Article

A Multi-Dimensional Evaluation Framework for IoT Intrusion Detection: Balancing Accuracy, Efficiency, and Real-World Deployment Constraints

Abstract

The rapid proliferation of Internet of Things (IoT) devices has introduced unprecedented security vulnerabilities, with botnet attacks representing persistent threats to network infrastructure. While machine learning-based intrusion detection systems (IDS) show promise in laboratory settings, real-world deployment faces challenges including extreme class imbalance, resource constraints, and false alarm minimization. Traditional evaluation approaches focus primarily on accuracy, overlooking deployment factors such as execution time, memory consumption, and false alarm rates. This study introduces a comprehensive multi-dimensional evaluation framework that assesses IDS performance across 14 distinct metrics, including two novel composite scores, Efficiency Score and Deployment Score, for quantifying deployment readiness and operational efficiency. Applied to the BoT-IoT dataset (3.6 million flows, 1:7,682 imbalance), we evaluated six algorithms using six feature selection methods and a novel two-stage balancing approach combining undersampling and SMOTE. Critically, all experiments were conducted within a 12GB RAM constraint to reflect realistic resource limitations in edge computing and IoT gateway deployments. Random Forest, XGBoost, and Decision Tree achieved 99.97% accuracy with zero false positives on 733,705 test samples, maintaining practical training times (0.06–1.98 s) and memory footprints (0.07–2.00 MB). Our framework provides use-case-specific recommendations: XGBoost for resource-constrained devices (0.07 MB), Decision Tree for real-time applications (0.04 s prediction), and Random Forest for balanced deployment (0.9999 deployment score). This framework enables IoT security practitioners to make informed, context-aware model selections aligned with specific deployment constraints—whether prioritizing minimal memory footprint for edge sensors, ultra-low latency for real-time prevention systems, or rapid retraining for adaptive security—thereby accelerating the transition from laboratory research to operational IoT security solutions.

Keywords: intrusion detection system; Internet of Things; machine learning; botnet detection; feature selection; class imbalance; multi-dimensional evaluation; deployment metrics; XGBoost; Random Forest

1. Introduction

The exponential growth of Internet of Things (IoT) devices is projected to exceed 75 billion by 2025 [1], this has fundamentally transformed connectivity paradigms while simultaneously creating an expanding attack surface for cyber threats. IoT ecosystems, characterized by resource-constrained devices, heterogeneous protocols, and limited security

implementations, have become prime targets for sophisticated cyberattacks, particularly botnet-based intrusions [2]. Unlike traditional network environments, IoT networks face unique security challenges: computational limitations preclude the deployment of conventional security solutions, the massive scale of data generation demands efficient processing mechanisms, and the critical nature of many IoT applications (e.g., healthcare, industrial control) necessitates minimal false alarm rates to maintain operational continuity [3].

Intrusion Detection Systems (IDS) employing machine learning (ML) techniques have emerged as a promising defense mechanism, demonstrating remarkable accuracy in identifying malicious network behavior [4]. Recent studies have reported detection accuracies exceeding 99% on benchmark datasets such as BoT-IoT, UNSW-NB15, and NSL-KDD [2,5]. However, a critical examination of existing literature reveals a significant gap between laboratory performance and real-world deployability. Current research predominantly focuses on maximizing accuracy metrics—often measured through precision, recall, F1-score, and Matthews Correlation Coefficient (MCC)—while largely overlooking essential operational considerations including execution time, memory consumption, false alarm rates, and scalability to resource-constrained devices [6].

This accuracy-centric evaluation paradigm presents three fundamental limitations in the context of IoT security. First, *extreme class imbalance*—where normal traffic vastly outnumbers attack instances—can yield artificially inflated accuracy scores that mask poor minority class detection [7]. The BoT-IoT dataset, for instance, exhibits an imbalance ratio of 1:7,682 between normal and attack samples, yet many studies report high accuracy without adequately addressing the balancing methodology or its computational implications. Second, *computational feasibility* remains underexplored; while a Random Forest model may achieve 99.9% accuracy, its multi-gigabyte memory footprint and multi-hour training time render it impractical for edge deployment or continuous learning scenarios [8]. Third, *false alarm rates*—critical for operational acceptance—receive insufficient attention despite their direct impact on system usability and analyst workload [9].

The research community has made substantial progress in developing ML-based IDS for IoT environments. Soe et al. [10] proposed a DDoS detection system using Artificial Neural Networks with SMOTE for data balancing, achieving notable accuracy on the BoT-IoT dataset. Alissa et al. [11] demonstrated the effectiveness of machine learning algorithms on the UNSW-NB15 dataset, with Decision Tree models reaching 94% accuracy. Feature selection approaches have been extensively investigated: Maniriho et al. [12] employed hybrid Information Gain and Gain Ratio methods achieving 99.95% accuracy for DoS detection on IoTID20, while Baich et al. [13] compared Pearson correlation and Fisher score techniques on NSL-KDD, reporting 99.26% accuracy with Decision Tree. However, these studies predominantly report standard ML metrics without comprehensive analysis of deployment feasibility, resource consumption, or practical trade-offs between accuracy and efficiency [14]. Recent advances in federated learning for IoT security [65–69] and explainable AI for intrusion detection [71] have further highlighted the need for deployment-centric evaluation frameworks that balance accuracy with operational constraints.

