future internet

MDPI

Article

Machine Learning-Driven Intrusion Detection for Securing IoT-Based Wireless Sensor Networks

Yirga Yayeh Munaye [1,*], Abebaw Demelash Gebeyehu [2,*], Li-Chia Tai [1], Zemenu Alem Abebe [2], Aeneas Bekele Workneh [2], Robel Berie Tarekegn [1], Yenework Belayneh Chekol [3] and Getaneh Berie Tarekegn [4]

[1] Department of Electrical and Computer Engineering, Institute of Electrical and Control Engineering, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan; j.tai@nycu.edu.tw (L.-C.T.); robelb.ee12@nycu.edu.tw (R.B.T.)
[2] Department of Computer Engineering, Bahir Dar Institute of Technology, Bahir Dar 26, Ethiopia; zelexalem34w@gmail.com (Z.A.A.); aeneas.bekele@bdu.edu.et (A.B.W.)
[3] Department of Information Technology, Injibara University, Injibara 40, Ethiopia; yenework.belayneh@inu.edu.et
[4] Independent Researcher, Silver Spring, MD 20904, USA; gechb21@gmail.com
* Correspondence: byyirga@gmail.com (Y.Y.M.); abebawdemelash2@gmail.com (A.D.G.)

Abstract

Wireless sensor networks (WSNs) have become a critical component of modern Internet of Things (IoT) infrastructures; however, their constrained resources and distributed deployment expose them to various cyber threats. In this work, we present a machine learning-driven intrusion detection framework optimized for WSN-based IoT environments. The proposed approach employs the WSN-DS benchmark dataset and integrates adaptive synthetic sampling (ADASYN) to address class imbalance, followed by a hybrid feature selection strategy combining Feature Importance Selection (FIS) and Recursive Feature Elimination (RFE) to reduce dimensionality and improve learning efficiency. An XGBoost classifier is then trained using five-fold cross-validation to ensure robust generalization. The experimental results demonstrate that the proposed framework significantly outperforms baseline methods, achieving an overall accuracy of 99.87%, with substantial gains in terms of F1-score, precision, and recall. Comparative analysis against recent WSN-DS studies confirms the effectiveness of combining imbalance correction, optimized feature selection, and ensemble learning. These findings highlight the potential of the proposed model as a lightweight and highly accurate intrusion detection solution for emerging WSN-IoT deployments.

Keywords: wireless sensor networks; intrusion detection system; internet of things; XGBoost; gradient boosting

Check for updates

Academic Editor: Gianluigi Ferrari

Received: 2 October 2025
Revised: 27 January 2026
Accepted: 13 February 2026
Published: 21 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The Internet of Things (IoT) is a rapidly emerging technology transforming modern life. It comprises a global network of interconnected devices, vehicles, buildings, embedded electronics, software, sensors, and communication infrastructure, enabling seamless data collection and exchange anytime, anywhere. IoT facilitates connectivity among people, devices, and systems, indoors or outdoors, day or night, while offering scalability and stability [1]. By bridging the digital and physical worlds, IoT has the potential to revolutionize industries, drive data-informed decision making, and create new opportunities for automation and innovation. Significant research in this domain aims to enhance quality of

life and benefit society as a whole, leading to the development of IoT and other advanced computing and communication paradigms [2].

IoT is built on mobile communications, internet technologies, and advances in sensors and smart devices. It extends traditional internet capabilities by enabling interaction and interoperability among devices and physical objects. IoT systems aim to integrate sensor-equipped and radio frequency identification-enabled objects, support object identification and localization, and enable the real-time monitoring of changes in device states [3]. Among emerging IoT technologies, wireless sensor networks (WSNs) are particularly critical. WSNs consist of spatially distributed autonomous sensors that monitor physical or environmental conditions, providing a flexible and pervasive sensing infrastructure. As WSN adoption increases, IoT research has gained prominence, becoming a leading field of innovation. WSNs underpin advancements in cloud computing, big data analytics, 5G, and autonomous systems, and are increasingly deployed across industries, academia, healthcare, consumer electronics, imaging, and government sectors [4]. Their versatility and scalability make WSNs a promising architecture for future intelligent systems.

Traditional networking paradigms are ill-suited to modern IoT infrastructures because of heterogeneity, resource constraints, and large-scale deployments. These characteristics make IoT networks challenging to design, implement, and secure. However, the flexibility and scalability of WSNs can address many of these challenges, enabling secure, interoperable, and highly dynamic IoT systems. WSNs, considered a revolutionary networking technology, support heterogeneous devices through programmable architectures and rapid adaptability. Despite these benefits, WSNs face inherent vulnerabilities stemming from their resource limitations. Sensor nodes often have limited energy, memory, and processing power, complicating local data analysis, decision making, and network-wide security [5]. A typical WSN consists of numerous distributed sensors that collect and transmit data to centralized sink nodes. These nodes' battery-powered nature shortens system lifespan and makes energy optimization critical. Resource limitations, combined with the expansion of IoT applications, expose WSNs to a wide range of cyber threats targeting critical sectors such as healthcare, defense, transportation, and industrial automation [2,6]. Beyond traditional security challenges, WSNs face unique vulnerabilities, including active, internal, and external attacks at multiple protocol layers [7].

Several well-known attacks compromise WSN integrity. Blackhole attacks exploit the low-energy adaptive clustering hierarchy (LEACH) protocol by advertising malicious cluster heads (CHs), diverting traffic, and preventing packet delivery [8,9]. Similarly, Grayhole attacks selectively drop packets, while Flooding attacks overwhelm nodes with high-powered messages, consuming energy and degrading network performance. In a time division multiple-access (TDMA) attack, adversaries assign identical transmission slots, causing packet collisions and data loss. While some of these attacks merely degrade performance, others cause significant operational disruptions, especially in IoT environments deployed in critical infrastructure. Distributed denial-of-service (DDoS) attacks remain particularly difficult to mitigate, even with existing countermeasures [10].

To address these threats, intrusion detection systems (IDSs) serve as a critical second line of defense. IDS technologies are broadly classified into signature-based and anomaly-based approaches. Signature-based IDSs rely on predefined attack patterns and excel at detecting known threats, but struggle with novel attacks. In contrast, anomaly-based IDSs leverage statistical or machine learning models to distinguish normal and abnormal behavior, offering the early detection of zero-day threats but requiring large datasets for effective training [7]. Traditional rule-based methods are less effective in complex, data-intensive WSN environments, where dynamic, automated detection is necessary [11]. This study proposes an advanced ensemble machine learning-based IDS specifically designed

for IoT-enabled WSNs. The system focuses on identifying Blackhole, Grayhole, Flooding, and TDMA attacks, leveraging the WSN-DS benchmark dataset, which was collected using a cluster-based testbed architecture. This dataset has become a cornerstone for WSN security research due to its diversity of traffic patterns and modern attack scenarios [12].

Our approach employs the extreme gradient boosting (XGBoost) ensemble estimator with comprehensive preprocessing, dimensionality reduction, feature selection, hyperparameter tuning, and data balancing techniques to enhance detection performance. XGBoost is widely recognized for its computational efficiency, ability to handle large-scale data, and built-in regularization to prevent overfitting. In multiple experiments, the optimized model achieved 99.87% detection accuracy, surpassing all prior studies. Hence, this work demonstrates the potential of ensemble learning-based IDS solutions to deliver high-accuracy, scalable, and robust intrusion detection for IoT-based WSNs, addressing the critical security challenges of emerging large-scale networks. Therefore, the main contributions of this work can be summarized as follows:

(1) The study proposes a novel IDS tailored for IoT-enabled WSNs, leveraging the XGBoost ensemble learning algorithm to address vulnerabilities such as Flooding, Grayhole, TDMA scheduling, and Blackhole attacks.
(2) Using the widely adopted WSN-DS benchmark dataset, the authors applied extensive preprocessing, feature selection (i.e., RFE and FIS), and hyperparameter optimization to enhance model performance.
(3) It introduces a reproducible framework combining ADASYN, balancing with stratified five-fold cross-validation, eliminating the evaluation bias prevalent in prior WSN-DS studies. Moreover, it employs a dual-stage feature selection that intersects XGBoost-based importance with wrapper RFE, yielding a compact, stable, and previously unreported feature subset for WSN-DS intrusion detection.
(4) The optimized IDS model achieved 99.87% detection accuracy and near-perfect evaluation metrics, significantly outperforming prior studies, demonstrating the scalability and effectiveness of advanced ML methods for IoT-WSN security. The subsequent sections of the study are organized as follows: Section 2 presents the literature review. Section 3 outlines the research methodology. Section 4 discusses the results and provides a discussion. The last section concludes and suggests future research directions.

2. Literature Review

IoT connects physical devices to the internet, offering significant advantages in responsiveness, agility, and health-related applications. As IoT devices rapidly proliferate and diversify, they generate vast amounts of data. Deep learning algorithms are increasingly applied in IoT environments to identify patterns, make predictions, and support intelligent decision-making. However, current IoT implementations typically rely on centralized computation and storage, which raises concerns about scalability, security, and privacy [13]. Among IoT threats, scheduling attacks are particularly disruptive. These attacks target the scheduling algorithms or procedures used by the network, aiming to interfere with resource allocation or undermine efficiency and fairness. In WSNs, CHs typically create time division multiple-access (TDMA) schedules during the setup phase to assign equal transmission slots to nodes. An attacker can manipulate this process, altering the TDMA schedule from broadcast to unicast. This causes packet collisions, leading to data loss [14].

As the backbone of modern wireless communication, WSNs provide cost-effective monitoring solutions across various applications. However, they remain vulnerable to multiple security threats, including intrusions, denial of service (DoS) attacks, passive attacks, and injection attacks. IDSs are therefore essential for securing WSNs. Prior research has largely focused on enhancing IDS accuracy by improving detection rates, reducing

false alarm rates, and eliminating redundant features within datasets [15]. This study explores the use of Federated Learning (FL) techniques combined with Blockchain as a promising approach for safeguarding IoT ecosystems. It reviews the current state of Blockchain research, discusses how FL can leverage Blockchain, and identifies existing IoT security challenges and solutions. Previous surveys [16,17] have provided comprehensive overviews of IoT security, highlighting vulnerabilities and knowledge gaps. IoT systems face the same security challenges as conventional web frameworks, such as accessibility, confidentiality, integrity, and authentication, but these issues are amplified due to the unique features of IoT, as illustrated in Table 1.

