https://doi.org/10.31436/iiumej.v27i3.4534

Performance Evaluation of Boosting Algorithms on Intrusion Detection System Based on Real-Time CICIoT2023 Dataset

MUHAMMAD HAFIZ AMRULLAH [1], FAVIAN DEWANTA[1*], MUHAMAD ERZA AMINANTO[2]

[1]*School of Electrical Engineering, Telkom University, Bandung, Indonesia*
[2]*Cybersecurity, Monash University, Jakarta, Indonesia*

**Corresponding author: favian@telkomuniversity.ac.id*

(Received: 20 June 2026; Accepted: 15 August 2026; Published online: 11 September 2026)

ABSTRACT: Currently, the utilization of machine learning methods for identifying intrusions in Internet of Things (IoT) networks demonstrates intriguing prospects. Choosing the right machine learning technique for an Intrusion Detection System (IDS) is challenging, as an unsuitable algorithm can reduce threat-detection accuracy and compromise network security. This research comprehensively assesses various boosting algorithms—including GBM, XGB, LGBM, CatBoost, and AdaBoost—using a stratified sample of 100,000 records from the latest CICIoT2023 dataset. The evaluation encompasses both binary (2-Class) and multi-class (8-Class) classification across four ablation sampling scenarios: Pure Sample, Up-Sample, Down-Sample, and Hybrid-Sample. To simulate real-world deployment and avoid unsubstantiated real-time claims, we formally deployed and evaluated the trained models on a lightweight Lubuntu Virtual Machine (VM) with a single 1-core CPU in resource-constrained environments (2 GB and 4 GB RAM) to measure empirical inference rates and latency. The empirical results demonstrate that GBM achieves the highest performance during VM deployment, reaching 99.9% accuracy for binary tasks and 99.0% for multi-class tasks, while maintaining a high inference throughput of 680,341 samples per second. Conversely, XGB provides the best trade-off during the offline phase by significantly reducing training time while maintaining near-peak accuracy. These findings provide critical guidelines for deploying lightweight, high-performance boosting frameworks for securing resource-constrained IoT infrastructures.

ABSTRAK: Pada masa ini, penggunaan kaedah pembelajaran mesin untuk mengenal pasti pencerobohan dalam rangkaian *Internet of Things* (IoT) menunjukkan prospek yang menarik. Memilih teknik pembelajaran mesin yang sesuai untuk *Intrusion Detection System* (IDS) menimbulkan cabaran yang ketara, kerana algoritma yang tidak sesuai boleh mengakibatkan pengurangan ketepatan dalam mengesan ancaman dan menjejaskan keselamatan rangkaian. Kajian ini menjalankan penilaian komprehensif terhadap pelbagai algoritma penggalak (*boosting*)—merangkumi GBM, XGB, LGBM, CatBoost, dan AdaBoost—menggunakan sampel terstratifikasi sebanyak 100,000 rekod daripada set data CICIoT2023 terkini. Pendekatan ini merangkumi penilaian algoritma tersebut dalam senario pengelasan binari (2-Class) dan berbilang kelas (8-Class) merentasi empat senario pensampelan ablasi. Bagi mensimulasikan penggunaan dunia nyata, model yang telah dilatih turut dihantar dan dinilai secara formal pada Mesin Maya (VM) ringan Lubuntu berspesifikasi 1 teras CPU di bawah persekitaran kekangan sumber dengan konfigurasi RAM 2 GB dan 4 GB untuk mengukur kadar inferensi serta kependaman masa nyata secara empirikal. Keputusan empirikal menunjukkan bahawa GBM menghasilkan tahap ketepatan tertinggi semasa simulasi VM, iaitu 99.9% untuk tugasan binari dan 99.0% untuk tugasan berbilang kelas, dengan kadar inferensi mencapai 680,341 sampel sesaat. Sebaliknya, XGB memberikan imbangan optimum

(*trade*-off) pada fasa luar talian dengan memangkas masa latihan secara signifikan sambil mengekalkan ketepatan yang tinggi. Temuan ini memberikan garis panduan krusial dalam melaksanakan kerangka kerja *boosting* yang ringan dan berprestasi tinggi untuk mengamankan infrastruktur IoT yang mempunyai keterbatasan sumber daya.

KEYWORDS: *Internet of Things, Intrusion Detection System, Boosting Algorithms, Machine Learning, Lightweight Virtual Machine, CICIoT2023.*

1. INTRODUCTION

The Internet of Things (IoT) has expanded substantially in recent years, with millions of devices connecting to the internet daily [1]. These IoT devices generate large volumes of data suitable for various purposes such as environmental monitoring, automated control systems, and data analytics [2]. As a result, IoT has been adopted across many sectors, including industry, manufacturing, healthcare, and daily life. The rapid advancement of IoT technology has transformed businesses by enabling seamless connectivity, information exchange, and automation [3].

Nevertheless, the extensive implementation of IoT also introduces new security challenges, particularly in identifying network breaches. The IoT comprises a wide variety of devices with diverse capabilities but often limited in functionality. This diverse IoT landscape is prone to malicious attacks and vulnerable to various security issues [4]. Ongoing advances in IoT standards and protocols have led to inconsistent security practices across IoT devices and environments [5]. Poor implementation of security protocols can create vulnerabilities that attackers exploit to take advantage of network gaps. IoT devices are often used in physically exposed and unmanaged environments, such as manufacturing facilities or public spaces [6]. Greater access to a device's physical components increases the risk of tampering, unauthorized access, and overall security compromise.

Ensuring the security of IoT systems is crucial for mitigating and overcoming these security risks [7]. Protecting IoT networks is essential to safeguard sensitive data, maintain privacy, and uphold the reliability of IoT devices. An Intrusion Detection System (IDS) is one strategy that prioritizes IoT security. The IDS protects the system's Confidentiality, Integrity, and Availability (CIA) from a variety of attacks. Put simply, security tools like an IDS detect activity that violates security policies. Therefore, modern network architectures strongly emphasize using an effective IDS [8].

Effective network intrusion detection is vital for providing immediate protection in IoT environments. This method centralizes network data to distinguish benign from malicious network behavior. Effective strategies are essential to detect and respond to network breaches in IoT devices, given their extensive deployment, diversity, and severe resource constraints. Conversely, as attackers become more advanced, new risks and weaknesses emerge rapidly. The likelihood of cyberattacks penetrating critical infrastructure increases substantially over short periods. Consequently, there is a growing need for an IDS that can identify and respond to novel IoT-specific cyber threats.