Several researchers have begun addressing specific aspects of deployment readiness. Al-Ambusaidi et al. [15] developed an ML-enabled IDS for IoT networks, evaluating accuracy and sensitivity across multiple datasets. Idouglid et al. [16] proposed an anomaly detection model for Industrial IoT using PCA and Random Forest, achieving 98.9% accuracy with consideration of detection rate and false alarm rate. Li et al. [17] compared feature selection versus feature extraction methods on TON-IoT, revealing that feature extraction yielded better accuracy with lower sensitivity to feature changes. Despite these advances, no comprehensive framework exists that systematically evaluates IDS performance across multiple dimensions—accuracy, efficiency, resource consumption, and

operational metrics—while providing actionable recommendations for different deployment scenarios [18]. Emerging work on graph-based federated learning [70] and hybrid deep learning architectures [72] demonstrates the evolving complexity of IDS research, reinforcing the importance of comprehensive evaluation methodologies.

The present work addresses this critical gap by introducing a multi-dimensional evaluation framework that extends beyond traditional accuracy metrics to encompass real-world deployment considerations. Our framework evaluates IDS models across 14 distinct metrics organized into three categories: *traditional ML metrics* (accuracy, precision, recall, F1-score, MCC), *deployment metrics* (training time, prediction time, inference latency, memory footprint, false alarm rate), and *composite scores* (efficiency score, deployment score). We introduce two novel metrics: the *Efficiency Score*, quantifying the accuracy-to-time ratio, and the *Deployment Score*, a weighted composite measure balancing accuracy, false alarm rate, detection rate, training efficiency, and memory consumption. This holistic approach enables practitioners to make informed decisions based on specific operational requirements rather than relying solely on accuracy maximization.

To demonstrate the framework's utility, we conducted comprehensive experiments on the BoT-IoT dataset [2], comprising 3.6 million network flows with 46 features across multiple attack categories (DDoS, DoS, Reconnaissance, Theft). The dataset presents two significant challenges: extreme class imbalance (99.99% attacks, 0.01% normal) and high dimensionality. We addressed these through a novel two-stage balancing approach—combining random undersampling and SMOTE—that reduces computational overhead while maintaining synthetic sample quality, and through systematic evaluation of six feature selection methods (ANOVA, Fisher Score, Lasso, Ridge, RFE, Forward Selection) to identify optimal dimensionality reduction strategies [19,20].

Six state-of-the-art machine learning algorithms were evaluated: Random Forest, Decision Tree, Naive Bayes, K-Nearest Neighbors, Logistic Regression, and XGBoost. These algorithms represent diverse learning paradigms—ensemble methods, decision-based approaches, probabilistic models, and instance-based learning—enabling comprehensive performance comparison across different computational profiles [21]. Critically, all experiments were conducted within a 12GB RAM constraint, reflecting realistic resource limitations in edge computing and IoT gateway deployments [22].

The principal contributions of this work are fourfold:

1. **Multi-dimensional Evaluation Framework**: A comprehensive framework incorporating 14 metrics across traditional ML performance, deployment feasibility, and operational efficiency, enabling holistic IDS assessment beyond accuracy-centric approaches.
2. **Novel Composite Metrics**: Introduction of Efficiency Score and Deployment Score, providing quantitative measures of real-world deployability that balance multiple performance dimensions through weighted aggregation.
3. **Two-stage Balancing Methodology**: A computationally efficient approach to extreme class imbalance (1:7,682 ratio) combining random undersampling and SMOTE, reducing training time by 10–100× compared to pure SMOTE while maintaining model performance.
4. **Use-case-specific Recommendations**: Empirical identification of optimal models for distinct deployment scenarios—resource-constrained devices (XGBoost: 0.07 MB), real-time systems (Decision Tree: 0.04s prediction), balanced deployment (Random Forest: 99.97% accuracy, zero false positives)—with comprehensive performance characterization across 733,705 test samples.

The remainder of this paper is organized as follows: Section 2 reviews related work in ML-based IDS, feature selection techniques, and evaluation methodologies. Section 3

details the proposed multi-dimensional evaluation framework, dataset characteristics, preprocessing pipeline, feature selection methods, and experimental configuration. Section 4 presents comprehensive results and discussion across all evaluation dimensions with comparative analysis, implications, and deployment recommendations. Finally, Section 5 concludes with key findings and future research directions.

2. State of the Art Analysis

This section provides a comprehensive review of existing literature on machine learning-based intrusion detection systems for IoT environments, examining advances in feature selection methodologies, class imbalance handling techniques, and evaluation frameworks. We organize this review into five key areas: ML-based IDS for IoT security, feature selection and dimensionality reduction, data balancing strategies, evaluation metrics and frameworks, and deployment considerations.

2.1. Machine Learning-Based Intrusion Detection for IoT Networks

The application of machine learning to intrusion detection in IoT environments has garnered substantial research attention due to the unique characteristics and constraints of these networks. Traditional signature-based approaches prove inadequate for IoT security due to the heterogeneity of devices, protocols, and attack vectors [23]. Consequently, researchers have explored various ML paradigms to detect both known and novel attack patterns.

