoT, ensemble learning methods, and hyperparameter tuning methods.

2.1. IoT Architecture

IoT is a network of various smart devices connected to the Internet for communication and information sharing [12, 13]. These devices range from everyday electronics to sensors and actuators, all connected through a gateway that links them to the cloud [12, 13]. The IoT system's architecture is typically composed of various layers, which work together to collect, process, share and protect data.

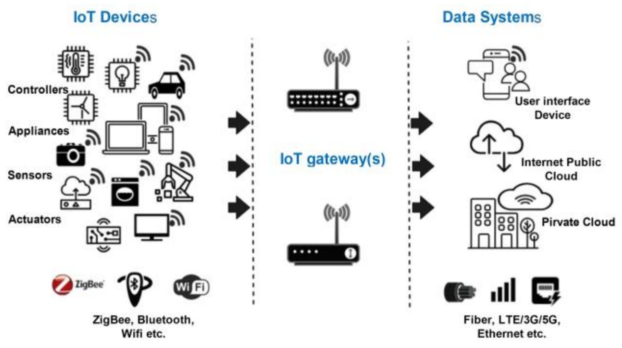


Fig.1.IoT gateway connectivity and communication

As shown in Fig. 1, the IoT gateway facilitates communication between the sensor network and the Internet and performs tasks such as protocol translation, data aggregation, filtering, local storage, and device control to improve security [13]. The connection between devices is achieved through various technologies, such as 3G, RFID, Wi-Fi, Bluetooth, ZigBee, Z-wave, power line communication, fibre optics, and ethernet [12, 14]. While the middleware stores, processes, and analyzes the massive amounts of data collected from IoT devices, the security plane, which spans across all layers, protects the IoT system and its data from cyber threats. In addition, business models, policies, and decisions that drive the IoT system are managed by the business layer. With the improved monitoring, response, and analytics offered by the application layer, IoT has been implemented across many fields such as healthcare, smart cities, smart homes, agriculture, military, transportation, industry, and smart grids [12-15]. This is evidence that IoT is transforming various industries by increasing efficiency and services.

2.2. Intrusion Detection in IoT

IoT has introduced a paradigm shift in how devices interact and communicate, yet it has also opened the door to a multitude of security threats. IoT systems are especially susceptible to various attacks such as denial of service (DoS) and distributed denial of service (DDoS), malware infections, data breaches, and unauthorized access [1, 2]. This vulnerability stems from their decentralized nature, limited resources, lack of consistent security standards, and so on. These threats are further exacerbated by the sheer scale of IoT networks, which can consist of thousands or even millions of interconnected devices which making it more challenging to secure them effectively [5, 12, 15, 16]. As a result, IDSs have become essential for identifying and detecting malicious activities within IoT environments by helping to safeguard against potential threats and vulnerabilities [5, 7, 12]. This is crucial for protecting the integrity and security of connected devices and networks. However, when developing a security strategy, it is important to consider the limitations and challenges, such as constraints on device resources and the potential for data overload [1, 3, 7].

As IoT systems grow increasingly complex,implementing an effective IDS that integrates various techniques such as anomaly detection, ML, and network monitoring has become a top priority. This approach helps to safeguard against both known and emerging threats, which ensures a more robust defence [1, 3]. Existing approaches to intrusion detection in IoT include signature-based, anomaly-based methods, and hybrid-based methods [1, 7, 17, 18]. The signature-based IDS detectsknown threats by comparing traffic to predefined patterns but is ineffective against novel or evolving attacks [7, 19]. The anomaly-based IDS uses techniques such as ML to identify deviations from normal behaviour, making it effective for previously unseen attacks [1, 18, 19]. However, it often faces challenges such as high false positive rates (FPR) and they also require large volumes of labelled data for effective training, which can be cumbersome to obtain in many IoT environments [19-21]. Recent advances in ML, especially ensemble learning, have helped overcome these challenges. Combining the strengths of multiple models boosts detection accuracy and makes the system more reliable [8, 17, 22, 23]. The hybrid-based method combines both approaches to offer a more comprehensive defence against a wide range of threats.

2.3. Ensemble Learning Methods

Ensemble learning is an ML paradigm that utilizes the predictions of multiple base models to create a more accurate and stable final prediction [23-25]. It is based on the idea that a group of weaker models can work together to create a stronger model [23, 25]. Consequently, the approach helps to reduce overfitting and makes the model better at generalizing to new data [25]. The three primary types of ensemble learning techniques include bagging (e.g, Random Forest (RF)), boosting (e.g, AdaBoost and Gradient boosting (GBoost), etc,) and stacking [23, 24]. In cybersecurity, especially within IoT ecosystems, ensemble learning has become increasingly popular [25]. It is valued for its ability to manage complex ML issues, such as missing features, imbalanced datasets, etc., which are often encountered in intrusion detection tasks [20, 22, 24]. Ensemble models have been effectively used to detect network intrusions, phishing, and malware attacks [1, 6, 24]. Their ability to combine different models and capture various perspectives of the data makes ensemble models especially useful for identifying complex and multi-faceted cyber threats [1, 6, 22]. This approach allows them to better handle sophisticated attacks.

However, their performance depends a lot on how well their hyperparameters are configured [9]. Getting the right setting is critical to ensuring effectiveness and best performance. Moreover, optimizing these models for IoT intrusion detection is challenging due to the high-dimensional, imbalanced dataset that can lead to overfitting. Furthermore, their complexity and the limited resources of IoT devices require effective optimization methods [5, 7, 9]. The lack of standardized benchmarks for IoT intrusion detection further complicates mode development and comparison [5, 7, 9]. These challenges make careful hyperparameter tuning important for improving performance in IoT environments.

2.4. Hyperparameter Tuning

Hyperparameters are settings that control the model’s learning process, such as the learning rate, number of layers, number of epochs, or number of trees [10, 26-29]. Unlike model parameters, which are learned from the data, hyperparameters are set before training and significantly impact the model’s performance [26, 29]. Therefore, selecting the right hyperparameters is key to achieving high accuracy, preventing overfitting, and ensuring the model can generalize well to new data. Hyperparameter tuning or HPO involves selecting the best hyperparameters for the ML model to improve its performance on unseen data [26, 27]. The HPO process includes several components: an estimator with its respective objective function, a search space for possible hyperparameter values, an optimization method to find the best combinations, and an evaluation function to assess the performance of different hyperparameter setups [10, 26]. Many techniques for hyperparameter tuning have been developed, as shown in Fig. 2, each with its strengths and limitations.

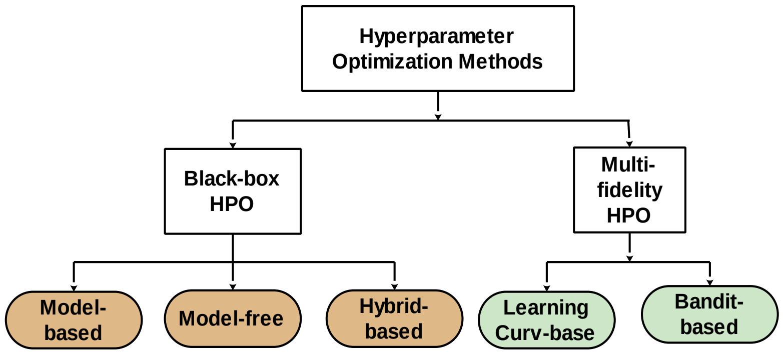


Fig.2. Categories of HPO methods

The two main categories of HPO methods are black-box optimization and MF-HPO [26, 30]. Blackbox HPO (BHPO) methods focus on finding the optimal inputs to yield optimal output, without requiring the knowledge or modelling of the internal working or mechanics of the model to be optimized [26]. Moreover, Blackbox HPO methods can be classified into three categories, namely: model-based methods, model-free methods, and advanced hybrid methods [30-32]. Fig. 2 presents the categories of HPO methods. Model-based HPO methods construct probabilistic surrogate models to predict or approximate the objective function and guide the search process for the selection of the hyperparameters [30]. Its techniques include Bayesian optimization with tree-structured parzen estimators (BO-TPE), BO with Gaussian process (BO-GP), and neural networks with surrogate models (NN-SM) [28, 30]. In contrast, model-free HPO methods utilize the hyperparameter space through direct evaluation, without relying on the objective function [10]. Its techniques include GS, RS, genetic algorithms (GA), particle swarm evolutionary strategies (PSES), and more [10, 30].