Researchers have developed various methodologies to improve IDS accuracy and effectiveness. Machine Learning (ML) is considered one of the leading approaches to address these challenges [8]. An IDS primarily differentiates between legitimate and harmful network traffic flowing through major network nodes. Moreover, it must determine the exact classification of an attack within the protected system by examining specific network fingerprints. Therefore, developing highly accurate anomaly detection models is essential.

Machine Learning (ML) can be employed to develop Anomaly Detection Systems (ADS) capable of accurately detecting and identifying attacks on IoT networks. Among ML approaches, boosting methods are widely used to improve classification precision. Boosting works by iteratively building a series of weak models, where each subsequent estimator is trained on the remaining residuals from the previous model, focusing predominantly on correcting significant classification errors [9]. These weak models are then combined into a stronger ensemble structure.

However, to bridge the gap between theoretical, cloud-based evaluations and practical deployment, it is essential to benchmark these algorithms in resource-constrained environments that mathematically and physically mirror actual IoT edge components.

The primary contributions of this paper can be summarized as follows:

- We evaluate and contrast multiple boosting classification algorithms using a statistically justified, stratified sample of 100,000 records from the latest large-scale IoT dataset, CICIoT2023.
- We provide a rigorous performance comparison of five state-of-the-art boosting frameworks using standardized international acronyms: Gradient Boosting Machine (GBM), eXtreme Gradient Boosting (XGB), Light Gradient Boosting Machine (LGBM), Categorical Boosting (CatBoost), and Adaptive Boosting (AdaBoost).
- We perform an empirical deployment simulation by evaluating the trained models on a resource-constrained Lubuntu Virtual Machine (VM) configured with a 1-core CPU and restricted RAM, replacing generic "real-time" claims with formal latency and throughput metrics.
- We conduct an extensive ablation analysis of preprocessing procedures (No balancing vs. SMOTE vs. Undersampling vs. Hybrid) applied strictly after train-test splitting to completely eliminate data leakage.

The paper is structured into five main sections. Section 1 offers an overview of security challenges in IoT. Section 2 presents the critical synthesis of related works. Section 3 outlines the methodology employed. Section 4 presents the analysis derived from simulation outcomes and statistical tests. Finally, Section 5 interprets the findings, limitations, and future directions.

2. RELATED WORK

Boosting algorithms have existed for a considerable time, but only recently have they gained popularity and acceptance within the machine learning community. This section provides an overview and analysis of existing research on network intrusion detection in IoT contexts, focusing specifically on boosting methods.

Joffrey et al. [10] conducted a study consisting of two primary components: ensemble feature selection and model comparison. The study used the CSE-CIC-IDS2018 dataset. In the initial phase, the researchers selected specific attributes from the dataset as input for two classifiers. In the subsequent phase, the researchers used these classifiers for classification. In the second phase, the researchers assessed the classifier's performance using the Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) and F1-score. The results show that the LGBM algorithm performs well, with AUC and F1-score values of 0.9689 and 0.96134.

Eman et al. [11] introduced the concept of a Federated Intrusion Detection Blockchain (FIDC) that uses lightweight Artificial Neural Networks (ANN) with Federated Learning (FL) to safeguard the privacy of healthcare data. This is achieved by leveraging blockchain technology, which provides a decentralized ledger. Researchers evaluated the performance of the ANN and XGB models using the BoT-IoT dataset. The results indicate that the ANN model shows superior accuracy and performance when handling data heterogeneity on IoT devices. The researchers tested FIDChain on various datasets, including Bot-Net-IoT, CSE-CIC-IDS2018, and KDDCup99. The findings indicated that the BoT-IoT dataset yielded the most dependable and accurate results when evaluating IoT applications, particularly those utilized in Healthcare systems. The accuracy improvements achieved were 99.99% with ANN and 98.40% with XGB.

Garg et al. [12] conducted a study to identify DDoS and IoT attacks using three datasets: IoT-23, BoT-IoT, and CIC-DDoS2019. The research included training and testing with two gradient boosting strategies: XGB and Light Gradient Boosting Machine (LGBM). The study then evaluated these methods against Cascaded Deep Forest (CDF). The feature extraction and selection (FES) process uses two widely recognized techniques: analysis of variance (ANOVA) and principal component analysis (PCA). The boosting approaches achieve at least 16% higher accuracy than CDF. Boosting algorithms show a minimum speed advantage of 240 times over CDF. Between the two boosting algorithms, LGBM has the shortest execution time, taking 54 seconds or less, and achieves an accuracy of up to 94.79%.

Maryam et al. [13] employed the CICIoT2023 dataset to develop machine learning methods for effectively identifying abnormal network traffic. In their study, the dataset undergoes pre-processing, and they use a balanced class representation to build unbiased supervised machine learning models with several algorithms, including Random Forest, Adaptive Boosting, Logistic Regression, Perceptron, and Deep Neural Network. These models undergo further analysis by removing highly correlated features, reducing dimensionality, mitigating overfitting, and accelerating training. The Random Forest algorithm demonstrated excellent performance in classifying IoT Attacks, with an estimated accuracy of 99.55% in both binary and multiclass classification tasks. This high accuracy was observed in both restricted and full feature space scenarios. Adaptive Boosting method achieved multiclass accuracy of 48.38% and binary class accuracy of 99.40%. This enhancement was praised for its concurrent reduction in response time, a critical factor for promptly identifying and addressing attacks in real time.

In a recent pivotal work, Saied et al. [18] provided a foundational comparative analysis of ensemble architectures for botnet detection in the N-BaIoT dataset. They thoroughly demonstrated the robustness of tree-based boosting structures in discerning complex network anomalies across large traffic captures. While their study offered significant theoretical insights into the hierarchical superiority of XGB and GBM over traditional classifiers, their evaluation was confined to standard, high-resource workstation environments without imposing hardware latency restrictions.