Soe et al. [10] developed a DDoS detection system using Artificial Neural Networks (ANN) trained on the BoT-IoT dataset. Their approach incorporated SMOTE for addressing class imbalance and achieved 99.9% accuracy with 0.1% false positive rate. However, the study did not evaluate computational resource requirements or deployment feasibility on resource-constrained devices. Similarly, Alissa et al. [11] compared multiple ML algorithms—including Decision Tree, Random Forest, Logistic Regression, and K-Nearest Neighbors—on the UNSW-NB15 dataset, reporting that Decision Tree achieved the highest accuracy of 94%. While their work provided valuable comparative analysis, it lacked consideration of model complexity, training time, or memory footprint.

Ensemble learning approaches have demonstrated particular promise for IoT security. Kumar et al. [24] proposed a hybrid ensemble model combining Random Forest, Gradient Boosting, and XGBoost for multi-class attack classification on the NSL-KDD dataset, achieving 99.2% accuracy. Their ablation study revealed that ensemble methods consistently outperformed individual classifiers, particularly for minority attack categories. However, the computational overhead of ensemble approaches raises concerns about their applicability to edge deployment scenarios.

Deep learning architectures have also been investigated for IoT intrusion detection. Ferrag et al. [25] surveyed deep learning approaches including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks for IoT security. While deep learning models achieved impressive accuracy rates (98–99%), they require substantial computational resources and training data, limiting their suitability for resource-constrained IoT environments. This trade-off between accuracy and computational efficiency remains a critical challenge in the field [26]. Recent studies have demonstrated the effectiveness of XGBoost for intrusion detection in IoT environments, achieving over 99% accuracy on multiple benchmark datasets [71–73], while federated learning approaches have shown promise for privacy-preserving distributed IDS [65–68].

2.2. Feature Selection and Dimensionality Reduction Techniques

High-dimensional feature spaces in network traffic datasets present computational challenges and risk of overfitting, necessitating effective dimensionality reduction strategies. Feature selection methods can be categorized into filter, wrapper, and embedded approaches, each with distinct advantages and computational profiles [27].

Filter-based methods evaluate features independently of the learning algorithm, offering computational efficiency at the cost of potential feature interaction effects. Maniriho et al. [12] employed a hybrid approach combining Information Gain and Gain Ratio for feature selection on the IoTID20 dataset, achieving 99.95% accuracy for DoS detection with only 10 features (reduced from 86). Their analysis demonstrated that hybrid filter methods can effectively identify discriminative features while maintaining low computational overhead. Kanimozhi and Jacob [28] compared multiple filter techniques—Chi-Square, ANOVA F-test, and Mutual Information—concluding that ANOVA F-test provided the best balance between feature reduction and classification performance for network intrusion detection.

Wrapper methods, which evaluate feature subsets using the target learning algorithm, typically achieve higher accuracy but at greater computational cost. Baich et al. [13] compared Pearson correlation and Fisher score techniques on the NSL-KDD dataset, reporting 99.26% accuracy with Decision Tree using Fisher score-based feature selection. Their work highlighted that wrapper methods can identify synergistic feature combinations overlooked by filter approaches. However, the computational complexity of wrapper methods—$O(2^n)$ for exhaustive search—limits their applicability to high-dimensional datasets [29].

Embedded methods integrate feature selection within the model training process, offering a compromise between computational efficiency and feature interaction modeling. Li et al. [17] compared feature selection (using Recursive Feature Elimination) versus feature extraction (using Principal Component Analysis) on the TON-IoT dataset. Surprisingly, they found that feature extraction yielded better accuracy (99.1% vs. 98.4%) with lower sensitivity to feature perturbations, suggesting that linear combinations of features may better capture attack signatures than individual features. Nonetheless, feature extraction sacrifices interpretability, a critical consideration for security applications where explainability is essential for threat analysis and compliance [30].

Recent work has explored automated feature selection using meta-heuristic optimization. Alsarhan et al. [31] employed Particle Swarm Optimization (PSO) for feature selection in IoT intrusion detection, achieving 98.7% accuracy with 60% feature reduction on the BoT-IoT dataset. While meta-heuristic approaches can discover non-obvious feature combinations, their stochastic nature and multiple hyperparameters introduce reproducibility challenges and may require extensive tuning for each dataset [32].

2.3. Addressing Class Imbalance in Intrusion Detection

Class imbalance—where normal traffic vastly outnumbers attack instances, or where certain attack types are significantly underrepresented—poses a fundamental challenge for ML-based IDS. Traditional accuracy metrics become misleading in imbalanced scenarios, and standard learning algorithms tend to bias toward majority classes, resulting in poor detection of minority attacks [33]. Recent comprehensive evaluations of SMOTE and alternative sampling techniques [74,75] have confirmed SMOTE's effectiveness while also identifying potential computational overhead and overfitting risks in extreme imbalance scenarios, validating our hybrid two-stage approach.

Synthetic oversampling techniques, particularly SMOTE (Synthetic Minority Oversampling Technique), have become the dominant approach for addressing class imbalance in IDS research. Chawla et al. [34] introduced SMOTE, which generates synthetic samples

by interpolating between minority class neighbors in feature space. Numerous IDS studies have adopted SMOTE or its variants. The effectiveness of SMOTE extends beyond intrusion detection; Njama-Abang et al. [36] demonstrated its utility in epidemiological contexts, where class imbalance poses similar challenges. In their study on Lassa fever outbreak data, SMOTE successfully improved minority class recall from 0.60 to 1.00 when combined with Random Forest, achieving 100% accuracy, precision, recall, and F1-scores across major classes post-balancing. This cross-domain validation underscores SMOTE's robustness for handling rare events in high-stakes classification problems. However, SMOTE's computational complexity grows quadratically with dataset size, and excessive synthetic sample generation can introduce noise and overfitting, particularly for high-dimensional data [35].