Similarly, hybrid HPO methods combine the strengths of both model-based and model-free techniques by using elements from each approach to improve performance [30, 33]. Some of the techniques include HGS, halving randomised search (HRS), a genetic algorithm with surrogate models (GA-SM), BO with evolutionary algorithms (BO-EA), and reinforcement learning with evolutionary algorithms (RL-EA) [26, 30, 31]. Furthermore, MF-HPO methods aim to find the best hyperparameter settings by using different levels of accuracy in function evaluations to guide the optimization process [26]. They focus on using cheaper, lower-accuracy evaluations to explore the hyperparameter space first, then switching to more expensive, higher-accuracy evaluations later [26, 30]. This helps speed up the tuning process. As shown in Fig. 2, some methods treat HPO as a multi-armed bandit problem, like hyperband, successive halving, and others. Other methods, such as freeze-thaw BO (FTBO) and learning curve extrapolation (LCE), use model learning curves to predict final performance [26, 30].

This paper will empirically evaluate HPO methods such as HGS, RS, and BO-TPE to improve the performance of ensemble learning models in IoT ecosystems. While these methods have been widely used in other domains, they have yet to be fully explored for IoT-based IDSs. Despite growing interest in ensemble learning and hyperparameter tuning, IoT-based IDS research still faces significant gaps and is often overlooked [34]. Many studies rely on default or generic methods that do not address the unique challenges of IoT, such as high dimensionality, imbalanced classes, and real-time processing needs, which often result in suboptimal model performance [17, 34]. In addition, existing evaluation metrics such as accuracy and precision may not fully capture performance, especially when false positives or negatives can have serious consequences [17]. Thus, to improve IDS research and security in IoT, more tailored evaluation frameworks are needed.

2.5. Related Works

This section examines related studies on ML-based intrusion detection, focusing on methods such as DL, ensemble learning, and hyperparameter tuning within IoT environments. Table 1 presents a summary of these studies. Bakır and Ceviz [7] compared several ensemble and tree-based techniques for detecting attacks. After using feature selection, XGBoost was optimized with a GA, leading to improved accuracy and computation time. However, they did not report the FPR, which is essential for assessing the model's performance.

Table 1. Summary of related works

Study	Techniques Used	Optimization Method	Attacks	Key Findings	Limitations
[7]	Ensemble, Tree-based (XGBoost)	GA	Various attacks	Significant improvement in accuracy and computational time.	No false positive rates reported.
[35]	Various ML classifiers (RF, DT)	GS	DDoS attacks	RF and DT outperformed other models after optimization.	Limited to DDoS attack detection.
[9]	DL, ML techniques (RF)	RS	DDoS attacks	RF showed 98.78% detection rate, best performance.	Only DDoS attack detection and binary classification.
[36]	FL-XGBoost	BO	Spoofing attack	The solution achieved an accuracy of 96.0%.	Limited only to spoofing attacks, and no false positive rate was reported.
[5]	DL	GS, RS	Various Attacks	Improved performance compared to existing models.	Accuracy could be further improved.
[37]	Traditional Supervised (PCA, KBest)	GS	-	Better results in binary classification.	Limited to binary classification and no false positive rate reported.
[21]	Traditional, Tree-based ensemble (GBoost, XGBoost)	BO	Various Attacks	GBoost and XGBoost had the best accuracy and recall.	No detailed evaluation results.
[38]	DTs	BO	Botnet attacks	Improved performance compared to existing solutions.	Limited to DDoS detection and binary classification.
[39]	Traditional ML and EnsembleXGBoost,	RS	DDoS attacks	XGBoost achieved better results with accuracy of 97%.	Limited to DDoS detection and binary classification.

Similarly, Sanchez et al. [35] focused on detecting DDoS attacks with different ML classifiers. They used GS for hyperparameter tuning and tested various datasets. Although their work is limited to one type of attack, they found that RF and DT models performed better after optimization. In another study, Gaur et al. [9] suggested a combination of DL and traditional ML for DDoS detection. By utilizing an RS for optimization, they discovered that RF surpassed other models with a detection rate of 98.78%. However, their work only focuses on DDoS attacks, limiting its broader application.

Furthermore, Guemebe et al. [36] employed federated BO and XGBoost techniques for predicting spoofing attacks in the IoMT domain, where the solution achieved a prediction accuracy of 96%. However, the work is limited to binary classification. In parallel, Kunang et al. [5] introduced a DL-based IDS for IoT attacks. Using GS and RS to optimize the models, their solution demonstrated improved results over existing models and reported higher overall accuracy. Mohy-Eddine et al. [37] also suggested a traditional supervised technique for detecting intrusions in IoT and utilized Principal Component Analysis (PCA) and KBest for feature reduction and grid search for optimization. Although their solution was unsuccessful in binary classification, it is limited to this area. In the same vein, Lai et al. [21] compared traditional tree-based and ensemble techniques for IoT attack detection and optimized the models with BO. Their results demonstrated that GBoost and XGBoost were superior in terms of accuracy and recall, but they did not provide detailed evaluation results. Injadat et al. [38] used Decision Trees (DT) for botnet detection in IoT which is optimized via BO. Their solution was not effective in detecting DDoS attacks but was limited to detecting DDoS attacks. On the same note, Jiyad et al. [39] proposed an ensemble ML method that utilizes explainable AI (XAI) to predict DDoS attacks with the solution incorporating SHAP and LIME for interpretability. With a 97% accuracy rate, the study showed that XGBoost-based models performed better than classifiers.

The studies summarized in Table 1 highlight a clear trend of using ML for attack detection, especially DDoS. These studies focus on optimizing models utilizing hyperparameter tuning methods like GS, RS, BOs, and GAs. The main goals are to improve detection accuracy and computational efficiency by often incorporating feature selection or dimensionality reduction. However, several studies that implemented GS noted its time-consuming nature and challenges with dimensionality [10, 29]. Several studies focus on just a single attack or binary classification which limits their generalizability. Additionally, important evaluation metrics, such as the FPR, are often overlooked, and some models still need accuracy improvement. There is also a lack of multi-class or broader attack detection, which indicates areas for further research. In this paper, we aim to evaluate alternative black box HPO techniques in enhancing ensemble learning models for IoT IDS. Unlike most studies, we assess our proposed solutions via binary classification and multi-classification to demonstrate the performance against various attacks.

3. METHODOLOGY

This section outlines the methodology used in our experiment, including the steps taken to work with the dataset. We detail the data pre-processing techniques applied, the tree-based models selected, and the HPO methods employed to fine-tune the models. In addition, we describe the performance metrics used to assess the performance of the models and the overall process is shown in Fig.3. In the experiment, we worked with both imbalanced and balanced datasets, splitting the data into 80% for training and 20% for testing. We conducted four experiments: First, we trained and tested the RF and XGBoost models using their default settings. Then we fine-tuned the models using BO with Tree-structured Parzen Estimators (BO-TPE) in the second experiment, followed by HGS in the third experiment. In the fourth experiment, the RS is used for optimizing the models. The results from these fine-tuned experiments were compared with the results from the default baseline, with further details on the evaluation parameters provided in Section 5.