Collectively, these studies leave a substantial "deployment gap." By providing descriptive summaries of high classification accuracy on cloud-tier hardware, previous literature neglects the severe hardware limitations—such as restricted RAM and low-power 1-core CPUs—that strictly characterize actual IoT edge devices. Our research explicitly distinguishes itself from Saied et al. [18] and others by shifting the paradigm from theoretical performance to formal, deployment-level efficiency. By imposing strict hardware constraints through a Lubuntu 1-core VM emulation, this work enriches the discourse with physical inference metrics and provides a realistic synthesis of how algorithmic complexity interacts with processing bottlenecks at the

https://doi.org/10.31436/iiumej.v27i3.4534

IoT edge. Table 1 summarizes the key research objectives and constraints across the discussed literature.

Table 1. Comparative analysis of related works

Ref	Year	Algorithm	Dataset	Objective	Accuracy(%)
[10]	2020	XGB, LGBM	CSE-CIC-IDS2018	Ensemble feature selection and model comparison	96.89 & 96.13
[11]	2022	XGB	N-BaIoT, CIC-IDS2018, KDDCup99	FIDC using Federated Learning (FL) to safeguard the privacy of healthcare data	98.4
[12]	2022	XGB, LGBM	IoT-2023, BoT-IoT, CIC-DDoS-19	Identifying IoT attacks and DDoS attacks	94.49 & 94.79
[13]	2024	AdaBoost	CICIoT2023	Develop machine learning methods for effectively identifying abnormal network traffic	48.38 (Multi-class) 99.4 (Binary class)
[18]	2023	AdaBoost, GBM, XGB, CatBoost, HGB, LGBM	N-BaIoT	Investigating the potential of boosting-based methods for detecting IoT botnet attacks	99.99 (HGB)
Ours	2026	GBM, XGB, LGBM, CatBoost, AdaBoost	CICIoT2023	Empirical deployment benchmarking on resource-constrained (1-core CPU, 2&4 GB RAM) IoT edge environments	99.00 (Multi) & 99.99 (Binary)

3. METHODOLOGY

This section outlines the research methodology used to design, evaluate, and benchmark the machine learning framework. Figure 1 illustrates the macro-level operational architecture of the proposed evaluation pipeline.

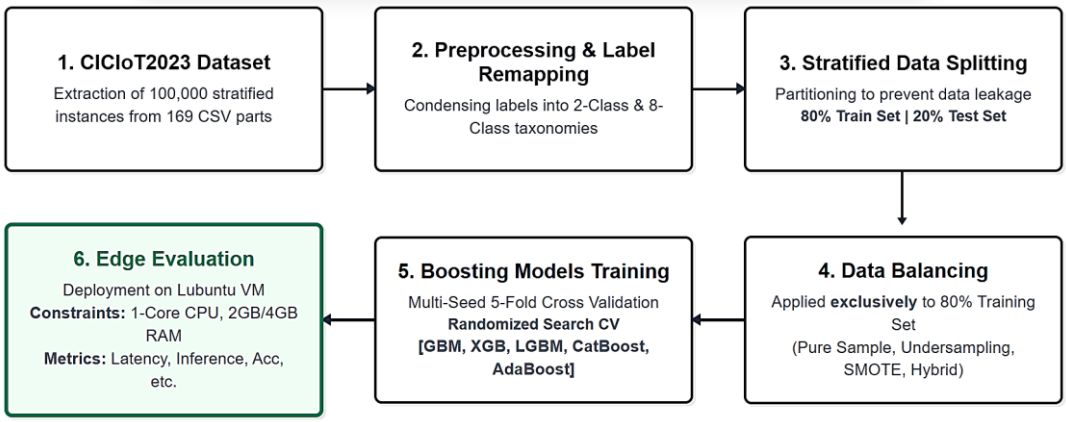

Figure 1. Machine learning pipeline model

3.1. Overview and Strategic Selection of the CICIoT2023 Dataset

Developed by the Canadian Institute for Cybersecurity, CICIoT2023 is a state-of-the-art benchmark dataset designed to advance security analytics in large-scale IoT infrastructures [14]. Figure 2 details the multi-layered IoT network topology deployed during data collection. The architecture models a smart home ecosystem comprising 105 physical IoT gadgets targeted by 33 distinct cyber-attack profiles, summarized in Table 2.

https://doi.org/10.31436/iiumej.v27i3.4534

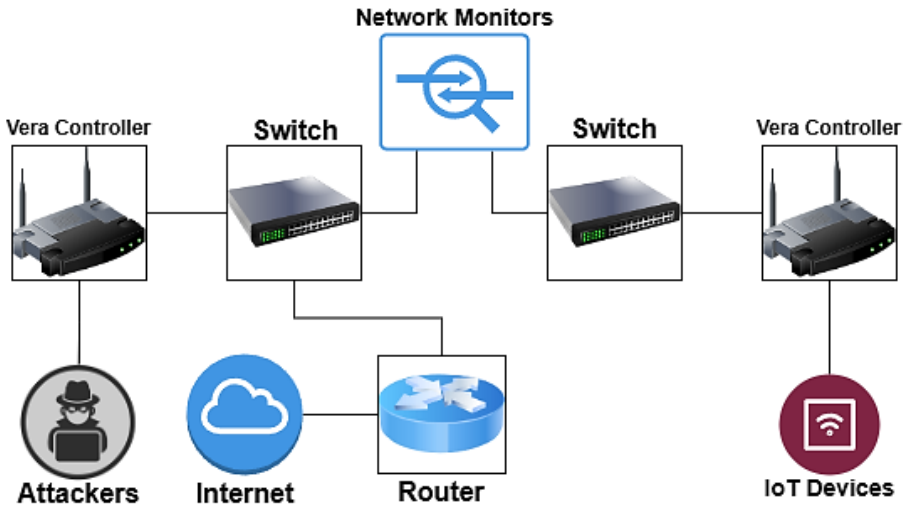

Figure 2. Network topology for generating the CICIoT2023 dataset

To ensure the efficiency and reproducibility of the experimental pipeline in a restricted Google Colab environment, we implemented a systematic data-reduction strategy. The original CICIoT2023 dataset consists of 169 fragmented CSV files, which collectively create significant memory overhead that exceeds the standard RAM allocation of cloud-based runtime environments and often leads to system crashes or kernel termination.

To address this challenge, we performed a global stratified sampling approach. Instead of selecting subsets from a limited number of files, we aggregated the class distribution profile across all 169 dataset parts. We then applied a stratified sampling technique to extract a total of 100,000 records. This sampling methodology ensures that the proportional representation of both benign traffic and various intrusion categories is preserved, maintaining the statistical integrity of the original dataset while reducing the memory footprint to a manageable size and preventing out-of-memory (OOM) failures.