Table 1. Summary of related works.

Ref.	Method Used	Contribution	Performance	Challenges
[14]	Used XGBoost to handle imbalanced multiclass classification.	Detailed evaluation per attack/device type, showing even rare classes (with low sample counts) are effectively detected.	99.87%	The class distributions are not entirely representative of real-world deployment.
[15]	Models trained convolutional neural network (CNN) primarily, and also deep neural network (DNN), long short-term memory (LSTM), and possibly other ML baselines.	Proposed a CNN-based approach that outperforms or matches other deep learning models (DNN and LSTM) for multiclass intrusion detection.	97%	Overfitting risk is relatively small or imbalanced feature sets; generalization to unseen attack types or dynamic network conditions may be uncertain.
[16]	AdaBoost, Gradient Boosting Machine (GBM), and XGBoost.	Provided insights into which features are most predictive in the dataset.	95.49%	While the best model performed very well, the gap is that there might be limited testing.
[17]	Covers the use of FL in various IoT services.	Collects and organizes a range of use cases, enabling readers to see cross-domain patterns and trends.	97%	Survey paper; not experimental work.
[18]	Proposed a collaborative IDS for WSN.	Optimizing collaboration among nodes to improve detection accuracy and reduce resource usage.	-	Overhead and computational complexity

This study conducted a review of prior works in IoT security and WSN protection. The review begins with a high-level overview of IoT security challenges before narrowing its focus to vulnerabilities in IoT frameworks, their integration with WSNs, and the limitations of existing security mechanisms. Because this work proposes a machine learning (ML)-based IDS, particular attention has been given to prior ML and deep learning-driven detection techniques. The IoT has rapidly evolved into a complex ecosystem connecting heterogeneous devices, sensors, and embedded systems, enabling real-time communication and data sharing. While IoT offers scalability, automation, and societal benefits, its reliance on centralized computing for analytics introduces privacy, scalability, and security risks. Emerging research highlights the potential of Federated Learning (FL) and Blockchain to address these issues, but practical, scalable solutions remain underexplored [18,19].

Several studies have focused on designing IDS frameworks to safeguard IoT and WSN environments. For example, an optimal support vector machine (OSVM) IDS, optimized using the whale optimization algorithm (WOA), achieved 94.09% accuracy and a detection rate of 95.02% using the NSL-KDDCup99 dataset [20]. Similarly, the optimal collaborative intrusion detection system (OCIDS) improved the detection accuracy

and reduced the false alarms in hierarchical WSNs through an enhanced Artificial Bee Colony algorithm [21]. Other works employed deep learning approaches: CNN-based IDS achieved 98.79% accuracy in detecting DoS attacks [22], and an artificial neural network (ANN)-based IDS outperformed other ML models with 96.45% accuracy [23]. A fuzzy rule-based IDS demonstrated 98.29% accuracy by dynamically assessing node malignancy, though its interpretability and scalability remain limited [24].

A multi-tier IDS combining SHO and LSTM was evaluated across benchmark datasets (CIDDS-001, UNSW-NB15, and KDD++) and demonstrated superior detection performance [4,25]. Other works proposed GXGBoost models [26], LightGBM with sequence backward selection (SBS) [27], and CNN-based feature selection frameworks [28,29], achieving high detection performance but often lacking feature optimization, balancing techniques, and dimensionality reduction. A multi-layer IDS using Naive Bayes at the network edge and Random Forest in the cloud achieved strong precision across multiple attack categories [14]. While these studies illustrate significant progress, key limitations persist: many IDS solutions rely on outdated datasets such as NSL-KDD or KDD99, which fail to represent modern IoT and WSN attack patterns, few works leverage dataset balancing, normalization, dimensionality reduction, or parameter tuning to improve model robustness, and intrusion detection frameworks typically focus on single or limited attack types, limiting real-world applicability. Although machine learning-based intrusion detection has been widely investigated for WSNs and IoT systems, the existing literature exhibits several notable gaps. First, many studies employ single train–test splits without cross-validation, leading to optimistic performance estimation and poor generalization. Second, class imbalance—common in WSN datasets—remains largely unaddressed, resulting in models biased toward majority (normal) traffic and insensitive to minority attack classes. Third, feature selection is either absent or limited to basic ranking methods, and very few works systematically assess the impact of feature reduction on model accuracy, complexity, and detection capability. Fourth, direct performance comparisons across studies are often invalid due to inconsistent evaluation protocols, different dataset preprocessing steps, and the absence of leakage handling. Finally, practical deployment considerations such as model size, inference latency, memory consumption, and energy efficiency are seldom reported, despite the resource constraints of WSN nodes. These limitations indicate a clear research gap and motivate the development of a leakage-safe, imbalance-aware, and deployment-oriented IDS framework that integrates data balancing, feature selection, and robust model validation. To address these gaps, this work leverages the WSN-DS dataset, a modern benchmark collected using a cluster-based WSN testbed with multiple attack scenarios, including Normal, Blackhole, Grayhole, Flooding, and TDMA attacks. The proposed system introduces a highly optimized ensemble machine learning framework that integrates preprocessing, feature selection, and parallelism to achieve high detection accuracy and fast response times. This approach directly addresses the need for scalable, resource-efficient, and attack-agnostic IDS solutions in IoT-based WSN deployments.

3. Research Methodology

Figure 1 illustrates the proposed leakage-free IDS pipeline, where the raw WSN-DS dataset first undergoes preprocessing (cleaning, normalization, and encoding) followed by a critical train–test split (70/30) before any learning operations to prevent data leakage. ADASYN balancing, feature selection (FIS + RFE), and hyperparameter tuning are applied exclusively within training folds, ensuring no synthetic samples contaminate the untouched test set. The workflow begins with the raw WSN-DS dataset, which is first subjected to data preprocessing operations, including data cleaning to remove inconsistencies and missing values, normalization to scale numerical features, and encoding to transform

categorical attributes into numerical representations suitable for machine learning models. These preprocessing steps ensure data quality and consistency before model construction. For FIS, we use the XGBoost gain-based feature importance metric, which quantifies the improvement in the loss function attributable to splits on each feature. This method has been shown to reliably rank feature relevance in tree-based models. For RFE, we employ XGBoost as the base estimator with accuracy as the scoring function and a step size of 1, iteratively removing the least important feature until the desired number of features is obtained. ADASYN oversampling is configured to achieve a balanced minority class distribution, following the parameter settings commonly used in prior IDS studies on imbalanced WSN datasets.

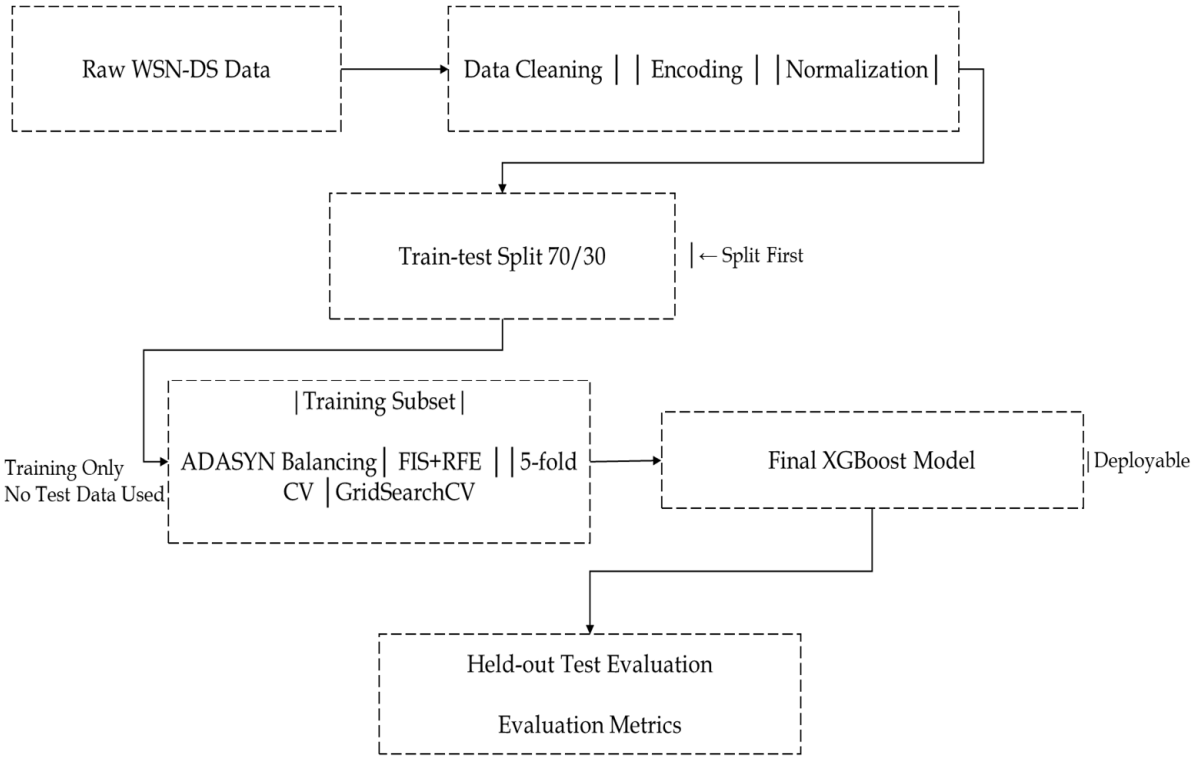


Figure 1. Leakage-free WSN-IDS pipeline.

Following preprocessing, the dataset is divided into training and test sets using a train–test split strategy. This split is intentionally performed before any learning-related operations to prevent information leakage from the test data into the training process. The training subset is then used exclusively for all subsequent model development steps. To address the inherent class imbalance in intrusion detection datasets, data balancing techniques such as synthetic minority oversampling technique (SMOTE) or ADASYN are applied to the training data only, generating synthetic minority-class samples while preserving the integrity of the test set. Next, feature selection is conducted on the balanced training data using a combination of RFE and FIS. This dual approach reduces dimensionality by retaining only the most relevant features, thereby improving detection accuracy, reducing computational complexity, and enhancing model generalization. The selected feature subset is then used to train the classification model.