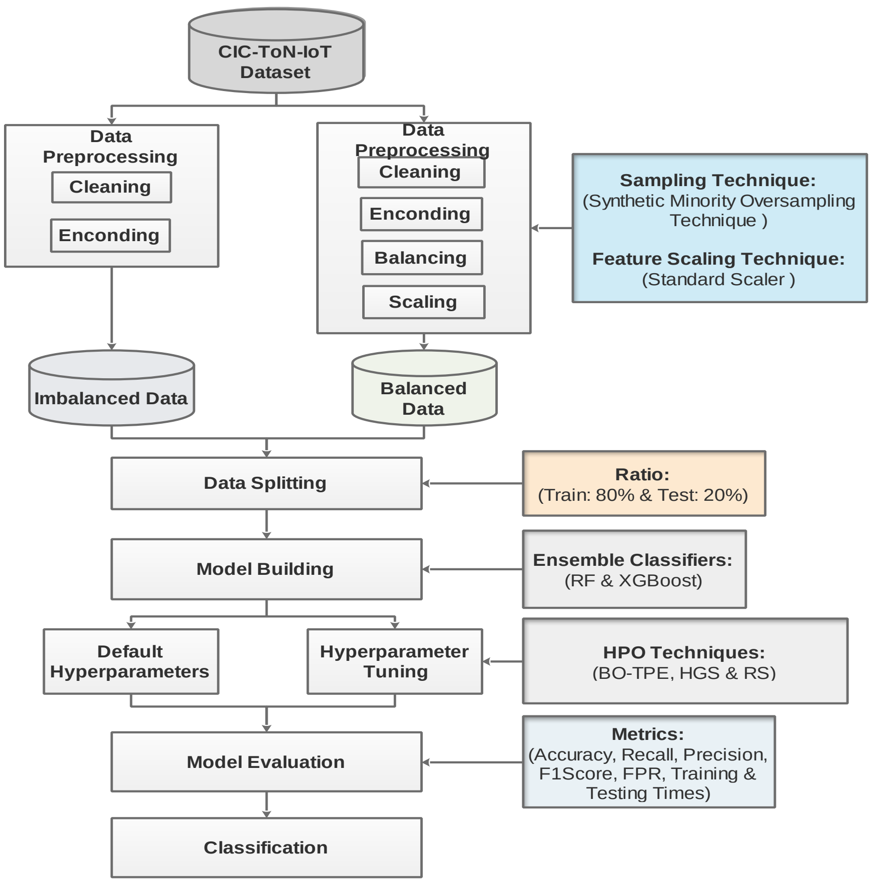

CIC-ToN-IoT Dataset
Data Preprocessing
Cleaning
Enconding
Data Preprocessing
Cleaning
Enconding
Balancing
Scaling
Sampling Technique: (Synthetic Minority Oversampling Technique)
Feature Scaling Technique: (Standard Scaler)
Imbalanced Data
Balanced Data
Data Splitting
Ratio: (Train: 80% & Test: 20%)
Model Building
Ensemble Classifiers: (RF & XGBoost)
Default Hyperparameters
Hyperparameter Tuning
HPO Techniques: (BO-TPE, HGS & RS)
Model Evaluation
Metrics: (Accuracy, Recall, Precision, F1Score, FPR, Training & Testing Times)
Classification

Fig.3. Proposed Approach workflow

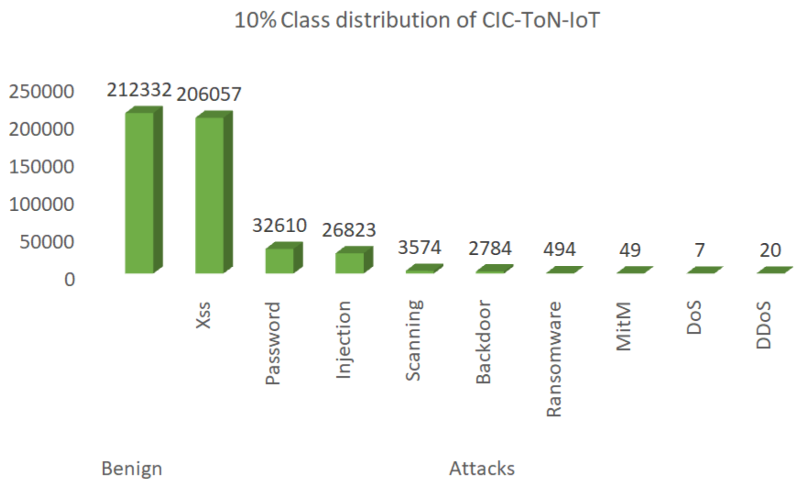


Fig.4. Class distribution of benign and attacks

3.1. Data Collection

The dataset used in this study is the CIC-ToN-IoT dataset released in 2019, which is an alternative version of the UNSW-ToN-IoT dataset. The TON-IoT dataset contains various telemetry data from IoT sensors, IoT network traffic, and operating system logs, all generated using an industrial network testbed [40]. The inventors employed the CICFlowMeter version 4 tool to extract features from the TONIoT dataset, which resulted in 83 features and 5,351,760 data samples [11]. This dataset includes both normal traffic flow and various cyber-attack events such as DoS, DDoS, injection, backdoor, Man-In-The-Middle (MitM), scanning, ransomware, password attacks, and Cross-Site Scripting (XSS).In the context of this paper, due to the computational system limitations of the machine used for the experimentation, we could not use the full dataset, thus we utilized only 10% of the CICToNIoT dataset, which amounted to 484,750 data samples, as shown in Fig. 4. To sample the dataset we set the fraction to 0.10 and the random state to 42, which enabled us to randomly select 10% of the entire dataset and store it to a new CSV file.

3.2. Data Pre-processing

To prepare the dataset for analysis, several pre-processing steps are applied, as we worked with both imbalanced and balanced datasets. First, handle missing values by removing the affected entries with missing values and extra whitespaces. Consequently, irrelevant columns were also dropped, leaving us with 79 useful features. Then we also handled categorical data by applying the label encoding technique to transform categorical features into numerical values (usually integers) for both binary (Bening = 0, attack = 1) and multi-class classifications. Furthermore, we applied SMOTE to address the imbalance in the data [11]. SMOTE resamples the data by oversampling the minority class and undersampling the majority class, leading to a more balanced dataset for training. As depicted in Fig. 4, for binary classes, the prior class distribution was 212332 and 272418 samples for benign and attacks, respectively. After applying SMOTE, the minority class was synthetically oversampled to balance the distribution, resulting in 272418 samples for both benign and attack classes, respectively.

Similarly, for multi classes, the benign class had 212332 samples, followed by xss with 206057, password with 32610, injection with 26823, scanning with 3574, backdoor with 2784, ransomware with 494, MITM with 49, DDos with 20, lastly Dos class with 7 samples. After applying SMOTE, each minority class was synthetically oversampled to match the 212332 samples of the majority class, thus, resulting in equal class distribution across all classes. Finally, we applied feature scaling or standardization to make sure that all features were of equal importance during model training. In this case, we standardized the features with the Standard Scaler method to ensure that they all have comparable scales for more effective modeling. Additionally, the feature selection method was used to reduce the number of features and eliminate those that are redundant or irrelevant. These highlighted measures are essential for enhancing model performance and reducing computational complexity.

3.3. Selected Ensemble Learning Models

This subsection discusses the two widely known tree-based ensemble learning models, based on bagging and boosting, adopted for predicting attacks.

3.3.1. Random Forest

This is a bagging ensemble learning model that combines multiple DTs to make predictions [8]. Each tree in the forest provides a classification, and the algorithm works in two stages: first, it creates the classifiers, DTs, and then it makes predictions by taking the majority vote from all the trees. This process helps reduce variance and prevent overfitting to the training data [23, 41]. The final prediction is the one on which most of the trees agree. As shown in equation (1), the model represents the most common prediction from all the trees, and B is the total number of DTs in the forest [41]. $\mathrm{T_i}(\mathrm{x})$ is a prediction of DTs for a given input x and the final prediction $\hat{y}$ for the input x,is obtained by majority voting:

$$\hat{y} = mode([T_1(x), T_2(x), \dots, T_B(x)]) \tag{1}$$

3.3.2. Extreme Gradient Boosting

This is an ensemble learning model that also uses DTs and operates within the GBoosting machine architecture [42]. The technique utilizes boosting, which combines the predictions of multiple poor learners to create a strong learner through additive training strategies [41, 42]. It helps reduce overfitting by simplifying the target functions and improves computational efficiency by using parallel calculations [7, 42]. The final prediction is achieved using the sigmoid function to summarize all the DTs' predictions. Equations (2) to (3) represent the formula of the XGBoost classifier.

$$\hat{y}_i = \frac{1}{1+e^{-F(x_i)}} \tag{2}$$

Where $F(x_i)$ is the model's output for the *i*th instance given by:

$$F(x_i) = \sum_{t=1}^{T} f_t\,(x_\mathrm{i}) \tag{3}$$

Where $f_t(x_i)$ is the prediction of the t^{th} tree for the *i*th instance.