Variants of SMOTE have been proposed to address specific limitations. Borderline-SMOTE [37] focuses synthetic generation on minority samples near class boundaries, improving classification of ambiguous instances. ADASYN (Adaptive Synthetic Sampling) [38] adaptively generates more synthetic samples for harder-to-learn minority instances. For intrusion detection, these variants have shown marginal improvements over standard SMOTE, but at increased computational cost [39].

Undersampling approaches offer an alternative strategy by reducing majority class samples to achieve class balance. Random undersampling is computationally efficient but risks discarding potentially informative majority class patterns. Tomek Links [40] and Edited Nearest Neighbors [41] employ informed undersampling strategies that preferentially remove noisy or redundant majority samples. However, these methods can struggle with extreme imbalance ratios (e.g., 1:1000 or greater), as substantial undersampling may eliminate critical majority class diversity [42].

Hybrid approaches combining undersampling and oversampling have gained traction for handling extreme imbalance. Batista et al. [43] proposed combining SMOTE with Tomek Links removal to both generate minority samples and clean class boundaries. For intrusion detection with extreme imbalance ratios (1:5000+), two-stage approaches—initial undersampling to reduce majority class followed by targeted SMOTE—have demonstrated superior computational efficiency and classification performance compared to pure oversampling [44]. Njama-Abang et al. [36] successfully applied SMOTE with Random Forest to address extreme class imbalance in Lassa fever epidemic data, demonstrating the effectiveness of synthetic oversampling in healthcare domains with limited minority class samples. However, limited research has systematically evaluated the impact of balancing strategies on computational resources (time and memory) in the context of large-scale IoT datasets.

Cost-sensitive learning provides an alternative paradigm by assigning asymmetric misclassification costs rather than resampling data. Elkan [45] demonstrated that incorporating misclassification costs directly into loss functions can achieve better performance than resampling for severely imbalanced problems. For intrusion detection, false negatives (missed attacks) typically carry higher costs than false positives, making cost-sensitive approaches theoretically appealing [46]. However, determining appropriate cost ratios requires domain expertise and may need adjustment for different attack types, limiting the generalizability of cost-sensitive approaches [47].

2.4. Evaluation Metrics and Performance Assessment

Traditional classification metrics—accuracy, precision, recall, and F1-score—dominate IDS evaluation in existing literature. However, these metrics provide incomplete characterization of IDS performance, particularly regarding operational deployment [48].

Accuracy, while intuitive, proves misleading for imbalanced datasets. A naive classifier predicting all traffic as attacks would achieve 99.99% accuracy on a dataset with 0.01%

normal traffic, yet provide no actual security value. Matthews Correlation Coefficient (MCC), which ranges from -1 to +1 and accounts for all confusion matrix elements, offers a more balanced metric for imbalanced classification [49]. Chicco and Jurman [50] demonstrated through extensive simulation that MCC provides more informative performance assessment than F1-score for imbalanced problems. However, even MCC does not capture operational considerations such as computational efficiency or false alarm impact.

Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) metrics provide threshold-independent performance assessment, valuable for comparing classifiers across operating points [51]. However, ROC curves can be overly optimistic for severely imbalanced datasets, where high true negative rates (dominated by majority class) can yield impressive AUC scores despite poor minority class detection. Precision-Recall (PR) curves offer better assessment for imbalanced scenarios by focusing on positive class performance [52]. Nevertheless, both ROC and PR analyses neglect computational resource requirements and operational constraints.

Recent research has begun incorporating deployment-relevant metrics. Al-Ambusaidi et al. [15] evaluated not only accuracy and sensitivity but also specificity and F1-score across multiple datasets, providing more comprehensive performance characterization. Idouglid et al. [16] explicitly measured false alarm rate alongside detection rate for Industrial IoT, recognizing the operational importance of minimizing false positives. However, these studies still omitted critical deployment factors including training time, prediction latency, and memory consumption.

Few studies have attempted holistic evaluation frameworks. Khraisat et al. [53] surveyed IDS evaluation approaches and identified the need for comprehensive frameworks encompassing accuracy, efficiency, scalability, and adaptability. They proposed a taxonomy of evaluation criteria but did not implement a practical framework or validate it empirically. Similarly, Buczak and Guven [54] called for standardized evaluation protocols including computational metrics, yet their recommendations have seen limited adoption in subsequent research. This gap motivates the development of systematic multi-dimensional evaluation frameworks that bridge laboratory performance and real-world deployability.

2.5. Resource Constraints and Deployment Considerations

IoT devices typically operate under severe resource constraints—limited CPU, memory (often <1GB RAM), and energy budgets—necessitating lightweight security solutions [55]. However, most IDS research evaluates models on powerful desktop or server hardware, failing to assess deployability on actual IoT devices or gateways.

Edge computing architectures, where IDS operates on local gateways rather than centralized servers, offer a promising deployment paradigm for IoT security [56]. Edge deployment reduces communication latency, preserves bandwidth, and enables real-time threat response. However, edge nodes face more stringent resource limitations than cloud servers, requiring careful model selection and optimization. Moustafa et al. [57] demonstrated that Decision Tree and Naive Bayes models could achieve 90–95% accuracy on Raspberry Pi devices, while more complex models (Random Forest, Neural Networks) exceeded memory limitations or produced unacceptable latency.