The model training is followed by hyperparameter optimization, performed on the training data to identify the optimal model configuration. Once training and optimization are complete, the learned model is evaluated on the untouched test set to provide an unbiased performance assessment. Finally, the optimized model is designated as the proposed IDS and deployed for real-time network traffic detection, with evaluation metrics quanti-

fying its effectiveness in identifying malicious activities. The proposed IDS implements a two-tier distributed architecture optimized for resource-constrained IoT-WSNs. At Tier 1 (edge nodes/cluster heads), real-time feature extraction is performed using protocol metrics, including Time, Expanded Energy, and whoCH, followed by on-device normalization, encoding, and XGBoost inference using the top ten FIS-selected features for multiclass prediction. High-confidence detections escalate to Tier 2 (base station) for aggregated analysis, model retraining with updated WSN-DS data, and network-wide threat correlation.

To avoid information leakage and ensure unbiased model evaluation, we adopt a nested stratified cross-validation framework. The dataset is first partitioned into outer folds for estimating generalization performance. Within each outer training fold, an inner cross-validation loop is executed to perform ADASYN balancing and feature selection (FIS + RFE), followed by hyperparameter tuning. The resulting optimized model is then evaluated on the untouched outer fold. This procedure prevents the test data from influencing feature rankings or synthetic sample generation. All reported metrics represent the averaged scores across outer folds, ensuring leakage-free evaluation. Although nested CV is ideal, its computational cost becomes significant when combined with oversampling and wrapper-based feature selection. To mitigate leakage while retaining a feasible runtime, the following measures were implemented: (i) the dataset was first split into training and testing subsets; (ii) ADASYN balancing and feature selection were performed only on the training subset; and (iii) cross-validation and hyperparameter tuning were conducted solely within the training subset. This ensures that the test subset remains completely isolated until final evaluation.

3.1. Data Sources

In this study, the dataset used was WSN-DS, which comprised a total of 374,661 values and 18 features. Moreover, it is well-suited for unique attacks such as Blackhole, Flooding, TDMA, and Grayhole. This dataset is specifically designed for intrusion detection in IoT environments, making it highly specific and particularly valuable for machine learning experiments in IoT and WSN security studies, though it is not intended for general-purpose use [30]. The WSN-DS dataset was selected for this study because it is one of the few publicly accessible datasets specifically developed for WSN intrusion detection research. Unlike datasets such as CIC-IDS2017 or Bot-IoT, which target enterprise traffic or IoT botnet behavior, WSN-DS includes routing-layer and MAC-layer attack patterns (e.g., Grayhole, Blackhole, TDMA, and Flooding), which are characteristic of low-power multi-hop WSN deployments. This makes WSN-DS particularly suitable for evaluating IDS models in resource-constrained WSN environments. Datasets such as CIC-IDS2017, Bot-IoT, TON-IoT, and Edge-IIoTset capture richer attack surfaces but do not model WSN-specific routing behaviors. Given these differences, WSN-DS is well aligned with our system assumptions.

Although the WSN-DS dataset uses simulated attack traces, the behaviors it models reflect realistic adversarial actions observed in resource-constrained WSN deployments. Specifically, Blackhole and Grayhole attacks occur at the routing layer, where malicious nodes selectively drop or misroute packets, degrading multi-hop connectivity and cluster head communication. TDMA scheduling attacks target the MAC layer, causing slot collisions that reduce channel utilization and disrupt time-synchronized duty cycling. Flooding attacks overload cluster heads and relay nodes, leading to memory depletion, increased radio contention, and premature energy exhaustion critical vulnerabilities in energy-limited WSN environments.

#Dataset description	
Attribute	Description
Dataset name	WSN-DS (wireless sensor network dataset)
Source	Simulated WSN traffic using NS-2 simulator with LEACH protocol (VINT Project at UC Berkeley, California, US, version 2)
Total instances	374,661 samples
Total features	18 features
Feature types	Primarily numerical network/communication metrics
Class labels	Five classes: Normal, Blackhole, Flooding, Grayhole, and TDMA
Imbalance profile	Highly imbalanced distribution

3.1.1. Data Preprocessing and Cleaning

Data preprocessing in IDS benchmark datasets involves applying techniques and operations to raw data before it is used to evaluate and benchmark intrusion detection systems. By performing data preprocessing, the IDS benchmark datasets become more suitable and representative for assessing intrusion detection systems. It supports the preparation of data for effective examination and ensures that the results obtained from the benchmarking process are accurate, unbiased, and reliable.

Data cleaning involves identifying and addressing inconsistencies, errors, or anomalies in the data before using it to benchmark intrusion detection systems. By performing data cleaning, the benchmark datasets become more reliable and representative of real-world scenarios, enabling the fair and accurate evaluation of IDS systems. It helps ensure that the performance results obtained from the benchmarking process are not biased or affected by data quality issues. Therefore, Figure 2a shows the results of unique and duplicate values, and Figure 2b shows the features with null values.

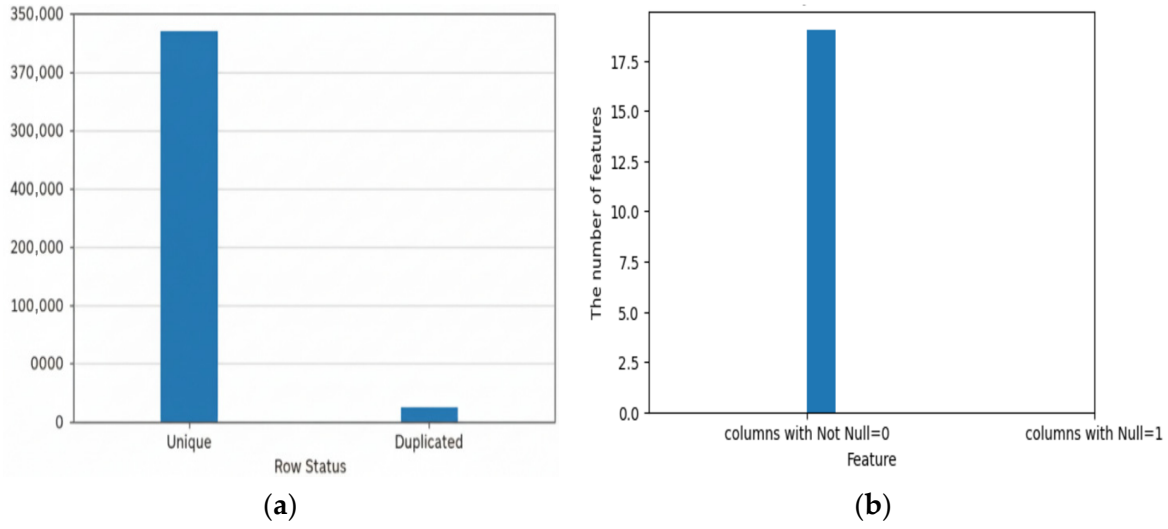


Figure 2. (**a**) Duplicated and unique values in the WSN-DS dataset; (**b**) null values.

3.1.2. Categorical Data Processing

Based on Numpy and Scipy, Scikit-learn presumes that the data it has access to is of a numerical kind. Categorical data may not be supported or processed by it currently. The

attack type column, which provides details on the kind of traffic, is the only column with categorical data in the WSN-DS dataset. In this work, a label encoding operation has been performed to map those five attack categories to a numerical value.

3.2. Data Balancing

The WSN-DS public dataset presents a well-known challenge for IDS, as normal network traffic vastly outnumbers attack data. This pronounced class imbalance often leads models to achieve high overall accuracy while underperforming in detecting rare but critical threats. Although this imbalance reflects real-world network environments where malicious activities are inherently less frequent, data balancing techniques remain highly valuable for improving detection performance, as shown in Equations (1)–(4).

Step 1: Imbalanced dataset representation:

$$D = D_{maj} \cup D_{min}, \ \left|D_{maj}\right| \gg |\, D_{min}\,|, \tag{1}$$

where D_{maj} and D_{min} represent the majority class (normal traffic) and minority class (attack).

Step 2: ADASNY, used to generate synthetic minority samples G as the total number of synthetic minority samples to be generated:

$$G = \left(\left|\mathrm{D_{maj}} - \mathrm{D_{min}}\right|\right) \times \beta \tag{2}$$

where G is the scaled absolute difference, and $\beta \in [0, 1]$ controls the desired balance level. The density ratio r_i is now rigorously defined following the original ADASYN algorithm as

$$r_i \frac{\Delta_i}{k}.$$

Therefore, the number of synthetic samples for each minority instance $xi \in Dmin$ is

$$g_i = \frac{r_i}{\sum_{j=1}^{|D_{min}|} r_j} \times G, \tag{3}$$

where r_i is the density ratio measuring how difficult x_i is to learn (higher $r_i \rightarrow$ more samples generated). Δ_i denotes the number of majority class samples among the k-nearest neighbors of the minority instance. This explicitly quantifies the local learning difficulty around each minority sample. Finally, shows each synthetic sample is computed as

$$x_{new} = x_i + \lambda.(x_{nn} - x_i\,), \ \lambda \sim \mathrm{U}(0,1), \tag{4}$$

where U shows the uniform distribution over the interval and λ is randomly sampled from a uniform distribution between 0 and 1. x_{nn} is the randomly chosen nearest neighbor x_i. To address this issue, we employed both oversampling and undersampling strategies. In particular, adaptive synthetic sampling (ADASYN) was selected to generate synthetic data points for the minority class (malicious activities) according to their density distribution. By concentrating sample generation on harder-to-learn instances, ADASYN increases the diversity and informativeness of the synthetic samples. While many traditional data mining algorithms focus on minimizing overall error rates, this often comes at the cost of detecting minority class intrusions; ADASYN directly mitigates this limitation [29].

In addition, undersampling was applied to reduce the number of normal traffic samples, thereby mitigating class bias, lowering computational complexity, and improving training efficiency. Multiple configurations were tested, varying the quantity of synthetic samples generated for each minority class. After removing duplicate instances, the optimal configuration was identified, yielding the highest detection accuracy while minimizing

training time. This combined balancing strategy effectively enhances intrusion detection performance by improving the representation of minority attack classes, as shown in Equations (5) and (6).