In essence, RF is chosen for its strength and the ability to scale while handling complex data. Also, creating multiple DTs and combining their predictions helps reduce overfitting and improve generalization [41]. Likewise, XGBoost, a faster and more efficient version of GBoosting, is selected for its speed and top-tier performance in ML competitions [42]. These models were selected for their effectiveness in classification tasks, particularly in cybersecurity. They handle imbalanced datasets, capture complex patterns, and offer interpretable results, making them suitable for IoT intrusion detection [8, 23, 41, 42]. The wide range of adjustable hyperparameters also allows fine-tuning to optimize performance, aligning with the objectives of this study.

3.4. Hyperparameter Tuning Methods

This subsection discusses the different black-box HPO methods selected for this study to optimize the ensemble learning models considered in this paper.

3.4.1. Halving Grid Search

This is an HPO technique that improves the traditional GS by adding a strategy called successive halving [27, 43]. In this method, instead of testing every possible hyperparameter combination, it begins with a broad range and gradually eliminates more promising options [27, 43]. This results in minimizing both computational costs and time, without sacrificing model performance, making it suitable for large search spaces or costly-to-train models [27].

3.4.2. Bayesian Optimization-TPE

This is an effective model-based HPO technique that uses a probabilistic model to divide the hyperparameter space into promising and less promising regions [26, 44]. By constructing separate models for configurations that are likely to improve performance versus those that are not, BO-TPE provides computational resources where they will have the most impact [26, 44]. This combination of exploration and exploitation makes searches more efficient than traditional methods.

3.4.3. Random Search

This is one of the most widely used model-free techniques that provides a practical alternative to GS by randomly sampling hyperparameter combinations from a predefined space [10, 26, 44]. Instead of testing every option systematically, it evaluates a subset of configurations, resulting in faster insights and results [10, 43, 44].

These techniques were selected because HGS efficiently narrows the search space and is scalablewhile RS is simple, fast, and effective for large or uncertain search spaces. Similarly, BO-TPE is highly efficient in finding optimal solutions by intelligently exploring the hyperparameter space and is ideal for complex, costly-to-train models.

3.5. Evaluation Metrics

The performance of the models in this experiment is evaluated using a variety of metrics to ensure a thorough assessment [1, 45]: Accuracy measures how often the model accurately predicts attacks and benign traffic, while precision and recall focus on the model's ability to accurately identify attacks and minimize false positives and false negatives, respectively. The F1score balances precision and recall, providing a useful metric for imbalanced datasets. The FPR highlights how often normal traffic is misclassified as an attack. Conversely, the confusion matrix provides a summary of both correct and incorrect predictions. It helps to derive key values such as true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Moreover, the training and testing times are tracked to measure the model's efficiency.

$$Accuracy = \frac{TP+TN}{TP + TN + FP + FN} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

$$F1\text{-}score = \frac{2 \times precision \times Recall}{Precision + Recall} \quad (7)$$

$$FPR = \frac{FP}{TP + TN} \quad (8)$$

As illustrated in Equations (4) to (8), TP represents attacks that are correctly classified, TN represents normal traffic correctly classified, FP represents normal (benign) traffic incorrectly classified as attacks, and FN represents attacks that are incorrectly classified as normal traffic. We chose these metrics due to their importance in intrusion detection tasks, where both FPs and FNs can have serious consequences. A high FPR can cause unnecessary alerts and wasted resources, while a high FNR rate may result in undetected attacks [1, 45]. Therefore, by combining these metrics, our study ensures a thorough evaluation of the model's performance.

4. Experimental Setup

The experiments in this study were conducted using a combination of open-source tools and frameworks on a system with specific hardware and software. The hardware setup includes an HP EliteOne 840 23.8-inch all-in-one desktop PC, running on Windows 11 Pro, powered by a 12th Gen Intel® Core i5-12500 processor and 8GB of RAM. The hardware configuration includes a high-performance system with a multi-core processor, a large RAM, and GPU support for training and tuning ensemble models. The software includes Python 3, executed within a Jupyter Notebook (version 6.5.4) environment with key libraries such as Pandas, Numpy, SciKit-learn, Seaborn, Matplotlib, and Hyperopt, among others. These libraries provide a wide range of ML algorithms and tools for data preprocessing, model evaluation, and hyperparameter tuning, thereby ensuring efficiency and aligning with the objectives of this study. This ensures effective handling of large datasets and complex models.

5. Results and Analysis

This section presents the evaluation results of the ensemble learning models with both default and tuned hyperparameters. Furthermore, we provide details about the hyperparameter tuning configurations and Table 2 shows the results of the default hyperparameter used as the baseline and obtained hyperparameter values used to optimize the performance of the models. We conducted tuning on two scenarios: the imbalanced CICToN-IoT dataset and the balanced dataset using SMOTE. Moreover, binary and multi-class classification were performed across all experiments. In experiment 1, we used the default hyperparameters for the RF and XGBoost models as a baseline for comparison with results from other experiments. In experiment 2, we

Table 2. Default and optimized hyperparameter values

			Imbalanced Dataset		**Balanced Dataset**	
Experiment	**Technique**	**Model**	**Hyperparameter Settings for Binary Classification**	**Hyperparameter Settings For multi-classification**	**Hyperparameter Settings for Binary Classification**	**Hyperparameter Settings For multi-classification**
1	Default	RF	max_depth: none, min_samples_split: 2, min_samples_leaf: 1, n_estimators: 100	max_depth: none, min_samples_split: 2, min_samples_leaf: 1, n_estimators: 100	max_depth: none, min_samples_split: 2, min_samples_leaf: 1, n_estimators: 100	max_depth: , min_samples_split: 2, min_samples_leaf: 1, n_estimators: 100
		XGBoost	learning_rate: none, max_depth: none, n_estimators: 100, subsample: none	learning_rate: none, max_depth: none, n_estimators: 100, subsample: none	learning_rate: none, max_depth: none, n_estimators: 100	learning_rate: none, max_depth: none, n_estimators: 100

			Imbalanced Dataset		**Balanced Dataset**	
Experiment	**Technique**	**Model**	**Hyperparameter Settings for Binary Classification**	**Hyperparameter Settings For multi-classification**	**Hyperparameter Settings for Binary Classification**	**Hyperparameter Settings For multi-classification**
2	BO TPE	RF	max_depth: 19, min_samples_split: 7, min_samples_leaf: 1, n_estimators: 100	max_depth: 18, min_samples_split: 5, min_samples_leaf: 7, n_estimators: 200	max_depth: 20, min_samples_split: 3, min_samples_leaf: 1, n_estimators: 300	max_depth: 20, min_samples_split: 9, min_samples_leaf: 9, n_estimators: 50
		XGBoost	learning_rate: 0.8426, max_depth: 9, n_estimators: 100	learning_rate: 0.1347, max_depth: 15, n_estimators: 50	learning_rate: 0.7798, max_depth: 12, n_estimators: 100	learning_rate: 0.6211, max_depth: 7, n_estimators: 150
3	HGS	RF	max_depth: 20, min_samples_split: 2, min_samples_leaf: 1, n_estimators: 200	max_depth: 20, min_samples_split: 10, min_samples_leaf: 2, n_estimators: 100	max_depth: 20, min_samples_split: 5, min_samples_leaf: 1, n_estimators: 200	max_depth: none, min_samples_split: 1, min_samples_leaf: 2, n_estimators: 200
		XGBoost	learning_rate: 0.1, max_depth: 7, n_estimators: 200.	learning_rate: 0.3, max_depth: 7, n_estimators: 200	learning_rate: 0.1, max_depth: 7, n_estimators: 200	learning_rate: 0.3, max_depth: 7, n_estimators: 200
4	RS	RF	max_depth: none, min_samples_split: 5, min_samples_leaf: 1, n_estimators: 100.	max_depth: 20, min_samples_split: 10, min_samples_leaf: 2, n_estimators: 100	max_depth: none, min_samples_split: 5 , min_samples_leaf: 1, n_estimators: 100	max_depth: none, min_samples_split: 5, min_samples_leaf: 1, n_estimators: 100
		XGBoost	learning_rate: 0.3, max_depth: 5, n_estimators: 200	learning_rate: 0.3, max_depth: 5, n_estimators: 200	learning_rate: 0.3, max_depth: 5, n_estimators: 200	learning_rate: 0.3, max_depth: 5, n_estimators: 200

utilized the Hyperopt library for BO using TPE as a search method that yielded BO-TPE to explore the search space and minimize the loss function. The maximum number of trials was set to 50 for binary and 10 for multi-class classification. This process was repeated on both imbalanced and balanced datasets to capture the best hyperparameters and the accuracy as an evaluation metric.