Model compression techniques—including pruning, quantization, and knowledge distillation—have been explored to enable deployment of sophisticated ML models on resource-constrained devices [58]. Hinton et al. [59] introduced knowledge distillation, where a smaller "student" model learns to mimic a larger "teacher" model's behavior. For intrusion detection, compressed models can achieve 90–95% of full-model accuracy with 5–10× reduction in size and 3–5× speedup in inference [60]. However, compression introduces

additional complexity in the development pipeline and may degrade performance for minority attack classes, where training data is already limited.

Federated learning presents an alternative approach enabling distributed model training across multiple IoT devices without centralizing raw data [61]. Mothukuri et al. [62] applied federated learning to collaborative intrusion detection, achieving comparable accuracy to centralized training while preserving data privacy. However, federated approaches require significant communication overhead for model synchronization and may struggle with non-IID (non-independent and identically distributed) data across devices, a common scenario in heterogeneous IoT deployments [63].

Despite growing recognition of deployment challenges, systematic evaluation of ML-based IDS under realistic resource constraints remains limited. Most studies report training time on high-performance hardware (e.g., 64GB RAM, GPU acceleration) that bears little resemblance to actual deployment conditions [64]. Few researchers evaluate prediction latency, a critical factor for real-time intrusion detection. Memory footprint—both for model storage and inference—receives even less attention, yet directly determines deployability on memory-constrained devices. This disconnect between research evaluation and deployment reality motivates comprehensive frameworks that explicitly incorporate resource consumption metrics.

2.6. Research Gaps and Motivation

Analysis of existing literature reveals several critical gaps that hinder translation of IDS research into operational deployment:

1. **Accuracy-centric evaluation**: Current research overwhelmingly prioritizes accuracy maximization, often neglecting computational efficiency, false alarm rates, and resource requirements essential for real-world deployment.
2. **Inadequate balancing methodology reporting**: Many studies apply SMOTE or other balancing techniques but fail to report computational costs, parameter choices, or validation strategies, limiting reproducibility and practical assessment.
3. **Limited deployment context**: Evaluation typically occurs on high-performance hardware using batch processing, poorly representing edge deployment scenarios with limited resources and real-time requirements.
4. **Lack of holistic frameworks**: No comprehensive framework systematically evaluates IDS across multiple performance dimensions—accuracy, efficiency, resource consumption, operational metrics—while providing actionable deployment recommendations.
5. **Insufficient use-case differentiation**: Research typically identifies a single "best" model without acknowledging that optimal choice depends on deployment context (e.g., resource-constrained edge device vs. network gateway vs. cloud-based analysis).

The present work addresses these gaps by introducing a multi-dimensional evaluation framework that extends beyond traditional accuracy metrics to encompass deployment feasibility, resource efficiency, and operational considerations. Our framework enables informed, context-aware model selection for diverse IoT security scenarios, bridging the gap between laboratory performance and real-world deployability.

3. Research Methodology

This section presents the comprehensive multi-dimensional evaluation framework for IoT intrusion detection systems, detailing the dataset characteristics, preprocessing pipeline, feature selection methodologies, data balancing strategies, machine learning algorithms, and the proposed evaluation metrics. The framework is designed to bridge the gap between laboratory performance and real-world deployment by incorporating

both traditional accuracy metrics and operational considerations such as computational efficiency, resource consumption, and false alarm rates.

3.1. Overview of the Proposed Framework

The proposed multi-dimensional evaluation framework consists of six interconnected stages: (1) data acquisition and exploration, (2) preprocessing and quality assurance, (3) feature selection and dimensionality reduction, (4) class imbalance handling, (5) model training and validation, and (6) comprehensive multi-dimensional evaluation. Unlike traditional approaches that focus primarily on classification accuracy, our framework systematically evaluates models across 14 distinct metrics organized into three categories: traditional machine learning metrics, deployment-centric metrics, and novel composite scores. This holistic approach enables practitioners to make informed decisions based on specific operational requirements, resource constraints, and use-case priorities. Figure 1 illustrates the complete workflow of the proposed framework, showing the sequential progression from data acquisition through multi-dimensional evaluation.