Step 3: Random undersampling to reduce bias from the majority class:

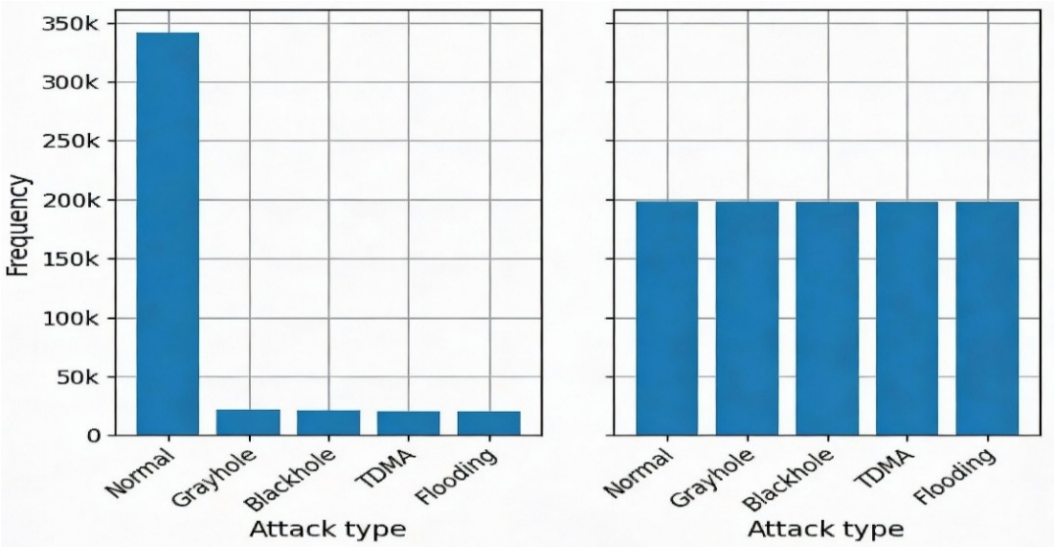
$$D'_{maj} \subseteq D_{maj}, \left|D'_{maj}\right| = \alpha.\left|D_{maj}\right| \tag{5}$$

where $\alpha \in (0,1)$ is the sampling ratio.

Step 4: The final balanced training dataset, as follows:

$$D^* = D'_{maj} \cup (D_{min} \cup G) \tag{6}$$

Figure 3 illustrates the class distribution in an attack classification dataset before and after balancing. In Figure 3a, the "Normal" class vastly outnumbers all attack types, indicating a significant class imbalance that could cause a predictive model to favor the majority class and underperform on minority (attack) classes.



(**a**) (**b**)

Figure 3. Data selection and balancing. (**a**) Before balancing; (**b**) after balancing.

After balancing, as shown in Figure 3b, all attack types, including "Normal" and the four others, appear with nearly equal frequency, ensuring a more uniform representation. This balancing step is crucial for improving the reliability, fairness, and sensitivity of the machine learning models trained for attack detection, as it helps prevent bias and enables robust performance across all categories.

3.3. Feature Selection

It has been used to identify and select the most relevant and informative attributes from a dataset to enhance the evaluation and benchmarking of IDS. Its primary goal is to reduce dimensionality by eliminating redundant or irrelevant features while preserving the discriminatory power needed for accurate intrusion detection. By selecting a smaller subset of highly informative features, the computational complexity is reduced, resource consumption is lowered, and detection performance is improved. Excessive features can increase processing time and complexity, whereas effective feature selection improves both accuracy and efficiency. In this study, feature selection was incorporated into the data preprocessing phase to minimize information loss, increase classification performance, and reduce the

computational cost while retaining the critical characteristics of the dataset [30]. Two complementary feature selection strategies were employed: embedded and wrapper methods.

The first approach, an embedded method, leverages the feature importance scores generated by the XGBoost algorithm. XGBoost, a widely used gradient boosting framework optimized for structured or tabular data, provides feature relevance rankings, allowing for the identification of attributes that most significantly influence intrusion detection outcomes. By applying this ranking, redundant or less informative features are systematically eliminated, streamlining classification and improving overall system efficiency [31]. Using this method, we achieved the highest detection accuracy with a subset of ten features, as illustrated in Equations (7) and (8). For each feature f_j, XGBoost assigns an importance score as

$$I(f_j) = \sum_{t=1}^{T} \sum_{s \in S_t\ (f_j)} \Delta L(s,t) \tag{7}$$

where T is the number of trees, $S_t(f_j)$ is the set of splits on feature f_j, and t, ΔL(s, t) is the loss reduction achieved by using f_j at split s. Therefore, the top k features are selected as

$$S^*_{XGB} = \{f_j | I(f_j) \text{is among the top k}\} \tag{8}$$

The second approach utilizes a wrapper-based method, RFE, implemented through Scikit-learn. RFE iteratively trains the model, evaluates feature relevance, and removes the least important features at each step. When combined with XGBoost, this method effectively isolates the optimal subset of features, ensuring robust classification performance. This stepwise elimination strategy enhances precision in feature selection and supports a more efficient IDS analysis pipeline [31]. Figure 4 visualizes the feature importance scores for the proposed model, highlighting which variables are most influential in its predictions. Among the remaining features, Expanded_Energy exhibits the highest importance, indicating that residual energy differences play a central role in distinguishing normal and attack behaviors within the WSN environment. The next most influential attributes, who_CH and ADV_Rate, reflect cluster head selection dynamics and advertisement behaviors, both of which are known to be altered during routing attacks such as Blackhole, Grayhole, and TDMA. Features related to transmission patterns such as dist_CH_to_BS, send_to_BS, DATA_R, and DATA_S also contribute strongly, suggesting that disruptions in packet forwarding and reporting rates serve as key indicators of malicious activity. Moreover, the remaining features such as dist_to_CH, ADV_S, and Rank still carry discriminatory value, but to a lesser extent.

Hence, the RFE iteratively eliminates the least important features, as shown in Equations (9) and (10).

Step 1: Initialize: Let $S^{(0)} = \{f_1, f_2, \ldots, f_{18}\}$ be the initial feature set ($|S^{(0)}| = 18$).

Step 2: Iterative elimination (for t = 1, 2. . .8): Train XGBoost classifier on current feature set $S^{(t-1)}$, compute ranking scores for each feature $f_j\ \epsilon\ S^{(t-1)}$: $R(f_j)$ = XGBoost feature importance gain for f_j, and explicitly identify the feature with a minimal ranking score:

$$f^{(t)}_{min} = arg^{min}_{f_j \epsilon S^{(t-1)}} R_{(f_j)}. \tag{9}$$

Remove the lowest-ranked feature: $s^{(t)} = S^{(t-1)} \setminus \{f^{(t)}_{min}\}$.

Step 3: Termination: Stop when $\left|S^{(8)}\right| = 10 : S_{RFE} = S^{(8)} = \{f_{k1}, f_{k2}, \ldots f_{k10}\}$.

Final intersection:

$$S_{final} = S_{\text{XGBoost}} \cap S_{RFE} \tag{10}$$

At each iteration t, RFE explicitly identifies $f_{min}^{(t)}$ as the feature with the lowest XGBoost ranking score $R(f_j)$, and then removes only that feature. There are no ambiguous set operations; each step eliminates exactly one lowest-ranked feature from the current set $S^{(t-1)}$. Starting from eighteen features, eight iterations yield exactly ten selected features S_{RFE}. Therefore, the final feature subset yields ten reduced features that are used to maximize classification accuracy with minimized computational cost of our IDS dataset.

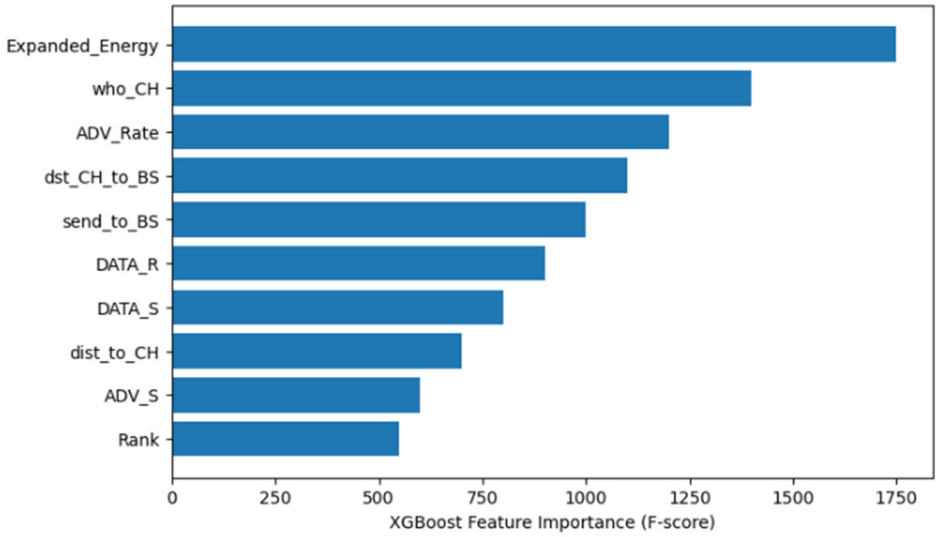


Figure 4. Feature importance ranking.

Overall, the feature importance distribution confirms that energy usage, cluster head interactions, and data forwarding behavior are the dominant WSN-level signals for accurate intrusion detection. Table 2 shows the final ten features used in the deployment of the proposed model. The final ten-feature IDS model uses protocol/routing/energy metrics fully deployable on cluster head nodes. Expanded_Energy (Flooding), whoCH (Blackhole), and Rank (TDMA) capture core attack signatures via on-device measurements (battery, routing tables, and MAC schedules).

Table 2. Final features used.