As shown in Table 2, for experiments 3 and 4, we used HalvingGridSearchCV and RandomSearchCV as search methods, respectively, with a 5-fold cross-validation and accuracy as a metric to find the best hyperparameter combination for both classification methods on imbalanced and balanced datasets. Additionally, for experiment 3 the factor was set to 2 for each iteration and for experiment 4 the number of iterations for random combinations was set to 10. Finally, the hyperparameters used for tuning RF include: the number of trees, the maximum depth of the tree (max_depth), the minimum number of samples required for splitting (min_samples_split), and the minimum number at a leaf node (min_samples_leaf). For XGBoost, we use the learning rate (learning_rate), the number of boosting rounds (n_estimators) and the maximum tree depth (max_depth). Table 2, presents the best hyperparameter results for the models after optimization.

5.1. Default Hyperparameter Baseline

Fig. 5. presents the results obtained using default hyperparameters and Fig. 6. shows the computation time where IMBD is the imbalanced dataset and BLD is the balanced dataset. As shown, XGBoost slightly outperformed RF on the imbalanced dataset, achieving 99.23% accuracy, 99.32% F1-score, and a 1.34% FPR, compared to RF's 86.13%, 48.87%, and 40.13%. XGBoost also had faster prediction times, taking 0.510s for binary classification and 0.949s for multi-classification as shown in Fig. 5. After balancing the dataset, XGBoost still outperformed RF in binary classification with 99.32% accuracy. However, for multi-classification, RF outperformed with 93.72% accuracy, 93.69% F1-score, and a lower FPR of 6.27%. Despite this, RF took longer to predict, requiring 71.974s. These results reveal that XGBoost performed better on the imbalanced dataset with higher accuracy, F1-score, and faster prediction times. After balancing the dataset, XGBoost remained superior in binary classification, while RF remained superior in multi-classification despite slower predictions.

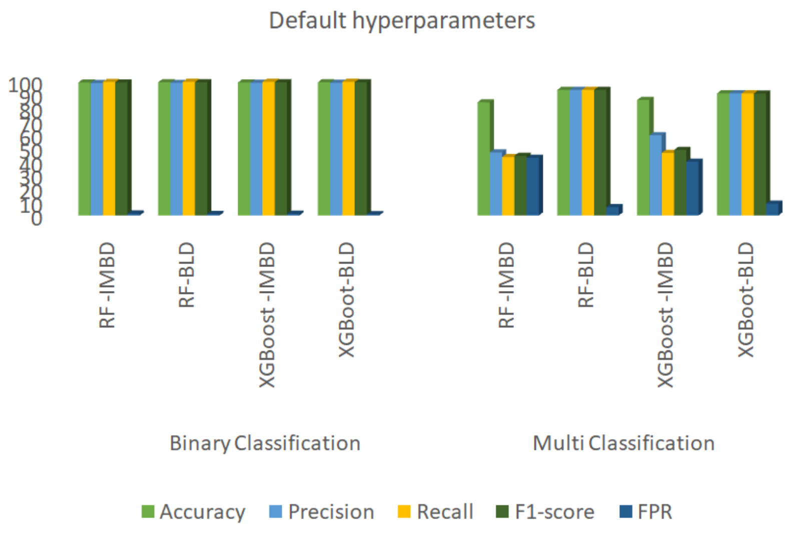


Fig.5. Model performance with default hyperparameters

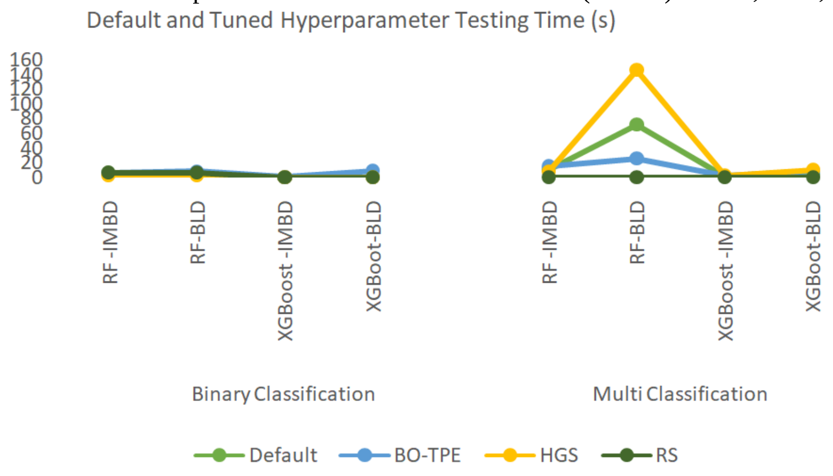


Fig.6. Testing time for default and tuned hyperparameters

5.2. Models Tuned with BO-TPE

Fig. 7 presents the performance of the models after tuning with BO-TPE. The RF model was slightly outperformed by XGBoost, achieving 99.18% accuracy for binary classification and 86.09% for multi-classification. RF also had a higher F1-score of 99.28% for binary classification and a lower FPR of 3.94% in multi-classification, but slower prediction times with 5.368s for binary and 15.205s for multi-classification as shown in Fig. 6.

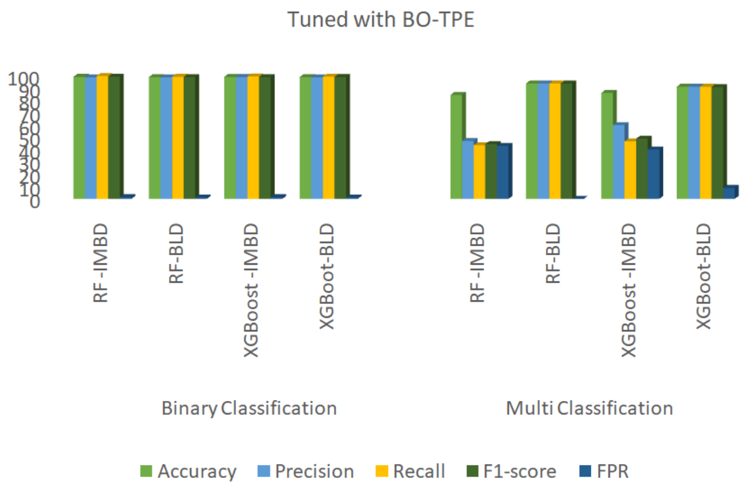


Fig.7. Model performance after tuning with BO-TPE

Accordingly, after balancing the dataset, XGBoost outperformed RF with an accuracy of 98.96% for binary classification and 93.51% for multi-classification. It also achieved an F1-score of 98.97% for binary and 93.46% for multi-classification. Moreover, XGBoost had an FPR of 1.35% for binary classification and 6.47% for multi-classification with a lower prediction time of 8.873s. As presented, the findings indicate that RF performed better on the imbalanced dataset, achieving higher accuracy and F1-score but slower prediction times. After balancing, XGBoost achieved superior results in accuracy, F1-score, and prediction time, especially in multi-classification.