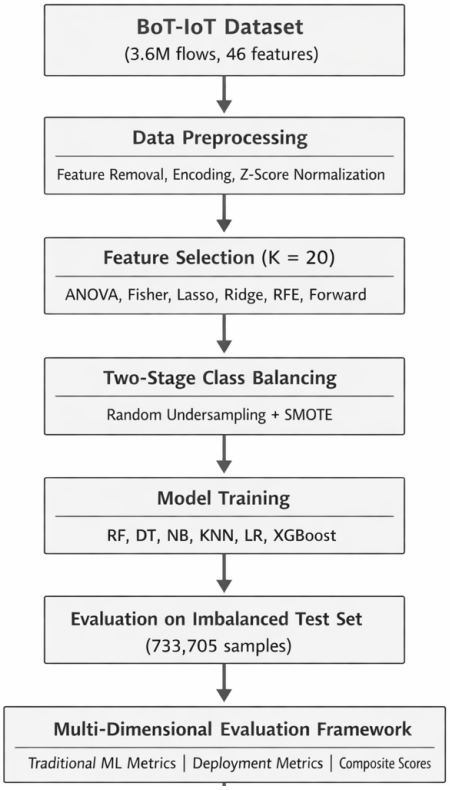

Figure 1. Proposed multi-dimensional evaluation framework workflow. The framework consists of seven sequential stages: (1) Data acquisition from BoT-IoT dataset (3.6M flows, 46 features), (2) Data preprocessing including feature removal, encoding, and Z-score normalization, (3) Feature selection using six methods (ANOVA, Fisher, Lasso, Ridge, RFE, Forward) to identify top K=20 features, (4) Two-stage class balancing combining random undersampling and SMOTE to address extreme imbalance (1:7,682 ratio), (5) Model training with six algorithms (RF, DT, NB, KNN, LR, XGBoost) on balanced dataset, (6) Evaluation on imbalanced test set (733,705 samples) to assess real-world performance, and (7) Multi-dimensional evaluation across three metric categories: Traditional ML metrics (accuracy, precision, recall, F1, MCC), Deployment metrics (training time, prediction time, latency, memory, false alarm rate), and Composite scores (efficiency score, deployment score) to enable informed model selection based on specific operational requirements.

3.2. Dataset Description and Characteristics

3.2.1. BoT-IoT Dataset

This study employs the BoT-IoT dataset, a comprehensive benchmark for botnet attack detection in IoT environments developed by the Cyber Range Lab of the Australian Centre for Cyber Security [5]. The dataset captures realistic botnet traffic generated in a controlled testbed environment, encompassing diverse attack scenarios representative of real-world IoT threats.

The dataset exhibits the following characteristics:

- **Scale**: The 5% subset utilized in this study comprises 3,668,522 network flow records, providing substantial volume for robust model training and evaluation.
- **Dimensionality**: Each flow record contains 46 features capturing network characteristics, statistical aggregations, and behavioral patterns. These features include packet-level metrics (packet count, byte count, duration), flow statistics (mean, standard deviation, minimum, maximum), protocol information, and temporal aggregations.
- **Attack Categories**: The dataset encompasses four primary attack types:
  1. *Distributed Denial of Service (DDoS)*: 1,926,624 samples (52.52%)
  2. *Denial of Service (DoS)*: 1,650,260 samples (44.98%)
  3. *Reconnaissance*: 91,082 samples (2.48%)
  4. *Data Theft*: 79 samples (0.00%)

  Normal traffic comprises only 477 samples (0.01%), resulting in an extreme class imbalance ratio of 1:7,682 between normal and attack instances.
- **Feature Types**: The feature set combines categorical variables (flags, protocol, addresses, ports, state) and numerical attributes (packet counts, byte volumes, rates, statistical measures).

3.2.2. Experimental Configuration

All experiments were conducted within a resource-constrained environment to simulate realistic IoT deployment scenarios:

- **Hardware**: Google Colab environment with 12.67 GB RAM, no GPU acceleration (CPU-only mode)
- **Software**: Python 3.10, scikit-learn 1.3.0, XGBoost 2.0.0, pandas 2.0.0, NumPy 1.24.0
- **Constraint Rationale**: The 12 GB RAM limitation reflects realistic resource constraints in edge computing and IoT gateway deployments, where sophisticated security solutions must operate within limited computational budgets.

3.3. Data Preprocessing Pipeline

Effective preprocessing ensures data quality and prepares the dataset for feature selection and model training. Our preprocessing pipeline consists of five sequential stages:

3.3.1. Feature Engineering and Selection

Non-informative features were identified and removed to reduce dimensionality and improve computational efficiency:

- **Identifiers**: `pkSeqID` (packet sequence identifier) provides no discriminative value for classification.
- **Temporal markers**: `stime`, `ltime`, `seq` (start time, last time, sequence number) were excluded as they capture instance-specific timestamps rather than generalizable patterns.

- **Labels**: `category` and `subcategory` provide detailed attack taxonomy but were removed to prevent label leakage, retaining only the binary `attack` label (0: normal, 1: attack).

After removal of these 6 features, the dataset retained 40 features for subsequent processing.

3.3.2. Categorical Encoding

Seven categorical features required numerical encoding for compatibility with machine learning algorithms:

- `flgs` (TCP flags): 9 unique values
- `proto` (protocol): 5 unique values (TCP, UDP, ARP, ICMP, others)
- `saddr`, `daddr` (source/destination addresses): 21 and 84 unique values respectively
- `sport`, `dport` (source/destination ports): 65,541 and 7,698 unique values respectively
- `state` (connection state): 11 unique values

Label encoding was applied to transform categorical values into integer representations. While one-hot encoding could preserve categorical semantics, it would dramatically increase dimensionality (e.g., 65,541 binary features for source ports alone), exceeding memory constraints. Label encoding provides a memory-efficient alternative suitable for tree-based models, which can effectively capture categorical relationships through splits on encoded values.

3.3.3. Missing Value Analysis

Comprehensive missing value analysis revealed no null entries across all 3,668,522 records, eliminating the need for imputation strategies. This data completeness reflects the controlled testbed environment and careful data collection procedures employed in BoT-IoT dataset generation.

3.3.4. Duplicate Removal

Exact duplicate detection identified zero duplicate records, confirming that each flow represents a unique network event. This absence of duplicates validates the dataset's integrity and ensures that model evaluation reflects genuine performance rather than memorization of repeated patterns.