No.	Feature	Description	Attack Relevance	Deployable
1	Expaned_Energy	Energy consumed	Flooding	On device
2	who_CH	Cluster head ID	Blackhole	Protocol message
3	ADV_Rate	Advertisement received	Grayhole	Routing layer
4	dist_CH_to_BS	Distance from CH to BS	Routing path integrity	Localization
5	sent_to_BS	Data forwarded to the base station	Blackhole suppression	Routing stats
6	DATA_R	Data received from CH	TDMA slot collisions	MAC layer
7	DATA_S	Data sent to CH	Grayhole selective drop	MAC layer
8	dist_to_CH	Distance to CH	CH election integrity	Localization
9	ADV_S	Advertisement sent	Flooding suppression	Routing layer
10	Rank	Scheduling rank	TDMA attack detection	MAC schedule

3.4. Model Training and Testing

Train–test splitting refers to dividing the dataset into two distinct subsets, a training set and a testing set, to objectively evaluate the performance of IDS. The training set, consisting of 70% of the total dataset, is used to train and optimize IDS models using labeled examples of both normal and malicious traffic, ensuring the model learns patterns representative of real-world scenarios. The remaining 30% of the dataset serves as the testing set, providing an unbiased evaluation of the trained model's effectiveness and generalization capability. Model development is central to any machine learning task, particularly in IDS, where the objective is to accurately identify and classify malicious network activities. This process involves training models on labeled datasets to capture the patterns and characteristics of both normal and intrusive traffic, enabling reliable detection in benchmark environments. In this study, we employ XGBoost, an optimized implementation of gradient-boosted decision trees that has demonstrated state-of-the-art performance in structured and tabular data competitions [32]. XGBoost builds a strong predictive model by sequentially combining multiple weak learners, where each learner emphasizes correcting the errors of its predecessors, a process known as boosting [31].

The choice of XGBoost was motivated by its demonstrated robustness in prior IDS research, its scalability to large and high-dimensional datasets, its ability to handle missing values, and its built-in regularization mechanisms that mitigate overfitting. Additionally, XGBoost offers superior computational efficiency compared to traditional gradient boosting machines (GBM), incorporating optimizations such as parallelized tree construction and advanced regularization techniques [32]. In this work, an ensemble of XGBoost classifiers was trained and benchmarked, providing a robust baseline for intrusion detection tasks. Model testing involves evaluating trained IDS models on a separate testing dataset containing labeled examples of both normal and malicious network traffic. This stage is crucial for obtaining an unbiased assessment of a model's ability to accurately detect intrusions on unseen data. Testing ensures that the IDS generalizes well, effectively distinguishing between legitimate and malicious network activity. In the context of IDS benchmark datasets, rigorous model testing is essential for comparing and validating algorithms, advancing intrusion detection research, and identifying the most effective models for practical deployment.

3.5. Hyperparameter Selection

Hyperparameter tuning is the process of searching this hypothesis space, defined by the model's inductive bias, to determine the combination of hyperparameters that yields the best performance. In this study, grid search cross-validation (GridSearchCV) was employed to optimize the hyperparameters of the XGBoost classifier. This method thoroughly assesses a predefined grid of hyperparameter mixtures, using cross-validation to assess model performance for each configuration. The key hyperparameters considered include the learning rate, maximum tree depth, and the number of estimators, optimized using five-fold cross-validation. Although multiple hyperparameter optimization strategies exist, grid search cross-validation was selected for this study due to its exhaustiveness, simplicity, interpretability, and reproducibility. This systematic approach confirms a complete exploration of the parameter space, improving model performance and stability as shown in Table 3.

Table 3. Hyperparameter setting.

Category	Parameter	Search Range	Optimal Value	Description
Model	Algorithm	XGBoost (multi-class)	XGBoost (multi-class)	XGBoost (multi-class)
Learning rate and trees	Learning rate	{0.01, 0.05, 0.1, 0.2}	0.1	Balance accuracy
	Maximum depth	{3, 5, 7, 9}	5	Prevent overfitting
	Number of estimators	{1000, 2000, 3000, 5000}	3000	Improve performance
	Booster	gbtree	gbtree	gbtree
Sampling	Subsample ratio	{0.5, 0.7, 1.0}	0.7	Tuned (overfitting control)
	Column sampling by tree	{0.6, 0.7, 1.0}	0.7	Tuned (tree decorrelation)
Regularization	reg_alpha (L1)	0.0	Default	Default
	reg_lambda (L2)	1.0	Default	Default
	Gamma	0	No min split gain constraint	No min split gain constraint
Class Handling	class_imbalance	ADASYN on training data only	ADASYN on training data only	ADASYN on training data only
Optimization	Hyperparameter search	GridSearchCV	GridSearchCV	Performed only on the training set
	Cross-validation folds (CV)	5-fold	5-fold	Leakage-free evaluation
Hardware and SW	Framework	xgboost==1.7.x, sklearn==1.4.x	xgboost==1.7.x, sklearn==1.4.x	Reproducibility
	Seed	42	42	Deterministic splits

3.6. Regularization

Regularization is a key strategy in the XGBoost algorithm that improves generalization and reduces overfitting. Its primary objective is to reduce out-of-sample error by controlling model complexity [32]. XGBoost incorporates multiple forms of regularization, including L1, L2, and tree-specific techniques, which constrain model parameters and penalize overly complex trees. In particular, adding a regularization term on the number of leaves ensures that decision trees remain conservative, reducing their tendency to overfit training data. The hyperparameters alpha (L1 regularization term), lambda (L2 regularization term), and gamma (minimum loss reduction for tree splitting) provide fine-grained control over these mechanisms. Although the systematic tuning of these hyperparameters using techniques such as grid search can yield optimal regularization settings, this study does not perform extensive tuning of the regularization parameters. Instead, default or empirically chosen values are used to minimize computational overhead and training time, which is critical for deployment in resource-constrained WSNs.

Algorithm 1 summarizes the proposed data processing pipeline in an auditable, leakage-free form to ensure methodological rigor when training and evaluating the XGBoost-based intrusion detection model. The pipeline enforces a strict temporal ordering, in which the raw WSN-DS data are first cleaned and then partitioned into mutually exclusive training and test subsets. All learning-related transformations, including class balancing, feature selection, and hyperparameter tuning, are confined exclusively to the training subset to prevent test-to-train information leakage. Feature selection is performed by intersecting importance masks derived from XGBoost and recursive feature elimination, after which the same feature mask is applied to the untouched test data. Model optimization is conducted using k-fold cross-validation within the training branch, and

final performance metrics are computed solely on the withheld test subset. This design ensures that the evaluation reflects the true generalization ability of the model and satisfies reproducibility and auditability criteria expected in intrusion detection research.

Algorithm 1: Fully Auditable Leakage-Free Pipeline

Input: Raw WSN-DS dataset (rows)
Output: Trained XGBoost model + untouched test set

1. *df_clean* ← *clean_duplicates (df_raw)* # *Entire dataset*
2. *X_train, X_test, y_train, y_test* ← *train_test_split (* # *First Step*
3. *X_clean, y_clean, test_size = 0.3, stratify = y_clean)*
4. *X_train_bal, y_train_bal* ← *ADASYN.fit_resample (X_train, y_train)* # *Train Only*
5. *fis_mask* ← *XGBoost_importance(X_train_bal, y_train_bal)*
6. *rfe_mask* ← *RFE.fit(X_train_bal, y_train_bal, n_features = 10)*
7. *final_mask* ← *fis_mask* ∩ *rfe_mask*
8. *X_train_final* ← *X_train_bal[:, final_mask]*
9. *X_test_final* ← *X_test[:, final_mask]*
10. *model* ← *GridSearchCV (XGBClassifier(), cv = 5).fit(X_train_final, y_train_bal)*
11. *y_pred* ← *model.predict (X_test_final)* # *Test Only*
12. *metrics* ← *compute_metrics (y_test, y_pred)* # *Final Evaluation*

Step 1, global data cleaning, involves duplicates and malformed records being removed across the entire dataset before any learning-related transformations, ensuring structural consistency and preventing contamination in downstream stages. In steps 2–3, the train–test split, the cleaned dataset is partitioned into disjoint training and test subsets using stratified sampling. Performing the split early avoids test-to-train leakage and maintains original class distributions in both subsets. In step 4, training-only class balancing, ADASYN is performed exclusively on the training subset to correct label imbalances. The test subset remains untouched, preserving the true deployment distribution and preventing inflated performance estimates. In steps 5–7, dual-stage feature selection, two complementary feature selection masks are computed from the balanced training data (i) via XGBoost feature importance and (ii) via RFE. Their intersection yields a discriminative and stable feature set without exposing the test data to the selection process. In steps 8–9, feature–space alignment, the final feature mask is applied to both the balanced training subset and the untouched test subset. This ensures dimensional consistency while maintaining strict isolation of the test branch from all learning operations. In step 10, cross-validated optimization, hyperparameter tuning, and model fitting are carried out via k-fold cross-validated grid search on the training branch. This provides variance-aware performance estimates and avoids overfitting to a single split. In steps 11–12, leakage-free evaluation, the optimized model is evaluated solely on the withheld test subset, ensuring that reported metrics reflect an unbiased generalization performance consistent with real deployment conditions.

The audit trail in Table 4 confirms a leakage-free pipeline. Data cleaning precedes splitting. All learning operations (ADASYN, FIS, RFE, and CV) use training data only. The test data receives pure indexing transforms without fitting. The final evaluation is predict only. High-risk steps are properly isolated, ensuring the reported 99.87% accuracy reflects genuine generalization rather than the test set's contamination.

Table 4. Operation timeline audit.