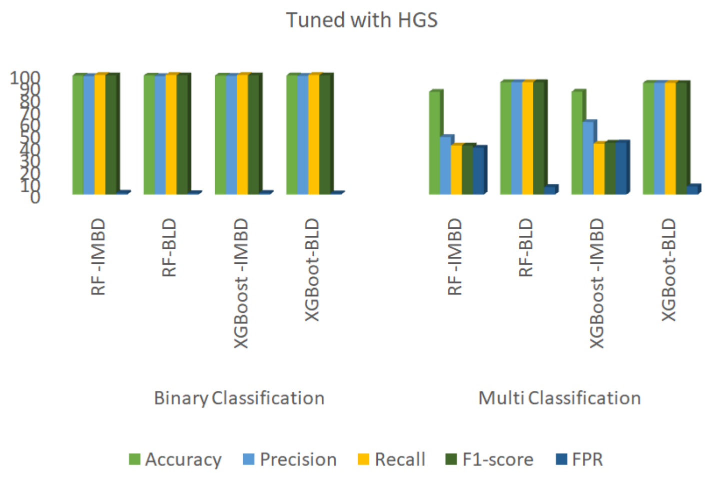


Fig.8. Model performance after tuning with HGS

5.3. Models Tuned with HGS

Fig.8 presents the results of the performance of the models after tuning with HGS. On the imbalanced dataset, XGBoost demonstrates improved performance in several areas. It achieved a higher accuracy, with 99.24% for binary classification and 85.58% for multi-classification and produced a higher F1-score of 99.32% for binary classification. In addition, as presented in Fig. 6, XGBoost showed faster prediction times, taking only 0.293s for binary classification and 2.121s for multi-classification but had a higher FPR of 99.32% for binary classification.

After balancing the dataset, XGBoost outperformed RF in binary classification with 99.34% accuracy, a 99.34% F1-score, and a FPR of 1.02%, along with faster prediction times with 0.089s. RF excelled in multi-classification, achieving 93.76% accuracy, a 93.73% F1-score, and an FPR of 6.22%, but it had a longer prediction time of 146.265s. Overall, the result reveals that XGBoost performed better on the imbalanced dataset with higher accuracy, F1-score, and more accurate predictions. After balancing, XGBoost remained superior in binary classification, while RF was more advantageous for multi-classification, though slower. This suggests that XGBoost could be ideal for efficient binary classification tasks, while RF may be better for multi-classification when prediction time is less of a concern.

5.4. Models Tuned with RS

Fig. 9 shows the performance of the models after tuning with RS. On the imbalanced dataset, the RF model slightly outperformed XGBoost in binary classification, achieving an accuracy of 99.21% and an F1-score of 99.30%. However, XGBoost had a lower FPR of 1.38% and a faster prediction time of 0.598s as shown in Fig. 6. In multi-classification, XGBoost achieved an accuracy of 86.01% and an F1-score of 49.79%, while RF had a higher FPR of 38.79% while both models had the same prediction time of 0.598s. Similarly, after balancing the dataset, both models achieved accuracy and an F1-score of 99.30% in binary classification. XGBoost had a lower FPR of 1.04% and a faster prediction time at 0.383s. In multi-classification, RF outperformed XGBoost with an accuracy of 93.65%, an F1-score of 93.61%, and an FPR of 6.33%, but took longer to predict with 0.383s. Overall, RF slightly outperformed XGBoost in binary classification, while XGBoost was faster with a lower FPR on the imbalanced dataset while XGBoost led in accuracy and F1-score in multi-classification. Nevertheless, after balancing the dataset, both models performed similarly in binary classification, but RF was better in multi-classification, despite taking longer to predict.

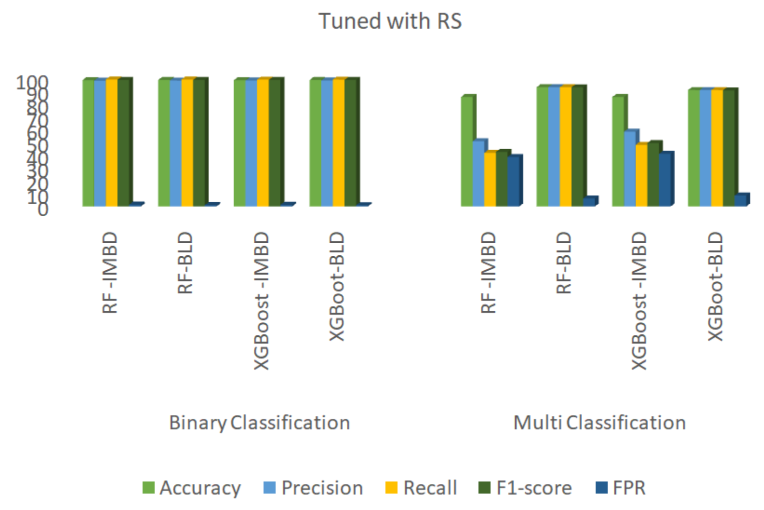


Fig.9. Model performance after tuning with RS

The findings shown in Fig. 5 to 9 are important for enhancing IoT IDS. By utilizing an approach such as HGS, practitioners can create more accurate, efficient, and reliable IDS suited to the unique needs of IoT environments. Furthermore, these insights can guide future research on better tuning methods, hybrid models, and real-time detection frameworks.

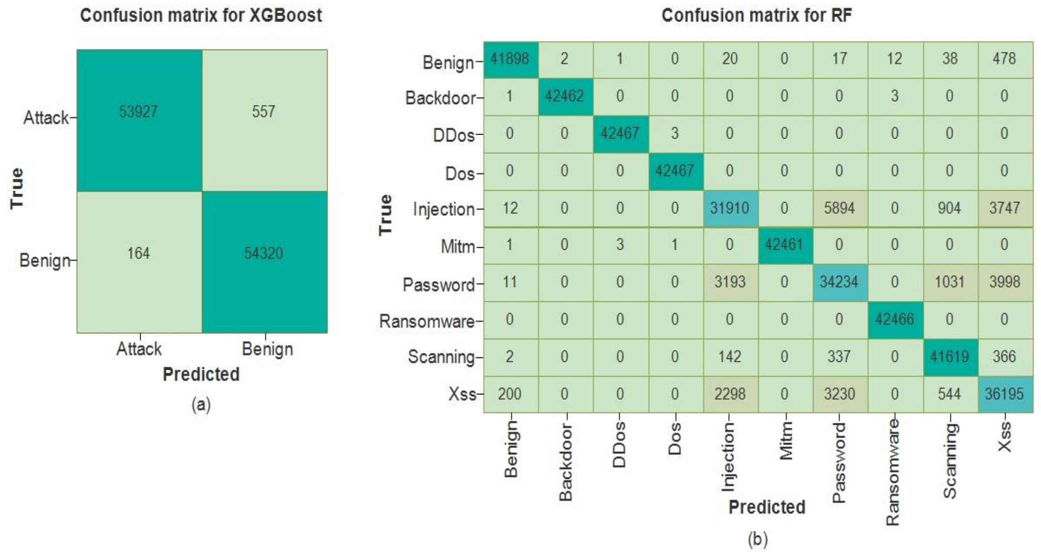
6. DISCUSSIONS

In this section, we examine and provide some key insights into optimizing ensemble learning models for IoT intrusion detection. The findings focus on the effectiveness of HPO methods, their impact on attack types, the challenges faced, and comparison with the state-of-the-art methods. The findings of the default hyperparameters were compared with the other three tuned with BO-TPE, HGS, and RS. The analysis shows that hyperparameter tuning with HGS on the balanced dataset produced promising results. As shown in Table 3, the performance showed modest improvements.

Table 3. Effectiveness of HGS on a balanced dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	FPR (%)
XGBoost	>0.02	>0.02	>0.03	>0.02	<0.02
RF	>0.04	>0.05	>0.04	>0.04	<0.05

For binary classification, the XGBoost model saw a minor increase in accuracy, precision, and F1score by 0.02%. The model's recall also improved by 0.03%, while the FPR decreased by 0.02%. Similarly, for multi-classification, the RF showed a slight improvement in accuracy, precision, and F1-score by 0.04%, with recall increasing by 0.05% and FPR decreasing by 0.05%. In conclusion, hyperparameter optimization using HGS on a balanced dataset proved to be more effective than other techniques, BO-TPE and RS considered in this study. Furthermore, the confusion matrix shown in Fig.10 presents the performance of the XGBoost and RF models after applying the HGS HPO technique and its impact on detection models against attacks.