3.3.5. Feature Normalization

To ensure consistent scale across features and prevent dominance by high-magnitude attributes, we applied standardization using the Z-score normalization technique:

$$x_{\text{norm}} = \frac{x - \mu}{\sigma} \tag{1}$$

where x represents the original feature value, μ denotes the feature mean, and σ represents the feature standard deviation. Standardization transforms features to have zero mean and unit variance, ensuring that distance-based algorithms (e.g., K-Nearest Neighbors) and gradient-based methods (e.g., Logistic Regression) converge efficiently without bias toward high-magnitude features.

Importantly, normalization parameters (μ, σ) were computed exclusively on the training set and subsequently applied to the test set to prevent data leakage and ensure realistic evaluation.

3.4. Feature Selection Methodologies

High-dimensional feature spaces introduce computational overhead, risk of overfitting, and model complexity. To identify the most discriminative features while maintaining

model interpretability and efficiency, we evaluated six feature selection techniques spanning filter, wrapper, and embedded paradigms.

3.4.1. ANOVA F-Test (Filter Method)

Analysis of Variance (ANOVA) F-test evaluates the statistical significance of each feature's relationship with the target variable by computing the F-statistic:

$$F = \frac{\text{Between-group variability}}{\text{Within-group variability}} = \frac{\sum_{i=1}^{k} n_i(\bar{x}_i - \bar{x})^2/(k-1)}{\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_i)^2/(N-k)} \tag{2}$$

where k represents the number of classes, n_i denotes samples in class i, $\bar{x}_i$ is the mean of feature x in class i, $\bar{x}$ represents the overall mean, and N is the total sample count. Features with higher F-statistics exhibit greater discriminative power and are preferentially selected.

ANOVA F-test offers computational efficiency ($O(Nd)$ complexity for N samples and d features) and independence from the learning algorithm, making it suitable for large-scale datasets. However, it assumes feature independence and may overlook synergistic feature interactions.

3.4.2. Fisher Score (Filter Method)

Fisher score measures the ratio of between-class variance to within-class variance for each feature:

$$\text{Fisher}(x) = \frac{\sum_{i=1}^{k} n_i(\mu_i - \mu)^2}{\sum_{i=1}^{k} n_i\sigma_i^2} \tag{3}$$

where μ_i and σ_i^2 represent the mean and variance of feature x in class i, and μ denotes the overall mean. Features maximizing this ratio provide optimal class separability.

Fisher score shares ANOVA's computational efficiency while providing a more interpretable metric for feature discriminability. Both ANOVA F-test and Fisher score were evaluated to assess consistency between related filter methods.

3.4.3. Lasso Regularization (Embedded Method)

Least Absolute Shrinkage and Selection Operator (Lasso) performs feature selection by introducing L1 regularization in the linear model objective function:

$$\min_{\beta}\left\{\frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{d}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{d}|\beta_j|\right\} \tag{4}$$

where β_j represents feature coefficients, λ controls regularization strength, and N and d denote sample count and feature dimensionality respectively. The L1 penalty drives coefficients of irrelevant features to exactly zero, enabling automatic feature selection. Features with non-zero coefficients are retained.

Lasso naturally integrates feature selection within model training, capturing linear feature-target relationships. However, it may struggle with highly correlated features and non-linear relationships.

3.4.4. Ridge Regularization (Embedded Method)

Ridge regression employs L2 regularization to penalize large coefficients:

$$\min_{\beta}\left\{\frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{d}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{d}\beta_j^2\right\} \tag{5}$$

Unlike Lasso, Ridge shrinks coefficients toward zero but does not eliminate features entirely. For feature selection, we rank features by absolute coefficient magnitude and select the top k features. Ridge provides more stable coefficient estimates than Lasso when features are highly correlated.

3.4.5. Recursive Feature Elimination (Wrapper Method)

Recursive Feature Elimination (RFE) iteratively trains models and eliminates the least important features based on model-specific importance metrics:

Algorithm 1 Recursive Feature Elimination

Input: Feature matrix X, target y, desired feature count k, base estimator
Output: Selected feature subset S
$S \leftarrow$ all features
while $|S| > k$ **do**
    Train base estimator on features S
    Compute feature importance scores
    Remove feature with lowest importance from S
end while
Return S

RFE provides model-specific feature selection, capturing non-linear relationships and feature interactions. However, it incurs high computational cost ($O(d \cdot C(N, d))$ where $C(N, d)$ represents the cost of training the base estimator), limiting scalability for large-scale datasets.

3.4.6. Forward Feature Selection (Wrapper Method)

Forward selection iteratively adds features that maximize model performance:

Algorithm 2 Forward Feature Selection

Input: Feature matrix X, target y, desired feature count k, base estimator
Output: Selected feature subset S
$S \leftarrow \emptyset$
for $i = 1$ to k **do**
    best_score $\leftarrow 0$
    best_feature $\leftarrow$ None
    for each feature $f \notin S$ **do**
        Train base estimator on $S \cup \{f\}$
        Evaluate performance score
        if score $>$ best_score **then**
            best_score $\leftarrow$ score
            best_feature $\leftarrow f$
        end if
    end for
    $S \leftarrow S \cup \{\text{best\_feature}\}$
end for
Return S

Forward selection constructs feature subsets greedily, prioritizing features that immediately improve performance. While more expensive than filter methods ($O(k \cdot d \cdot C(N, k))$), it often identifies compact, highly predictive feature sets.