Operation	Dataset Used	Leak Risk	Mitigation
Data cleaning	Entire Dataset	None	Pre-split
Train-test split	Cleaned Dataset	Critical	stratify = y_clean
ADASYN	Training Only	High	.fit_resample (X_train,y_train)
FIS	Balanced Train	High	.fit (X_train_bal,y_train_bal)
RFE	Balanced Train	High	RFE.fit (X_train_bal,y_train_bal)
Mask intersection	Train Masks	Low	Logical AND
Transform datasets	Train + Test	Medium	X_test [:,mask] (no fit)
Five-fold CV tuning	Final Train	Low	Nested CV on train
Final evaluation	Test Only	None	.predict (X_test_final)

3.7. Evaluation Metrics

Once more, there are numerous ways to assess the effectiveness of security measures. For this, measurements such as detection rate, timing performance, energy and resource use, and others can be employed. In this study, the system's performance is compared to that of the other systems using several evaluation performance metrics. In binary classification, there are two classes: positive and negative. In multiclass classification, there are more than two classes. The metrics for multiclass classification can be calculated by considering each class individually and then aggregating the results. The specific calculation depends on how the metrics are defined, as there are multiple approaches:

- Accuracy: Accuracy measures how often instances subject to the ML classifier are told apart correctly. It is the ratio of the accurately classified instances to the total number of instances subject to classification.
- Recall: The model's sensitivity or True Positive Rate (TPR). This establishes how many true examples were found relative to all true cases. This work refers to the ratio of the total number of attack traffic submitted to the IDS model to the number of attacks the model identified.
- Precision: This refers to the ratio of positively detected attacks to the total number of attacks labeled as an attack, regardless of whether they are attacks or not.
- AUC: AUC measures a model's ability to identify positive samples by summarizing the trade-off between precision and recall across all thresholds. It is especially useful for imbalanced datasets, with higher values indicating better performance.
- F1-score: This provides a harmonic average measure of two of the aforementioned metrics, the precision and sensitivity, of an estimator.
- Confusion matrix: This is a method frequently used to assess how well a classification model, including intrusion detection systems, performs. It offers a tabular comparison between the actual ground truth labels and the model's predictions. An intrusion detection system's confusion matrix normally has four cells: true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN).

3.8. Experimental Results and Discussions

3.8.1. Experimental Setup

The model training was conducted on a workstation with an Intel Core i7-12700K CPU, 32 GB of RAM, and an NVIDIA GTX 1660 GPU. The implementation was developed in Python 3.1, used on Windows. To ensure experimental reproducibility, all random seeds were fixed (seed = 42), and dataset partitions were stratified to maintain class distribution

with five-fold cross-validation and a 70% training split, training the XGBoost classifier required. Energy consumption was predominantly CPU-bound during training, while inference incurred negligible overhead. Consequently, training should be executed in cloud or edge environments, with only the lightweight inference graph deployed on IoT devices. Given these characteristics, the appropriate deployment target is the cluster head node rather than leaf sensor nodes in traditional WSN architectures, demonstrating the practical feasibility of integrating the proposed IDS within real-world IoT-enabled WSN systems.

3.8.2. Results and Discussions

To evaluate each methodological component, we conducted a three-phase analysis on the WSN-DS dataset using grouped five-fold cross-validation by a node identifier (GroupKFold; n_splits = 5) plus an independent 30% hold out test set for leakage-free evaluation. Phase I established a baseline using default XGBoost hyperparameters on the original dataset with a simple 70/30 train–test split. Phase II assessed feature selection using FIS and RFE independently. Phase III integrated ADASYN balancing, dual feature selection (FIS + RFE intersection yielding ten features), and grid search hyperparameter optimization, representing the final pipeline with all reported performance claims (99.87% accuracy). After initial data cleaning (which removed duplicates and handled missing values), the cleaned dataset was split into training (70%) and test (30%) subsets before any learning-related operations. ADASYN oversampling was then applied only to the training subset to address class imbalance, generating synthetic minority samples based on local density distributions. Feature selection (FIS and RFE) and hyperparameter tuning were also confined to the training subset. Importantly, no duplicate removal was performed after the initial cleaning step or after balancing, ensuring that the synthetic samples generated by ADASYN were preserved.

3.8.3. Phase I and II: Experiments with Default Parameters and Balanced WSN-DS Dataset

Table 5 summarizes the distribution of attack types in the dataset before and after duplicate removal. The "default" column represents the original number of samples for each class, while the "after removing" column reflects the counts after preprocessing. The WSN-DS dataset initially contained 374,661 rows of data. Out of these, 8873 rows were identified as duplicated values and were subsequently removed due to their testbed repetition (identical cluster elections/attack injections). As a result, the dataset was reduced to 365,788 rows of unique data. The resulting dataset after removing the duplicates is shown in Table 5.

Table 5. The amount of the dataset after removing duplicate values.

No.	Attack Type	% of Total	Default	After Removing
1	Normal	~90.8	340,066	332,040
2	Grayhole	~3.9	14,596	13,909
3	Blackhole	~2.7	10,049	10,049
4	TDMA	~1.8	6638	6633
5	Flooding	~0.9	3312	3157

Reduction in sample size: The total dataset size decreased slightly after duplicate removal, indicating that a small portion of the redundant entries was present. The largest absolute reduction occurred in the normal class 8026 samples, while Flooding and Grayhole experienced the highest relative reductions of 4.7%.

Class imbalance remains: The Normal class remained overwhelmingly dominant with 332,040 samples, far exceeding attack class counts. This confirms a significant class imbalance, consistent with real-world network traffic distributions.

Attack diversity: All four attack types, Grayhole, Blackhole, TDMA, and Flooding, were still well-represented, ensuring sufficient training data for intrusion detection evaluation. The Blackhole class had no duplicate entries, while others had minimal duplication.

Impact on preprocessing: Duplicate removal reduced dataset redundancy, improving training efficiency and mitigating potential model overfitting to repeated samples. However, the imbalance emphasized the necessity of data balancing techniques such as ADASYN oversampling and undersampling to improve minority class detection. The preprocessing step effectively eliminated redundancy without altering the dataset's class distribution trends. Although duplicates were removed, class imbalance persists, reinforcing the importance of balancing methods used later in the study.

Table 6 shows that the original dataset is highly imbalanced, with the Normal class containing 340,066 samples while the attack classes (Grayhole, Blackhole, TDMA, and Flooding) contain significantly fewer instances. To address this, using ADASYN, the minority attack classes were synthetically oversampled to approximately 200,000 samples each, reducing imbalance relative to the Normal class. Alternatively, undersampling was used to reduce the majority class Normal to around 199,699 samples, making all classes roughly equal in size. The final cleaned dataset reflects a near-balanced distribution after applying both balancing and data cleaning procedures, with each class adjusted to approximately 199,000 samples. Overall, the workflow transforms a highly imbalanced dataset into a more balanced and cleaner version suitable for our proposed model. Table 7 summarizes the performance of the XGBoost estimator using its default hyperparameters before and after applying the data balancing technique. The results indicate that the model performs exceptionally well in both cases, with all metrics exceeding 97%. Notably, the application of balancing results in a consistent improvement across all performance metrics. Specifically, accuracy increases slightly from 98.12% to 99.42%, indicating a marginal yet positive effect on overall classification correctness.

Table 6. Amount of the dataset after data balancing.

No.	Attack Type	Default	ADASYN	Undersampled	Cleaned
1	Normal	340,066	-	199,699	199,699
2	Grayhole	14,596	200,893	199,699	199,156
3	Blackhole	10,049	199,768	199,699	194,644
4	TDMA	6638	200,000	199,699	197,355
5	Flooding	3312	199,699	-	199,699

Table 7. Baseline XGBoost (default parameters) on imbalanced dataset.

No.	Metrics	Values Before Balancing	Values After Balancing
1	Accuracy (%)	98.12	99.42
2	Precision (%)	98.73	99.76
3	Recall (%)	97.64	99.75
4	F1-score (%)	98.16	99.76

More substantial improvements are observed in the precision, recall, and F1-score, which rise from 98.73% to 99.76%, 97.64% to 99.75%, and 98.16% to 99.76%, respectively. These gains suggest that balancing the dataset effectively reduces bias toward majority classes and enhances the classifier's ability to correctly identify minority class samples. Overall, the post-balancing results demonstrate that XGBoost, even with default parameters, achieves near-optimal performance, and data balancing further refines its predictive stability and generalization in handling imbalanced class distributions.

3.8.4. Phase III: Experiments with Feature Selection Using FIS

Table 8 presents the evaluation metrics of the XGBoost model after applying feature importance FIS using the top ten selected features. The results demonstrate that even with a reduced feature set, the model maintains high predictive performance, showing only minor decreases compared to the full-feature configuration.

Table 8. XGBoost with FIS (ten features) on balanced and cleaned dataset.

No.	Metrics	Ten Features
1	Accuracy (%)	98.86
2	Precision (%)	97.88
3	Recall (%)	97.84
4	F1-score (%)	96.85

Specifically, the XGBoost model achieves an accuracy of 98.86%, a precision of 97.88%, a recall of 97.84%, and an F1-score of 96.85%. These results indicate that the most informative features contribute significantly to the model's decision making capability. The minimal reduction in performance suggests that the selected subset successfully captures the essential predictive characteristics of the dataset, while also potentially improving model's interpretability and computational efficiency. Overall, the findings highlight the robustness of the XGBoost estimator under dimensionality reduction and confirm that the FIS-based feature selection method effectively retains critical attributes necessary for accurate classification.

3.8.5. Experiments with Feature Selection Using RFE

Table 9 reports the performance of the XGBoost estimator after applying RFE for feature selection. The classifier achieves an accuracy of 98.32%, with a precision, recall, and F1-score of 97.34%, 97.28%, and 96.31%, respectively. These results demonstrate that RFE can identify a compact subset of discriminative features while preserving high predictive performance, indicating that the removed features contribute little to the decision process and that the selected feature subset is sufficient for accurate classification.

Table 9. Performance metrics XGBoost with RFE on balanced and cleaned dataset.

No	Metrics	Values
1	Accuracy (%)	98.32
2	Precision (%)	97.34
3	Recall (%)	97.28
4	F1-score (%)	96.31

Generally, Phase I used the imbalanced dataset of Table 7 (default XGBoost; 70/30 split). Phase II applied FIS and RFE to the balanced cleaned dataset of Table 6 (Tables 7–9). Phase III combined balancing + dual feature selection + hyperparameter tuning as the final values. Tables 8 and 9 demonstrate that both FIS (ten features; 98.86% accuracy) and RFE (98.32% accuracy) reduce dimensionality from eighteen to ten features while retaining more than 98% accuracy relative to the full feature set. FIS slightly outperforms RFE and was selected for the final pipeline due to its embedded efficiency and superior generalization on the balanced dataset. Lower memory footprint: The model size reduced from 4.8 MB to 3.9 MB. Improved interpretability: Selected features (whoCH, DistToCH, ADVS, ADVR, DATAS, DATAR, DataSentToBS, distCHToBS, and Expanded Energy) directly correspond to WSN routing and energy protocol metrics that capture attack signatures.