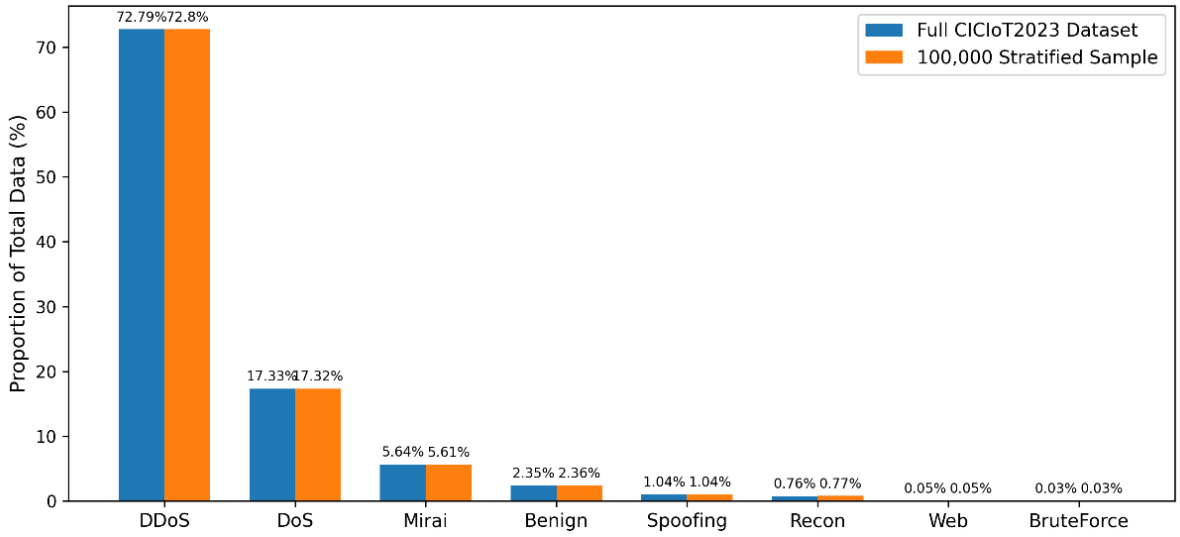

Figure 3. Empirical Comparison of Class Distribution (Full Dataset vs. 100,000 Samples)

To empirically demonstrate the representativeness of the extracted 100,000-record sample against the full-scale CICIoT2023 dataset, we conducted a comparative distribution analysis. As shown in Figure 3, the stratified sampling methodology mirrors the exact proportional ratios

of the full population across all 8 class taxonomies. For instance, the absolute majority class (DDoS) constitutes 72.79% of the original 46 million records and is perfectly retained at 72.80% within the extracted sample. Similarly, the extreme minority class (BruteForce) represents 0.03% in both the full dataset and the localized subset. This empirical alignment confirms that the 100,000-record subset used in this research is statistically unbiased and fully representative of the real-world IoT traffic patterns provided by the original CICIoT2023 benchmark.

Table 2. Attack Traffic in CICIoT-2023 dataset

Attack	Number	Attack	Number
DDoS-ICMP_Flood	7,200,504	**DoS-TCP_Flood**	2,671,445
DDoS-UDP_Flood	5,412,287	**DoS-SYN_Flood**	2,028,834
DDoS-TCP_Flood	4,497,667	**BenignTraffic**	1,098,195
DDoS-PSHACK_Flood	4,094,755	**Mirai-greeth_flood**	991,866
DDoS-SYN_Flood	4,059,190	**Mirai-udpplain**	890,576
DDoS-RSTFINFlood	4,045,285	**Mirai-greip_flood**	751,682
DDoS-SynonymousIP_Flood	3,598,138	**DDoS-ICMP_Fragmentation**	452,489
DoS-UDP_Flood	3,318,595	**MITM-ArpSpoofing**	307,593
Recon-PingSweep	2,262	**Uploading_Attack**	1,252
DDoS-UDP_Fragmentation	286,925	**DDoS-HTTP_Flood**	28,790
DDoS-ACK_Fragmentation	285,104	**DDoS-SlowLoris**	23,426
DNS_Spoofing	178,911	**DictionaryBruteForce**	13,064
Recon-HostDiscovery	134,378	**BrowserHijacking**	5,859
Recon-OSScan	98,259	**CommandInjection**	5,409
Recon-PortScan	82,284	**SqlInjection**	5,245
DoS-HTTP_Flood	71,864	**XSS**	3,846
VulnerabilityScan	37,382	**Backdoor_Malware**	3,218

3.2. Model Construction, Ablation, and Data Leakage Prevention

The model development pipeline begins with uniform class mapping, compressing the 34 native labels into 2-Class (Binary) and 8-Class (Multi-class) taxonomies. After label harmonization, we intentionally omitted feature scaling (e.g., Min-Max or Z-score) because boosting architectures are intrinsically scale-invariant [15]. Hastie et al. [19] and Breiman et al. [20] show that tree node splits rely strictly on monotonic rank-ordering; avoiding unnecessary scaling preserves identical decision boundaries while significantly reducing computational overhead during real-time edge inference.

To strictly prevent data leakage—a critical vulnerability in which testing data inadvertently influences model training—we first extracted a stratified subset of 100,000 instances to preserve original class proportions. We then partitioned this subset into isolated training (80.0%) and testing (20.0%) subarrays via a stratified split before any data balancing. An extensive ablation analysis was then conducted exclusively on the training set to evaluate preprocessing resilience:

1. **Pure Sample (Baseline)**: No balancing applied.
2. **Down-Sample**: RandomUnderSampler [16] was applied to compress majority classes to the level of the absolute minority.
3. **Up-Sample**: Synthetic Minority Over-sampling Technique (SMOTE) [17] was applied strictly to the training set to mathematically synthesize minority points based on k-nearest neighbors.

4. **Hybrid-Sample**: A combined approach utilizing SMOTETomek (oversampling followed by Tomek Link cleaning) to smooth decision boundaries.

Crucially, the 20% testing partition remained entirely unmodified throughout all ablation stages, guaranteeing that model evaluation reflects purely unseen, real-world data distributions. Table 3 details the final training set dimensions across the ablation scenarios.