True \ Predicted	Attack	Benign
Attack	53927	557
Benign	164	54320

True \ Predicted	Benign	Backdoor	DDos	Dos	Injection	Mitm	Password	Ransomware	Scanning	Xss
Benign	41898	2	1	0	20	0	17	12	38	478
Backdoor	1	42462	0	0	0	0	0	3	0	0
DDos	0	0	42467	3	0	0	0	0	0	0
Dos	0	0	0	42467	0	0	0	0	0	0
Injection	12	0	0	0	31910	0	5894	0	904	3747
Mitm	1	0	3	1	0	42461	0	0	0	0
Password	11	0	0	0	3193	0	34234	0	1031	3998
Ransomware	0	0	0	0	0	0	0	42466	0	0
Scanning	2	0	0	0	142	0	337	0	41619	366
Xss	200	0	0	0	2298	0	3230	0	544	36195

Fig.10. Model performance against attacks

Accordingly, for binary classification, the XGBoost model correctly classified 98.98%, about 53.927 instances of the attacks and 99.70% about 54.320 instances of the benign traffic. It misclassified only 1.02%, approximately 557 instances of malicious traffic as attacks and 0.30%, approximately 164 instances of attacks as malicious. In the same vein, for multi-classification, the RF model achieved perfect predictions for DoS and Ransomware attack classes and nearly perfect predictions for Backdoor, DDoS, and MitM attacks with 99.99% correct. It also classified malicious traffic and scanning attacks correctly in 98.66%, approximately 41.898 instances, and 98.04%, approximately 41.698 instances of cases, respectively. However, 1.34% of 568 instances of malicious traffic and 1.96%, approximately 847 instances of scanning attacks were misclassified as other attack types.

Despite these successes, the RF model faced difficulties in distinguishing between XSS, password, and injection attack classes, which share similar features. For the XSS attack class, the model correctly identified 85.23% (36.195 instances), but misclassified some as password attacks (7.61%), injection attacks (5.41%), scanning attacks (1.28%), and benign traffic (0.47%). For password attacks, the model correctly classified 80.61% (34.234 instances), but misidentified 9.41% (3.998 instances) as XSS attacks, 7.52% (3.198 instances) as injection attacks, and smaller percentages as scanning attacks and benign traffic. Similarly, for the injection attack class, the model identified 75.14% (31.910 instances) correctly but misclassified significant portions as password attacks (13.88%), XSS attacks (8.82%), scanning attacks (2.13%), and benign traffic (0.03%). These findings stress the importance of distinguishing features in ML models as overlapping traits between attack types can lead to misclassifications. This highlights the need for improved feature engineering and model refinement to improve detection accuracy.

Hyperparameter tuning, which is essential for optimizing ensemble learning models, presents numerous challenges. One of the primary challenges in this study is mainly due to the computational system capacity of the machinery device used for the experiments. Consequently, we could only use 10% of the CICToN-IoT dataset instead of the full set. Moreover, we were unable to explore higher values for hyperparameter tuning, such as setting the maximum number of evaluations to 70–100 for BO-TPE or using 10-fold cross-validation for RS and HGS optimization. To deal with misclassification trends, particularly for classes such as Password, XSS, and Scanning attacks, more thorough feature analysis is needed to identify and engineer distinctive features that help separate these classes. Dimensionality reduction techniques like

PCA and feature selection methods such as Analysis of Variance (ANOVA) could also improve class separability. Finally, to enhance model performance, particularly for multi-classification, we plan to implement a stacking ensemble model that complements the strengths of both XGBoost and RF algorithms.

To establish the effectiveness of our study, Table 4 compares the proposed solution with similar existing papers based on the same ensemble models and HPO techniques. The findings showed that the proposed model performs exceptionally well for binary classification compared to other studies [36, 39]. However, the results obtained in [9], showed a better accuracy of 96.82% compared to ours specifically for multi-classification. In addition, their solution achieved a precision, recall, and F1-score of 97.00%, respectively. Although the evaluation was performed on diverse datasets, our findings highlight the impact of the HPOs on ensemble methods to improve cybersecurity, particularly in protecting IoT ecosystems from numerous cyber threats.

Table 4. Comparison with state-of-the-art methods

Ref.	**Model**	**HPO**	**Dataset**	**Classification Method**	**Performance Results**
[36]	FL-XGBoost	BO	WUSTL-EHMS-2020	Binary	Accuracy (96.00%), Precision (96.00%), Recall (96.00%), and F1-score (96.00%).
[39]	XGBoost	RS	DDOS Attack Network Logs	Binary	Accuracy (97.00%), Precision (98.00%), Recall (96.00%), and F1-score (97.00%).
[9]	RF	RS	CICDDOS2019	Multi	Accuracy (96.82%), Precision (97.00%), Recall (97.00%), and F1-score (97.00%).
Proposed Solution	XGBoost RF	HGS	CICToNIoT(10%)	Binary Multi	Accuracy (99.34%), Precision (98.99%), Recall (99.70%), and F1-score (99.34%). Accuracy (93.76%), Precision (93.78%), Recall (93.76%), and F1-score (93.73%)

7. CONCLUSIONS

This paper has examined the significant impact of hyperparameter tuning on the performance of ensemble learning models for detecting IoT-related cyberattacks. We conducted four experiments using both imbalanced and balanced CICToNIoT datasets and assessed the models via binary and multi-classification. We used the default hyperparameters as a baseline for the RF and XGBoost models in the first experiment and later applied BO-TPE, HGS and RS to tune the models' hyperparameters in the remaining three experiments, respectively. The findings in comparison to the default baseline showed that hyperparameter optimization with HGS produced the best results on the balanced dataset. The XGBoost model performed best in binary classification and outperformed in multi-classification in terms of accuracy, F1-score and FPR. On the other hand, the RF model predicted various attacks such as DoS/DDoS, ransomware, backdoor, MitM, benign traffic, and scanning attacks. However, it had difficulty distinguishing between injection, password, and XSS attacks. These findings point to the importance of correctly adjusting hyperparameters based on IoT data features and the strengths of each ensemble model. By properly tuning XGBoosst with HGS, more accurate and efficient IDS for IoT environments can be created. This study provides excellent insights into designing real-time threat detection systems and the integration of ML into IoT security. Our future work will focus on enhancing model performance and robustness by using the full CIC-TON-IoT dataset and applying advanced feature selection and dimensionality reduction techniques to improve class separability, especially for misclassified attacks such as password, XSS, and injection. In addition, we aim to build a stacking ensemble model to combine the strengths of XGBoost and RF models for better multi-classification performance. Lastly, a comparison of these models with advanced DL models and the use of another similar benchmark dataset, such as CICIoT2023, is recommended.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

ACKNOWLEDGEMENTS

This research is supported by the Department of Computer Science and the Unit for Data Science and Computing at the North-West University Mafikeng campus.

REFERENCES

[1] B. Isong, O. Kgote, and A. Abu-Mahfouz, "Insights into Modern Intrusion Detection Strategies for Internet of Things Ecosystems," *Electronics,* vol. 13, p. 2370, 2024.

[2] N. Mishra and S. Pandya, "Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review," *IEEE Access,* vol. 9, pp. 59353-59377, 2021.

[3] B. R. Kikissagbe and M. Adda, "Machine learning-based intrusion detection methods in IoT systems: A comprehensive review," *Electronics,* vol. 13, p. 3601, 2024.

[4] S. Bharati and P. Podder, "Machine and deep learning for iot security and privacy: applications, challenges, and future directions," *Security and communication networks,* vol. 2022, p. 8951961, 2022.

[5] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," *Journal of Information Security and Applications,* vol. 58, p. 102804, 2021.