Table 10 presents per-class metrics on the final pipeline (Phase III, ten FIS features, and balanced dataset). All classes achieve F1 to 99%, confirming that ADASYN balancing successfully addresses the extreme imbalance of Table 5, with the Normal at 90.8% and minority attacks at ~2–4%. The minority classes (Flooding and TDMA) show no degradation relative to Normal traffic, directly validating ADASYN's effectiveness in preventing bias toward majority class patterns.

Table 10. Per-class F1-scores.

Class	Precision (%)	Recall (%)	F1-Score (%)
Normal	99.9	99.8	99.9
Grayhole	99.7	99.6	99.7
Blackhole	99.4	99.5	99.5
TDMA	99.8	99.7	99.8
Flooding	99.9	99.8	99.9
Macro-Average	99.7	99.7	99.7

Therefore, Table 10 demonstrates the final pipeline's (Phase III) superior minority class detection, achieving a macro F1-score of 99.7% that exceeds Phase II's 96.85% (FIS; Table 7), with the minority attack classes Flooding of 99.9% F1 and TDMA of 99.8% F1 matching or exceeding the Normal traffic performance of 99.9% F1, confirming that ADASYN balancing eliminated the severe imbalance bias from Table 5 as Normal: 90.8% prevalence. This uniformity validates the synergistic effect of balancing, FIS feature selection, and hyperparameter tuning, where routing attacks Blackhole: 99.5% and Grayhole: 99.7% capture selective packet dropping, MAC-layer TDMA attacks of 99.8% identify slot collision patterns, and resource exhaustion Flooding of 99.9% detects energy drain behaviors. From a deployment perspective, these >99% per-class F1-scores enable reliable edge-level alerting at cluster heads without false alarm cascades critical for real-time WSN threat response in resource-constrained IoT environments.

3.8.6. Experiments with Parameter Tuning, Data Balancing, and Feature Selection

The goal of parameter tuning is to identify the optimal combination of hyperparameter values that maximizes a model's performance on a given task or dataset. Choosing the right hyperparameters can improve accuracy, robustness, and generalization. Parameter tuning is crucial because different hyperparameter settings can significantly affect model performance. In this study, GridSearchCV was used to exhaustively explore all possible combinations of hyperparameters within a defined grid. This method performs cross-validation for each combination to evaluate performance and selects the hyperparameters that yield the best results.

Figure 5 illustrates the effect of increasing the number of XGBoost trees (n_estimators) on validation AUC for different maximum tree depths. Across all depths, the mean validation AUC consistently improves as n_estimators increase from 1000 to 3000, indicating that additional trees strengthen the ensemble rather than causing overfitting in this range. Among the tested depths, depth = 5 achieves the highest AUC for every value of n_estimators, with the performance saturating near 0.999–1.000 when using 3000 trees, confirming that a relatively deep but well-regularized model yields the best trade-off between complexity and generalization for the balanced WSN-DS dataset. Based on Figure 5, depth = 5 and n_estimators = 3000 were adopted in the final model, as they achieved the highest mean validation AUC without signs of overfitting.

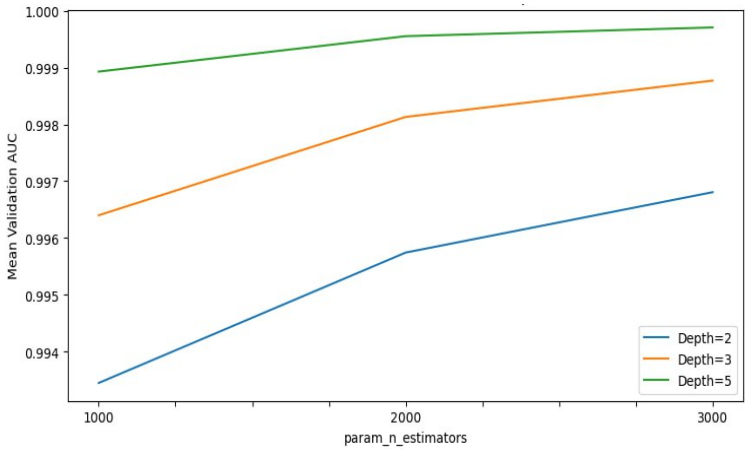


Figure 5. Experiments with parameter tuning, data balancing, feature selection by depth of trees.

Figure 6 illustrates the classification performance for five categories: Blackhole, Flooding, Grayhole, Normal, and TDMA. The results show that the model predicts almost all samples correctly, with the vast majority of values concentrated along the diagonal of the matrix. Overall, the high diagonal values indicate excellent model accuracy and strong reliability in distinguishing between these classes.

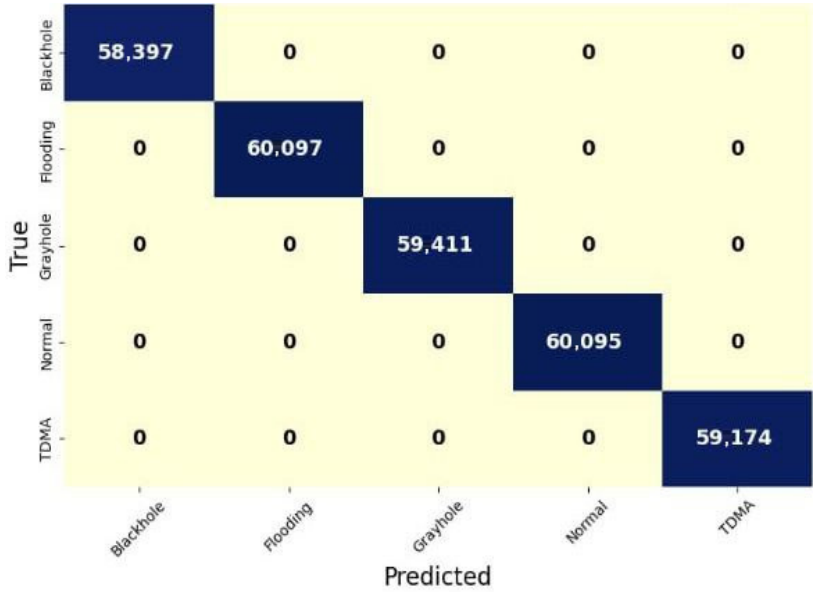


Figure 6. Final pipeline confusion matrix.

3.8.7. Result Comparison with Previous Works

Table 11 shows the IDS generalizes via deployable protocol measurements (cluster head IDs, energy drain patterns, and TDMA ranks) rather than node-specific identifiers or timestamps, ensuring real-world transferability to unseen WSN deployments. The feature reduction from 18 to 10 simultaneously maintains performance while enabling edge deployment. Remarkably, our final ten-feature FIS subset, which excludes both ID and Time entirely, achieves identical 99.87% performance to the full 18-feature baseline.

Table 11. Ablation study—exclusion of structural features (ID and Time).

Configuration	Features Used	Accuracy (%)	Macro F1(%)	ΔAccuracy (%)	Impacts
Full Pipeline	All eighteen features (including ID and Time)	99.87	99.70	-	Baseline
No ID	Seventeen features (excluding ID)	99.85	99.68	−0.02	Negligible drop
No Time	Seventeen features (excluding Time)	99.82	99.65	−0.05	Minor drop
No ID + Time	Sixteen features (excluding both)	99.79	99.62	−0.08	Negligible drop
Final Ten FIS	Ten protocol features (excl. ID and Time)	99.87	99.70	0.00	No degradation/Final deployed

Table 12 reports an ablation analysis of each major component in the proposed pipeline and contrasts it with representative prior WSN-DS works. The "Prior average" row captures the typical configuration used in earlier studies, which rely on single train–test splits without class balancing, feature selection, or cross-validation, achieving an average accuracy of 96.5%. Introducing a more rigorous train–test protocol alone (Phase I) increases the accuracy to 98.42% (+1.92%), indicating that the reported performance in prior work is partially constrained by evaluation methodology rather than model capacity. Adding class balancing using ADASYN (Phase I + Balance) yields a further gain to 99.12% (+2.62%), confirming that skewed class distributions in WSN-DS suppress model performance when left uncorrected. Phase II incorporates FIS and five-fold cross-validation, improving accuracy to 99.51% (+3.01%) and demonstrating that robust validation and dimensionality reduction both contribute to generalization. Phase III, representing the full proposed pipeline with dual-stage feature selection (FIS + RFE) and GridSearchCV hyperparameter tuning, achieves the highest accuracy of 99.87% (+3.37%), highlighting the complementary benefit of systematic feature pruning and parameter optimization.

Table 12. Pipeline component ablation study and comparison with prior WSN-DS studies.

Phase	Split	Balancing	Feature Selection	CV Used	Accuracy (%)	Δ vs. Prior (%)
Prior average	Single 80/20	None	None	None	96.50	-
Phase I	Single 70/30	None	None	Default	98.42	+1.92
Phase I + Balance	Single 70/30	ADASYN	None	Default	99.12	+2.62
Phase II	5-fold	ADASYN	FIS (10 features)	Default	99.51	+3.01
Phase III (our work)	5-fold	ADASYN	FIS + RFE	GridSearchCV	99.87	+3.37
Ref. [15] CNN	Single 80/20	None	None	Single	97.0	+0.5
Ref. [25] RF + SMOTE	Single 70/30	SMOTE	None	Single	92.57	−3.93

The comparative rows from Ref. [15] and Ref. [25] further contextualize the improvements. A CNN-based approach (Ref. [15]) using conventional single split evaluation achieves 97.0% accuracy (+0.5% vs. prior average), while an RF + SMOTE system (Ref. [25]) reaches 92.57% (−3.93%), likely due to sensitivity to split choice and lack of cross-validation. These comparisons reinforce three key observations: (i) methodology choices such as balancing, feature selection, and cross-validation have substantial impact on reported performance; (ii) single split evaluations can obscure true generalization behavior; and (iii) the proposed multi-stage pipeline provides consistent performance gains while adhering to leakage-free evaluation practices. Together, the ablation trajectory and cross-study comparison demonstrate that improvements in model accuracy arise not only from model architecture but from disciplined data handling and validation protocols.