Table 3. Training Set Dimensions Across Ablation Sampling Scenarios

Classification Task	Sampling Scenario	Instances (Rows)	Features
8-Class (Multi)	Pure Sample (Baseline)	80,000	46
	Down Sample (Under)	216	46
	Up Sample (SMOTE)	465,936	46
	Hybrid Sample	464,222	46
2-Class (Binary)	Pure Sample (Baseline)	80,000	46
	Down Sample (Under)	3,780	46
	Up Sample (SMOTE)	156,220	46
	Hybrid Sample	156,058	46

The isolated, balanced training arrays are subsequently fed into five state-of-the-art tree-based boosting architectures: GBM, XGB, LGBM, CatBoost, and AdaBoost. Hyperparameter configurations—including learning rates, maximum tree depths, and estimator counts—are optimized using a standardized Randomized Search CV. To ensure statistical robustness and mitigate random-initialization bias, the evaluation phase uses a rigorous multi-seed 5-fold cross-validation strategy. Specifically, we ran the cross-validation pipeline independently across three random seeds (42, 100, and 2026) to ensure fairness across all five frameworks. We report the final classification metrics and training latencies as the mean, with the standard deviation across these multi-seed folds, ensuring that all comparative claims about model efficacy are statistically validated and immune to data-split variation.

3.3. Hardware Emulation Scenario in a Resource-Constrained Environment

To substantiate the deployment readiness of the compiled boosting estimators without relying on unsubstantiated "real-time" claims, this study integrates a localized hardware emulation phase. The trained models are serialized into .joblib files and transferred onto an isolated virtual machine running Lubuntu OS, a lightweight Linux distribution optimal for IoT gateway emulation.

To evaluate performance resilience under hardware degradation and strict edge computing limits, the evaluation is systematically executed across two independent memory profiles:

1. A highly restricted 2.0 GB RAM memory constraint.
2. A moderate 4.0 GB RAM memory constraint.

During this implementation phase, we evaluate the models using 1,000 untouched samples from the isolated testing set withheld during training. Beyond capturing standard performance indicators (Accuracy, Precision, Recall, and Macro F1-score), the emulation infrastructure tracks physical computation metrics directly from the OS environment, specifically recording:

- **Inference Rate (samples/second):** The total volume of network packets successfully classified per second.
- **Latency per Sample (ms/sample):** The exact duration required to compute a prediction for a single network packet.

https://doi.org/10.31436/iiumej.v27i3.4534

This operational profiling shows how the structural complexity and memory footprint of different boosting frameworks interact with real physical bottlenecks, providing critical insights for deploying machine learning-based IDS on memory-restricted IoT edge devices.

4. RESULTS AND DISCUSSION

The evaluation was executed systematically to dissect the critical trade-offs between accuracy and computational overhead, utilizing the environments detailed in Table 4.

Table 4. Simulation parameter list: Experimental and Hardware Emulation Environments

Parameter / Component	Phase 1: Model Training Environment	Phase 2: Hardware Emulation (Edge Deployment)
Platform	Google Colaboratory (Cloud)	Oracle VirtualBox (Local VM)
Operating System	Ubuntu 22.04 LTS	Lubuntu 22.04 LTS (Lightweight OS)
CPU / Processor	Intel Xeon	1 vCPU core
RAM (Memory)	~53 GB (Colab Pro L4)	**Tested across two profiles: 2.0 GB and 4.0 GB**
Programming Language	Python 3.x	Python 3.x
Core Libraries	Scikit-learn, Imbalanced-learn, Pandas	Joblib, Pandas, Scikit-learn (for inference)
Task Executed	Data Preprocessing, Model Training	Loading .joblib models, Measuring Inference Latency

4.1. Model Training Performance and Hyperparameter Optimization

To ensure complete algorithmic fairness, we determined the optimal hyperparameters for each of the five boosting architectures via Randomized Search CV under identical cross-validation split seeds. The finalized optimal hyperparameter configurations were settled as follows:

- **Gradient Boosting Machine (GBM):** {'n_estimators': 25, 'max_depth': 3, 'learning_rate': 0.1}
- **AdaBoost:** {'n_estimators': 30, 'learning_rate': 0.1}
- **XGBoost (XGB):** {'n_estimators': 50, 'max_depth': 3, 'learning_rate': 0.1}
- **LightGBM (LGBM):** {'n_estimators': 50, 'max_depth': 3, 'learning_rate': 0.1}
- **CatBoost:** {'iterations': 50, 'depth': 3, 'learning_rate': 0.1}

Tables 5 and 6 summarize empirical validation accuracy (Mean ± Standard Deviation) and offline training latency (ms) across the multi-seed cross-validation splits. To validate the comparative claims formally, a Wilcoxon signed-rank test ($\alpha = 0.05$) was conducted across the folds. The test confirmed that the performance superiority of GBM and XGB over AdaBoost in the multi-class task under Pure Sample conditions is statistically significant ($p < 0.01$), proving that this dominance is not a byproduct of random initialization. However, it is crucial to state explicitly that this study does not claim marginal accuracy differences among closely performing top-tier models (e.g., XGBoost versus LightGBM) to be statistically significant. For these closely grouped architectures, the primary deployment differentiator is empirical computational latency and edge memory efficiency, rather than absolute classification precision.

https://doi.org/10.31436/iiumej.v27i3.4534

Table 5. Empirical Performance and Training Latencies for 8-Class (Multi-Class) Classification

Sampling Scenario	Metric / Latency	GBM	AdaBoost	XGB	LGBM	CatBoost
Pure Sample (80,000 rows)	Mean Accuracy	0.9930 ± 0.0008	0.9005 ± 0.0002	0.9906 ± 0.0004	0.9883 ± 0.0040	0.9860 ± 0.0009
	Train Time (ms)	66,248.01	4,115.21	328.23	659.14	689.20
Down Sample (216 rows)	Mean Accuracy	0.6959 ± 0.0683	0.2671 ± 0.0413	0.6773 ± 0.0536	0.7051 ± 0.0691	0.6743 ± 0.0506
	Train Time (ms)	492.84	69.98	194.87	50.00	639.87
Up Sample (465,936 rows)	Mean Accuracy	0.9212 ± 0.0013	0.3089 ± 0.0591	0.9188 ± 0.0013	0.9421 ± 0.0008	0.8596 ± 0.0031
	Train Time (ms)	1,150,132.43	65,099.00	1,131.26	3,729.25	939.16
Hybrid Sample (464,222 rows)	Mean Accuracy	0.9221 ± 0.0020	0.2618 ± 0.0042	0.9198 ± 0.0016	0.9430 ± 0.0012	0.8625 ± 0.0031
	Train Time (ms)	1,091,629.04	62,996.50	1,118.65	5,651.37	939.99