[6] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Communications Surveys & Tutorials,* vol. 22, pp. 1686-1721, 2020.

[7] H. Bakır and Ö. Ceviz, "Empirical enhancement of intrusion detection systems: a comprehensive approach with genetic algorithm-based hyperparameter tuning and hybrid feature selection," *Arabian Journal for Science and Engineering,* pp. 1-19, 2024.

[8] W. Chimphlee and S. Chimphlee, "Intrusion detection system (IDS) development using tree-based machine learning algorithms," *International Journal of Computer Networks and Communications,* vol. 15, pp. 93-109, 2023.

[9] M. V. Gaur and R. Kumar, "Hpddos: a hyperparameter model for detection of multiclass ddos attacks," *Mathematical Statistician and Engineering Applications,* vol. 71, pp. 1444-1470, 2022.

[10] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing,* vol. 415, pp. 295-316, 2020.

[11] M. Sarhan, S. Layeghy, and M. Portmann, "Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection," *Big Data Research,* vol. 30, p. 100359, 2022.

[12] R. Lohiya and A. Thakkar, "Application domains, evaluation data sets, and research challenges of IoT: A systematic review," *IEEE Internet of Things Journal,* vol. 8, pp. 8774-8798, 2020.

[13] Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: a deep learning–based intrusion detection framework for securing IoT," *Transactions on Emerging Telecommunications Technologies,* vol. 33, p. e3803, 2022.

[14] K. Saranya and A. Valarmathi, "A Comparative Study on Machine Learning based Cross Layer Security in Internet of Things (IoT)," in *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS),* 2022, pp. 267-273.

[15] A. Khanna and S. Kaur, "Internet of things (IoT), applications and challenges: a comprehensive review," *Wireless Personal Communications,* vol. 114, pp. 1687-1762, 2020.

[16] H. U. Rehman, M. Asif, and M. Ahmad, "Future applications and research challenges of IOT," in *2017 International conference on information and communication technologies (ICICT),* 2017, pp. 68-74.

[17] I. Martins, J. S. Resende, P. R. Sousa, S. Silva, L. Antunes, and J. Gama, "Host-based IDS: A review and open issues of an anomaly detection system in IoT," *Future Generation Computer Systems,* vol. 133, pp. 95-113, 2022.

[18] R. Alshamy and M. AKCAYOL, "Intrusion Detection Model using Machine Learning Algorithms On Nsl-Kdd Dataset," *International Journal of Computer Networks and Communications,* vol. 16, 2024.

[19] B. A. Tama and S. Lim, "Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation," *Computer Science Review,* vol. 39, p. 100357, 2021.

[20] M. A. Talukder, M. M. Islam, M. A. Uddin, K. F. Hasan, S. Sharmin, S. A. Alyami, *et al.*, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of big data,* vol. 11, p. 33, 2024.

[21] T. Lai, F. Farid, A. Bello, and F. Sabrina, "Ensemble learning based anomaly detection for IoT cybersecurity via Bayesian hyperparameters sensitivity analysis," *Cybersecurity,* vol. 7, p. 44, 2024.

[22] Y. Alotaibi and M. Ilyas, "Ensemble-learning framework for intrusion detection to enhance internet of things' devices security," *Sensors,* vol. 23, p. 5568, 2023.

[23] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access,* vol. 10, pp. 99129-99149, 2022.

[24] T. N. Rincy and R. Gupta, "Ensemble learning techniques and its efficiency in machine learning: A survey," in *2nd international conference on data, engineering and applications (IDEA)*, 2020, pp. 1-6.

[25] K. A. Nguyen, W. Chen, B.-S. Lin, and U. Seeboonruang, "Comparison of ensemble machine learning methods for soil erosion pin measurements," *ISPRS International Journal of Geo-Information,* vol. 10, p. 42, 2021.

[26] Z. Ma, S. Cui, and I. Joe, "An Enhanced Proximal Policy Optimization-Based Reinforcement Learning Method with Random Forest for Hyperparameter Optimization," *Applied Sciences,* vol. 12, p. 7006, 2022.

[27] C. H. Goay, N. S. Ahmad, and P. Goh, "Transient simulations of high-speed channels using CNN-LSTM with an adaptive successive halving algorithm for automated hyperparameter optimizations," *IEEE Access,* vol. 9, pp. 127644-127663, 2021.

[28] J. Xu, Q. Fu, and H. Li, "A novel deep learning-based automatic search workflow for CO2 sequestration surrogate flow models," *Fuel,* vol. 354, p. 129353, 2023.

[29] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv preprint arXiv:2003.05689,* 2020.

[30] M. Feurer and F. Hutter, "Hyperparameter optimization," *Automated machine learning: Methods, systems, challenges*, pp. 3-33, 2019.

[31] B. Zhang, R. Rajan, L. Pineda, N. Lambert, A. Biedenkapp, K. Chua, *et al.*, "On the importance of hyperparameter optimization for model-based reinforcement learning," in *International Conference on Artificial Intelligence and Statistics,* 2021, pp. 4015-4023.

[32] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 13, p. e1484, 2023.

[33] P. Brazdil, J. N. van Rijn, C. Soares, and J. Vanschoren, "Metalearning for hyperparameter optimization," in *Metalearning: Applications to Automated Machine Learning and Data Mining,* ed: Springer, 2022, pp. 103-122.

[34] A. Alsarhan, M. AlJamal, O. Harfoushi, M. Aljaidi, M. M. Barhoush, N. Mansour, *et al.*, "Optimizing Cyber Threat Detection in IoT: A Study of Artificial Bee Colony (ABC)-Based Hyperparameter Tuning for Machine Learning," *Technologies,* vol. 12, p. 181, 2024.

[35] O. R. Sanchez, M. Repetto, A. Carrega, and R. Bolla, "Evaluating ML-based DDoS detection with grid search hyperparameter optimization," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft),* 2021, pp. 402-408.

[36] B. Guembe, S. Misra, and A. Azeta, "Federated Bayesian optimization XGBoost model for cyberattack detection in internet of medical things," *Journal of Parallel and Distributed Computing,* vol. 193, p. 104964, 2024.

[37] M. Mohy-Eddine, A. Guezzaz, S. Benkirane, and M. Azrour, "An efficient network intrusion detection model for IoT security using K-NN classifier and feature selection," *Multimedia Tools and Applications,* vol. 82, pp. 23615-23633, 2023.

[38] M. Injadat, A. Moubayed, and A. Shami, "Detecting botnet attacks in IoT environments: An optimized machine learning approach," in *2020 32nd International Conference on Microelectronics (ICM),* 2020, pp. 1-4.

[39] Z. M. Jiyad, A. Al Maruf, M. M. Haque, M. S. Gupta, A. Ahad, and Z. Aung, "DDoS Attack Classification Leveraging Data Balancing and Hyperparameter Tuning Approach using Ensemble Machine Learning with XAI," in *2024 Third International Conference on Power, Control and Computing Technologies (ICPC2T),* 2024, pp. 569-575.

[40] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," *Sustainable Cities and Society,* vol. 72, p. 102994, 2021.

[41] E. K. Sahin, "Comparative analysis of gradient boosting algorithms for landslide susceptibility mapping," *Geocarto International,* vol. 37, pp. 2441-2465, 2022.

[42] V. Azizi and G. Hu, "Machine learning methods for revenue prediction in google merchandise store," in *Smart Service Systems, Operations Management, and Analytics: Proceedings of the 2019 INFORMS International Conference on Service Science,* 2020, pp. 65-75.

[43] S. Sivasubramaniam and S. Balamurugan, "An Accurate Hyperparameter Tuning Based Machine Learning Model For Heart Disease Identification," *Innovative Technologies and its Applications in Higher Education,* p. 193.

[44] H. Dong, D. He, and F. Wang, "SMOTE-XGBoost using Tree Parzen Estimator optimization for copper flotation method classification," *Powder Technology,* vol. 375, pp. 174-181, 2020.

[45] R. M. AlZoman and M. J. Alenazi, "A comparative study of traffic classification techniques for smart city networks," *Sensors,* vol. 21, p. 4677, 2021.