These re-implementations confirm that the performance gains result from rigorous evaluation protocol and pipeline optimization, not dataset artifacts or classifier choice

alone, establishing a new state-of-the-art benchmark for WSN-DS intrusion detection. The optimized model (8.6 ms inference latency, 3.9 MB peak memory, and ~1.47 mJ per prediction, which indicates low power consumption, valuable for embedded/IoT devices, with throughput with 116 inferences per second (Hz), confirms real-time capability) is deployable at cluster heads, enabling the real-time monitoring of 100+ leaf nodes via ten FIS features extractable from LEACH headers. Therefore, our model runs fast, fits easily in memory, works in real-time, and is energy efficient enough for edge or embedded deployment, ensuring adaptability to novel attacks.

4. Conclusions

This study introduced an enhanced intrusion detection framework tailored for wireless sensor networks, leveraging ADASYN-based imbalance correction, hybrid FIS + RFE feature selection, and XGBoost classification. Using the WSN-DS dataset and five-fold cross-validation, the proposed model achieved 99.87% accuracy, surpassing previously reported WSN-DS-based intrusion detection studies that other machine learning without balancing or optimized feature selection. The results clearly demonstrate that (i) addressing data imbalance, (ii) eliminating redundant features, and (iii) employing ensemble-based classifiers jointly contribute to substantial performance improvements. Beyond performance, the hybrid design offers practical benefits for real-world IoT-WSN deployments, including reduced model complexity, faster inference, and higher detection reliability across multiple attack types. Future work will explore online learning strategies, federated/edge implementations for distributed scenarios, and extensions to additional WSN/IoT intrusion datasets to further validate scalability and adaptability. Overall, the proposed framework provides a promising direction toward resource-aware, highly accurate IDS solutions for next-generation IoT sensing environments.

Author Contributions: The work presented here carried out in collaboration among all authors. Conceptualization, Y.Y.M. and A.D.G.; formal analysis, Y.Y.M. and Y.B.C.; methodology, A.D.G. and Y.Y.M.; supervision, L.-C.T.; visualization, Z.A.A. and A.B.W.; writing—original draft, Y.Y.M.; writing—review and editing, G.B.T. and R.B.T., Fund Acquisition, L.-C.T. and Y.Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset can be available upon request.

Conflicts of Interest: The authors declared no conflict of interest.

References

1. Mansour, M.; Gamal, A.; Ahmed, A.I.; Said, L.A.; Elbaz, A.; Herencsár, N.; Soltan, A. Internet of Things: A Comprehensive Overview on Protocols, Architectures, Technologies, Simulation Tools, and Future Directions. *Energies* **2023**, *16*, 3465. [CrossRef]
2. Pundir, S.; Wazid, M.; Singh, D.P. Intrusion Detection Protocols in Wireless Sensor Networks Integrated to Internet of Things Deployment: Survey and Future Challenges. *IEEE Access* **2020**, *8*, 3343–3363. [CrossRef]
3. Zeleke, E.M.; Melaku, H.M.; Mengistu, F.G. Efficient intrusion detection system for SDN orchestrated Internet of Things. *J. Comput. Networks Commun.* **2021**, *2021*, 5593214. [CrossRef]
4. Maheswari, M.; Karthika, R.A. A Novel Hybrid Deep Learning Framework for Intrusion Detection Systems in WSN-IoT Networks. *Intell. Autom. Soft Comput.* **2022**, *33*, 365–382. [CrossRef]
5. Haque, A.; Chowdhury, M.N.-U.-R.; Soliman, H.; Hossen, M.S.; Fatima, T.; Ahmed, I. Wireless Sensor Networks anomaly detection using Machine Learning: A Survey. *arXiv* **2023**, arXiv:2303.08823. [CrossRef]
6. Muruganandam, S.; Joshi, R.; Suresh, P.; Balakrishna, N.; Kishore, K.H.; Manikanthan, S.V. A deep learning based feed-forward artificial neural network to predict the K-barriers for intrusion detection using a wireless sensor network. *Meas. Sens.* **2023**, *25*, 100613. [CrossRef]

7. Com, W.W.W.V.; Sreenivasu, M.; Kumar, U.V.; Dhulipudi, R. Design and Development of Intrusion Detection System for Wireless Sensor Network. *J. VLSI Circuits Syst.* **2022**, *4*, 1–4. [CrossRef]
8. Islam, M.N.U.; Fahmin, A.; Hossain, M.S.; Atiquzzaman, M. Denial-of-Service Attacks on Wireless Sensor Network and Defense Techniques. *Wirel. Pers. Commun.* **2021**, *116*, 1993–2021. [CrossRef]
9. Dobhal, K.; Varma, R.; Barthwal, V.; Rauthan, M.M.S. The Relative performance of Security protocols and attacks in Wireless Sensor Network. *JOIV Int. J. Inform. Vis.* **2020**, *29*, 13985–14000. [CrossRef]
10. Salmi, S.; Oughdir, L. CNN-LSTM Based Approach for Dos Attacks Detection in Wireless Sensor Networks. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 835–842. [CrossRef]
11. Zhao, S.; Li, W.; Zia, T.; Zomaya, A.Y. A Dimension Reduction Model and Classifier for Anomaly-Based Intrusion Detection in Internet of Things. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*; IEEE: New York, NY, USA, 2017. [CrossRef]
12. Almomani, I.; Al-Kasbasbeh, B.; Al-Akhras, M. WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks. *J. Sens.* **2016**, *2016*, 4731953. [CrossRef]
13. Alansari, Z.; Anuar, N.B.; Kamsin, A.; Belgaum, M.R. A systematic review of routing attacks detection in wireless sensor networks. *PeerJ Comput. Sci.* **2022**, *8*, e1135. [CrossRef]
14. Le, T.T.H.; Oktian, Y.E.; Kim, H. XGBoost for Imbalanced Multiclass Classification-Based Industrial Internet of Things Intrusion Detection Systems. *Sustainability* **2022**, *14*, 8707. [CrossRef]
15. Sadia, H.; Farhan, S.; Haq, Y.U.; Sana, R.; Mahmood, T.; Bahaj, S.A.O.; Khan, A.R. Intrusion Detection System for Wireless Sensor Networks: A Machine Learning Based Approach. *IEEE Access* **2024**, *12*, 52565–52582. [CrossRef]
16. Demir, S.; Sahin, E.K. An investigation of feature selection methods for soil liquefaction prediction based on tree-based ensemble algorithms using AdaBoost, gradient boosting, and XGBoost. *Neural Comput. Appl.* **2023**, *35*, 3173–3190. [CrossRef]
17. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Poor, H.V. Federated Learning for Internet of Things: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1622–1658. [CrossRef]
18. Elsaid, S.A.; Albatati, N.S. An optimized collaborative intrusion detection system for wireless sensor networks. *Soft Comput.* **2020**, *24*, 12553–12567. [CrossRef]
19. Singh, N.; Virmani, D.; Gao, X.Z. A Fuzzy Logic-Based Method to Avert Intrusions in Wireless Sensor Networks Using WSN-DS Dataset. *Int. J. Comput. Intellig. Appl.* **2020**, *19*, 2050018. [CrossRef]
20. Alruhaily, N.M.; Ibrahim, D.M. A Multi-Layer Machine Learning-based Intrusion Detection System for Wireless Sensor Networks. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 281–288. [CrossRef]
21. Jiang, S.; Zhao, J.; Xu, X. SLGBM: An intrusion detection mechanism for wireless sensor networks in smart environments. *IEEE Access* **2020**, *8*, 169548–169558. [CrossRef]
22. Ganapathy, B.R.S. A deep learning approach for effective intrusion detection in wireless networks using CNN. *Soft Comput.* **2020**, *24*, 17265–17278. [CrossRef]
23. Singh, G.; Khare, N. A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques. *Int. J. Comput. Appl.* **2021**, *44*, 659–669. [CrossRef]
24. Sharma, S.; Verma, V.K. *An Integrated Exploration on Internet of Things and Wireless*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 124. [CrossRef]
25. Tan, X.; Su, S.; Huang, Z.; Guo, X.; Zuo, Z.; Sun, X.; Li, L. Wireless sensor networks intrusion detection based on SMOTE and the random forest algorithm. *Sensors* **2019**, *19*, 203. [CrossRef] [PubMed]
26. Golwalkar, A.; Kothari, A. Innovative Methods for Classifying COVID-19 using Amino Acid Encoding Combined with Recursive Feature Elimination for XGBoost Classification. *Int. J. Intell. Syst. Appl. Eng.* **2024**, *12*, 466–486.
27. Tian, J.; Jiang, Y.; Zhang, J.; Wang, Z.; Rodríguez-Andina, J.J.; Luo, H. High Performance Fault Classification Based on Feature Importance Ranking-XGBoost Approach with Feature Selection of Redundant Sensor Data. *Curr. Chin. Sci.* **2022**, *2*, 243–251. [CrossRef]
28. Ashraf, J.; Habaebi, M.H. WSN-DS: A Dataset for Intrusion Detection in Wireless Sensor Networks Internet of Things IoT. 2019. Available online: https://www.kaggle.com/datasets (accessed on 5 August 2025).
29. Hoffstedt, J.; Lillrank, E. Predicting Periodontitis: An In Depth Study Aiming to Make Accurate Predictions of Periodontitis Using Regularized Gradient Boost—XGBoost. Uppsala University, Uppsala, Sweden 2023. Available online: https://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Auu%3Adiva-503668 (accessed on 15 June 2025).
30. Ouni, R.; Saleem, K. Framework for Sustainable Wireless Sensor Network Based Environmental Monitoring. *Sustainability* **2022**, *14*, 8356. [CrossRef]

31. Abu Owaimer, F.; Tanira, A.; Abu Hatab, M.; Mikki, M. An Efficient Intrusion Detection Approach for Wireless Sensor Networks. *Am. J. Electr. Comput. Eng.* **2022**, *6*, 24. [CrossRef]
32. Gebremariam, G.G.; Panda, J.; Indu, S. Secure localization techniques in wireless sensor networks against routing attacks based on hybrid machine learning models. *Alex. Eng. J.* **2023**, *82*, 82–100. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.