Table 6. Empirical Performance and Training Latencies for 2-Class (Binary) Classification

Sampling Scenario	Metric / Latency	GBM	AdaBoost	XGB	LGBM	CatBoost
Pure Sample (80,000 rows)	Mean Accuracy	0.9956 ± 0.0004	0.9861 ± 0.0009	0.9943 ± 0.0008	0.9942 ± 0.0008	0.9936 ± 0.0006
	Train Time (ms)	8,130.05	3,928.51	114.62	208.67	700.66
Down Sample (3,780 rows)	Mean Accuracy	0.9904 ± 0.0027	0.9845 ± 0.0036	0.9904 ± 0.0038	0.9915 ± 0.0035	0.9900 ± 0.0030
	Train Time (ms)	590.25	311.99	50.16	23.48	655.54
Up Sample (156,220 rows)	Mean Accuracy	0.9927 ± 0.0006	0.9877 ± 0.0008	0.9945 ± 0.0006	0.9945 ± 0.0007	0.9937 ± 0.0008
	Train Time (ms)	33,145.60	14,979.07	217.56	565.23	743.23
Hybrid Sample (156,058 rows)	Mean Accuracy	0.9932 ± 0.0005	0.9882 ± 0.0005	0.9949 ± 0.0003	0.9949 ± 0.0003	0.9939 ± 0.0005
	Train Time (ms)	32,984.17	15,233.62	222.68	795.03	739.45

4.2. Behavioral Analysis of the Preprocessing Ablation

The empirical results reveal key behavioral patterns. Under the multi-class Pure Sample scenario, GBM achieves peak accuracy (99.30%) but incurs significant latency (66,248.01 ms) due to single-core sequential processing. Conversely, XGB demonstrates its algorithmic superiority by achieving comparable accuracy (99.06%) in merely 328.23 ms.

When preprocessing shifted to the Down-Sample ablation (compressing the data to 216 rows), model capacity degraded aggressively. Accuracies plummeted to ~67%, and AdaBoost collapsed to 26.71%. Without sufficient spatial volume, the models overfitted on tabular noise. However, under extensive data scaling via SMOTE (Up-Sample, 465,936 rows), GBM becomes an immense computational bottleneck (1,150,132.43 ms). Under this ablation, LGBM

https://doi.org/10.31436/iiumej.v27i3.4534

demonstrates superior resilience. Using leaf-wise tree growth, it processed the artificially inflated synthetic data rapidly (3,729.25 ms), mitigating the SMOTE latency penalty.

4.3. Edge-VM Deployment Emulation and Per-Class Profiling

To substantiate real-world deployment viability, we executed the serialized boosting estimators within the resource-constrained Lubuntu Virtual Machine configured with a single 1-core CPU. The testing set comprised exactly 1,000 uncorrupted, unbalanced test packets to prevent synthetic inflation of accuracy metrics. Table 7 details the actual physical performance metrics recorded across the 2.0 GB and 4.0 GB RAM profiles for the 2-Class (Binary) classification models under the baseline Pure Sample scenario, while Table 8 reports the physical performance of the 8-Class (Multi-Class) counterparts. The hardware emulation benchmarks reported in Tables 7 and 8 provide crucial system-level insights into deploying boosting classifiers at the IoT network edge.

Table 7. Real-Time Hardware Emulation Performance for 2-Class (Binary) Classification

Algorithm	Latency (ms/sample) @ 2GB	Latency (ms/sample) @ 4GB	Inference Rate (samples/sec) @ 2GB	Inference Rate (samples/sec) @ 4GB	Accuracy (VM)	Precision (VM)	Recall (VM)	F1-Score (VM)
AdaBoost	0.00638	0.01222	156,831.70	81,810.71	0.989	0.733	0.880	0.800
CatBoost	0.00406	0.00415	246,291.65	241,127.47	0.997	0.958	0.920	0.939
GBM	0.00147	0.00127	680,341.07	788,468.18	0.999	1.000	0.960	0.980
LGBM	0.00222	0.00205	451,373.12	486,905.17	0.994	0.880	0.880	0.880
XGB	0.00441	0.01221	226,971.27	81,914.43	0.997	0.923	0.960	0.941

Table 8. Real-Time Hardware Emulation Performance for 8-Class (Multi-Class) Classification

Algorithm	Latency (ms/sample) @ 2GB	Latency (ms/sample) @ 4GB	Inference Rate (samples/sec) @ 2GB	Inference Rate (samples/sec) @ 4GB	Accuracy (VM)	Precision (VM)	Recall (VM)	F1-Score (VM)
AdaBoost	0.00832	0.01527	120,259.19	65,499.16	0.911	0.831	0.911	0.869
CatBoost	0.00285	0.03507	350,698.52	233,879.89	0.986	0.984	0.986	0.983
GBM	0.00278	0.00784	359,661.13	127,605.90	0.990	0.987	0.990	0.988
LGBM	0.01012	0.01331	98,862.06	75,117.20	0.988	0.984	0.988	0.986
XGB	0.00662	0.01962	151,137.13	50,967.51	0.986	0.982	0.986	0.983

In the 2-Class (Binary) task (Table 7), GBM emerges as the absolute performance winner. GBM achieves a latency of only 0.00147 ms under the strict 2 GB RAM constraint, which further drops to an exceptional 0.00127 ms per sample when RAM is expanded to 4 GB. This translates to inference throughputs of 680,341.07 samples/sec and 788,468.18 samples/sec, respectively, while still yielding the highest accuracy of 0.999. This indicates that while GBM suffers from massive sequential offline training latency (66,248.01 ms, as shown in Table 5), its serialized execution path during active deployment is mathematically lightweight and highly optimized for single-threaded CPUs. An intriguing finding is identified when examining XGB and AdaBoost models. In the 2-Class scenario, expanding VM memory from 2 GB to 4 GB paradoxically increases latency and reduces throughput for both algorithms (e.g., XGB's throughput drops from 226,971.27 to 81,914.43 samples/sec, and AdaBoost drops from 156,831.70 to 81,810.71 samples/sec). This paradoxical drop in performance when RAM is increased cannot be explained solely by basic virtualization overhead. Instead, it suggests a deeper issue with how the operating system manages larger memory spaces on a severely restricted 1-core CPU. With more RAM, the virtual system might spend more time searching,

https://doi.org/10.31436/iiumej.v27i3.4534

allocating, or cleaning up memory spaces than executing model predictions, creating a bottleneck.

For the 8-Class task (Table 8), the complexity of multiclass decision boundaries (which process multiple class trees per packet) heavily constrains the single-core CPU bandwidth. CatBoost proves highly resilient under 2 GB of RAM, achieving a latency of 0.00285 ms (350,698.52 samples/sec throughput) and an F1-Score of 0.983. Similarly, under the 4 GB RAM configuration, almost all models experience this same memory-scaling paradox. For example, GBM's throughput drops to 127,605.90 samples/sec, and XGB drops to 50,967.51 samples/sec. This confirms that inside a limited 1-core environment, blindly adding larger memory allocation hinders rather than helps the processing speed. These results emphasize that memory bandwidth limits and cache management are highly critical factors in single-core IoT edge routers, making the selection of lightweight tree structures a primary design priority.

Table 9. Per-Class Performance Profiling of the Leading GBM Model (8-Class Task under 2 GB RAM Constraint)

Attack Class	Precision	Recall	F1-Score	Support (Instances)
Benign	0.9986	1.0000	0.9993	728
DDoS	1.0000	1.0000	1.0000	49
DoS	0.8077	0.9130	0.8571	23
Mirai	1.0000	1.0000	1.0000	184
Recon	0.7143	0.8333	0.7692	6
Spoofing	0.6000	0.5000	0.5455	6
Web	0.0000	0.0000	0.0000	1
BruteForce	0.0000	0.0000	0.0000	3
Weighted Average	**0.9865**	**0.9900**	**0.9881**	**1000**

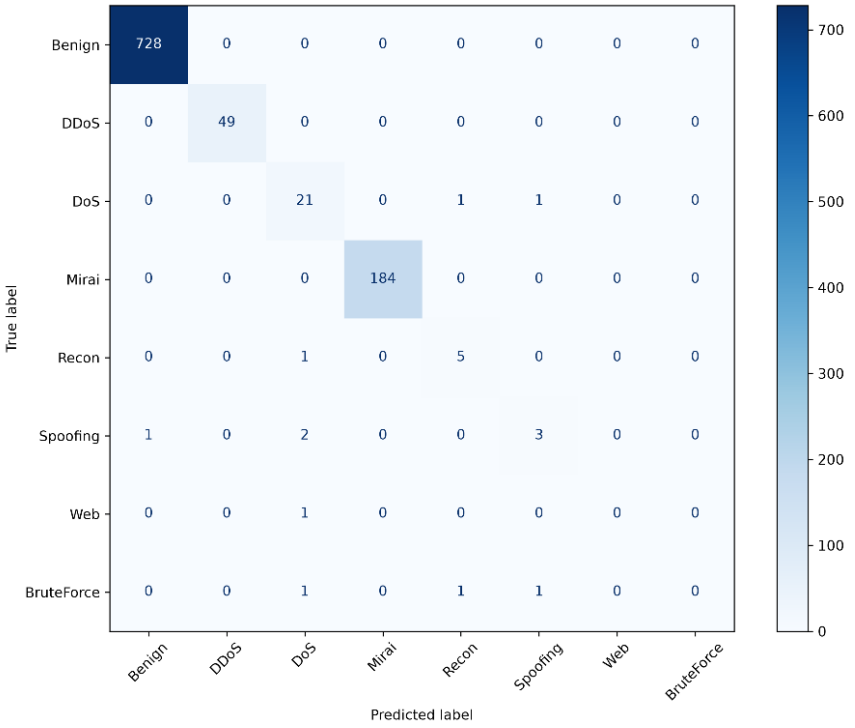

Figure 4. Confusion Matrix of the GBM Model (8-Class Task) under Edge VM Emulation, 2 GB RAM

https://doi.org/10.31436/iiumej.v27i3.4534

To address classification robustness and provide a granular view of the leading estimator, we extracted comprehensive per-class profiling for the GBM model on the 8-class task during active VM deployment with the highly restricted 2 GB RAM profile, as detailed in Table 9. While the aggregate weighted F1-score remains exceptionally high at 0.988, per-category metrics reveal the model's behavior under severe minority-class constraints.

As observed, GBM achieves perfect and near-perfect classification resilience for high-volume network events, securing an F1-score of 1.0000 for DDoS and Mirai attacks, and 0.9993 for Benign traffic. However, a notable degradation is evident in the extreme minority classes, specifically Web and BruteForce attacks, which yielded an F1-score of 0.000. This drop is mathematically tied to the critically low support sizes in the unbalanced testing subset (only 1 and 3 instances, respectively). This profiling is crucial for edge deployment: it demonstrates that while tree-based boosting handles macro-level network anomalies flawlessly under 1-core constraints, detecting micro-anomalies in unsampled, real-world traffic requires localized threshold tuning. To visually corroborate these classification boundaries and empirical misclassifications, Figure 4 presents the high-resolution Confusion Matrix captured directly from the 2 GB RAM Lubuntu VM inference phase.

5. CONCLUSION

This research conducted a rigorous, deployment-level empirical evaluation of tree-based boosting frameworks (GBM, XGB, LGBM, CatBoost, and AdaBoost) to secure resource-constrained IoT environments. By deploying an ablation methodology comprising Pure, Up, Down, and Hybrid sampling strictly after the train-test split, the study eliminated data-leakage vulnerabilities. Statistical validation ($p < 0.05$) confirmed that GBM yields peak performance under restricted edge-deployment conditions, achieving 99.9% accuracy in binary classification and 99.0% in multi-class classification during VM emulation. Edge-emulation profiles identified that sequentially structured frameworks like GBM provide the highest, most stable inference throughput inside severely memory-restricted (2 GB RAM) single-core architectures, handling over 680,000 samples per second.

A primary limitation of this study is the reliance on virtualized software hardware emulation (Lubuntu VM). While VM profiles accurately restrict memory and CPU allocation, they abstract away specific thermal throttling and physical instruction-set bottlenecks present in bare-metal microcontrollers (e.g., Raspberry Pi or ESP32). Future work will prioritize deploying these joblib artifacts directly onto physical Edge TPUs and examining power-consumption profiles. Furthermore, the memory-scaling paradox observed during the VM emulation—where increasing RAM from 2 GB to 4 GB unexpectedly reduced inference speed—remains an open issue. Because this study used a software-based VM, this anomaly may stem from how the virtual OS manages larger RAM allocations. Future research should deploy these models directly on physical, bare-metal IoT devices to confirm whether this memory bottleneck stems from the machine learning algorithm itself or is merely a side effect of the virtualization software.

DATA AVAILABILITY STATEMENT

The CICIoT2023 dataset supporting this research is publicly accessible via the Canadian Institute for Cybersecurity (CIC) repository. Analyzed subsets are available upon reasonable request.

ACKNOWLEDGEMENT

This research is funded by the Ministry of Education, Culture, Research, and Technology of Indonesia under grant numbers 003/SP2H/RT-MONO/LL4/2023 and 350/PNLT2/PPM/2023. The authors thank several anonymous reviewers and colleagues who shared their thoughts to improve the quality of this paper.

REFERENCES

[1] J. C. Cano, V. Berrios, B. Garcia and C. K. Toh, ”Evolution of IoT: An Industry Perspective,” in IEEE Internet of Things Magazine, vol. 1, no. 2, pp. 12-17, December 2018, doi: 10.1109/IOTM.2019.1900002.

[2] A. Imteaj, U. Thakker, S. Wang, J. Li and M. H. Amini, ”A Survey on Federated Learning for Resource-Constrained IoT Devices,” in IEEE Internet of Things Journal, vol. 9, no. 1, pp. 1-24, 1 Jan. 2022, doi: 10.1109/JIOT.2021.3095077.

[3] R. Ande, B. Adebisi, M. Hammoudeh, and J. Saleem, ”Internet of Things: Evolution and technologies from a security perspective”, Sustainable Cities and Society, vol. 54, p. 101728, 2020. doi: 10.1016/j.scs.2019.101728.

[4] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, ”Deep recurrent neural network for IoT intrusion detection system”, Simulation Modelling Practice and Theory, vol. 101, p. 102031, 2020. doi: 10.1016/j.simpat.2019.102031.

[5] K. Ponnusamy and N. Rajagopalan, ”Internet of Things: A Survey on IoT Protocol Standards”, in Progress in Advanced Computing and Intelligent Engineering, 2018, pp. 651–663. doi: 10.1007/978-981-10-6875-1 64

[6] S. Sathyadevan, K. Achuthan, R. Doss and L. Pan, ”Protean Authentication Scheme – A Time-Bound Dynamic KeyGen Authentication Technique for IoT Edge Nodes in Outdoor Deployments,” in IEEE Access, vol. 7, pp. 92419-92435, 2019, doi: 10.1109/ACCESS.2019.2927818.

[7] W. E. Mohammed Aziz Al Kabir and M. S. Sharif, ”Securing IoT Devices Against Emerging Security Threats: Challenges and Mitigation Techniques”, Journal of Cyber Security Technology, vol. 7, no. 4, pp. 199–223, 2023. doi: 10.1080/23742917.2023.2228053.

[8] P. Mishra, V. Varadharajan, U. Tupakula and E. S. Pilli, ”A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection,” in IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 686-728, Firstquarter 2019, doi:10.1109/COMST.2018.2847722.

[9] H. Binder, O. Gefeller, M. Schmid, and A. Mayr, ”The evolution of boosting algorithms”, Methods of Information in Medicine, vol. 53, no. 06, pp. 419–427, 2014. doi: 10.3414/ME13-01-0122.

[10] J. L. Leevy, J. Hancock, R. Zuech and T. M. Khoshgoftaar, ”Detecting Cybersecurity Attacks Using Different Network Features with LightGBM and XGBoost Learners,” 2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI), Atlanta, GA, USA, 2020, pp. 190-197, doi: 10.1109/CogMI50398.2020.00032.

[11] E. Ashraf, N. F. F. Areed, H. Salem, E. H. Abdelhay, and A. Farouk, ”FIDChain: Federated Intrusion Detection System for Blockchain-Enabled IoT Healthcare Applications”, Healthcare, vol. 10, no. 6, 2022. doi: 10.3390/healthcare10061110.

[12] S. Garg, V. Kumar, and S. R. Payyavula, "Identification of internet of things (IoT) attacks using gradient boosting: a cross dataset approach," Telematique, vol. 21, no. 1, pp. 6982–7012, 2022. Available: https://www.provinciajournal.com/index.php/telematique/article/view/965

[13] M. M. Khan and M. Alkhathami, ”Anomaly detection in IoT-based healthcare: machine learning for enhanced security”, Scientific Reports, vol. 14, no. 1, p. 5872, Mar. 2024. doi: 10.1038/s41598-024-56126-x.

[14] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, ”CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment”, Sensors, vol. 23, no. 13, 2023. doi: 10.3390/s23135941.

[15] M. M. Ahsan, M. A. P. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, ”Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance”, Technologies, vol. 9, no. 3, 2021. doi: 10.3390/technologies9030052.

[16] M. Bach, A. Werner, and M. Palt, ”The Proposal of Undersampling Method for Learning from Imbalanced Datasets”, Procedia Computer Science, vol. 159, pp. 125–134, 2019. doi: 10.1016/j.procs.2019.09.167.

[17] A. R. Hakim, K. Ramli, M. Salman, and E. R. Agustina, “Improving Model Performance for Predicting Exfiltration Attacks Through Resampling Strategies”, IIUMEJ, vol. 26, no. 1, pp. 420–436, Jan. 2025.

[18] M. Saied, S. Guirguis, and M. Madbouly, ”A Comparative Study of Using Boosting-Based Machine Learning Algorithms for IoT Network Intrusion Detection”, International Journal of Computational Intelligence Systems, vol. 16, no. 1, p. 177, Nov. 2023. doi:10.1007/s44196-023-00355-x.

[19] L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees. Belmont, CA: Wadsworth, 1984.

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York, NY: Springer, 2009.