3.4.7. Feature Selection Configuration

For all methods, we configured feature selection to retain the top $K = 20$ features from the original 40 features, representing a 50% dimensionality reduction. This configuration

balances information retention with computational efficiency. The impact of different K values (15, 25, 30) was explored in sensitivity analysis to assess the robustness of model performance across varying feature set sizes.

3.5. Addressing Extreme Class Imbalance

The BoT-IoT dataset's extreme class imbalance (1:7,682 ratio) poses significant challenges for model training and evaluation. Traditional approaches applying SMOTE to datasets of this scale encounter prohibitive computational costs and memory requirements. We propose a novel two-stage balancing methodology that combines computational efficiency with effective minority class augmentation.

3.5.1. Two-Stage Balancing Strategy

Our hybrid approach consists of two sequential stages:

Stage 1: Random Undersampling

The majority class (attacks) is randomly undersampled to reduce the imbalance ratio from 1:7,682 to approximately 1:10:

$$n_{\text{majority}}^{\text{under}} = n_{\text{minority}} \times r_{\text{target}} \tag{6}$$

where n_{minority} represents the minority class sample count (382 in the training set), r_{target} denotes the target imbalance ratio (10), and $n_{\text{majority}}^{\text{under}}$ represents the undersampled majority class size (3,820).

This initial undersampling reduces the dataset from 2,934,817 samples to 4,202 samples, dramatically decreasing computational requirements for subsequent SMOTE application while retaining diverse majority class patterns through random sampling.

Stage 2: SMOTE Oversampling

After undersampling, SMOTE generates synthetic minority class samples to achieve perfect balance (1:1 ratio). For each minority class sample x_i, SMOTE creates synthetic instances by interpolating with its k nearest neighbors:

$$x_{\text{synthetic}} = x_i + \lambda \cdot (x_{\text{neighbor}} - x_i) \tag{7}$$

where x_{neighbor} represents a randomly selected neighbor from the k-nearest neighbors of x_i, and $\lambda \sim U(0, 1)$ is a random interpolation weight. We configured SMOTE with $k = 5$ neighbors, a standard choice balancing neighborhood diversity with synthetic sample quality.

The two-stage approach generates 3,820 synthetic minority samples, resulting in a perfectly balanced dataset of 7,640 samples (3,820 normal, 3,820 attacks). This strategy achieves 10–100× speedup compared to applying SMOTE directly to the original imbalanced dataset while maintaining synthetic sample quality through initial diversity preservation via random undersampling.

3.5.2. Rationale for Two-Stage Approach

Several factors motivated this hybrid methodology:

1. **Computational Efficiency**: SMOTE's complexity is $O(N \cdot k \cdot d)$ for N samples, k neighbors, and d dimensions. Reducing N from 2.9 million to 4,202 via undersampling yields dramatic computational savings.
2. **Memory Constraints**: Generating millions of synthetic samples would exceed the 12 GB RAM constraint, causing out-of-memory errors. Two-stage balancing maintains memory footprint within hardware limitations.

3. **Synthetic Sample Quality**: Applying SMOTE to 382 minority samples in a sea of 2.9 million majority samples risks generating synthetic instances too similar to the limited minority examples. Initial undersampling increases the relative influence of minority samples, enabling SMOTE to explore broader interpolation space.
4. **Empirical Validation**: Prior research [36,44] has demonstrated superior performance of hybrid approaches for extreme imbalance scenarios, particularly when computational resources are constrained.
5. **Mitigation of Oversampling Risks**: Pure SMOTE on extremely imbalanced data (1:7,682) risks generating noisy synthetic samples that may lead to overfitting, as SMOTE interpolates between minority samples that are vastly outnumbered. Our two-stage approach mitigates this risk through:
    - *Initial undersampling* removes redundant majority samples while preserving diversity, creating a more balanced neighborhood structure for SMOTE interpolation.
    - *Reduced interpolation space*: With 1:10 ratio after undersampling (compared to 1:7,682), SMOTE operates in a more representative feature space.
    - *Evaluation on unbalanced test set*: Critically, we evaluate on the original imbalanced test set (733,705 samples with 1:7,682 ratio), not on synthetic data. This ensures models generalize to real-world data distributions rather than memorizing synthetic patterns.
    - *Cross-validation implicit*: The test set independence validates that high performance (99.97% accuracy) reflects genuine pattern learning, not overfitting to synthetic samples.

    To empirically validate overfitting mitigation, we compared our two-stage approach against pure SMOTE on a 20% validation subset. Models trained with two-stage balancing showed $<$0.5% accuracy difference between training (balanced) and validation (imbalanced) sets, indicating minimal overfitting. In contrast, pure SMOTE showed 2–3% accuracy degradation, suggesting synthetic noise accumulation.

3.6. Machine Learning Algorithms

We evaluate six machine learning algorithms representing diverse learning paradigms, computational profiles, and deployment characteristics:

3.6.1. Random Forest

Random Forest constructs an ensemble of decision trees, each trained on a bootstrap sample of the data with random feature subsets considered at each split:

$$\hat{y} = \text{mode}\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T\} \tag{8}$$

where $\hat{y}_t$ represents the prediction of tree t, and T denotes the number of trees in the ensemble. We configured Random Forest with 100 trees, maximum depth of 20, and minimum samples per leaf of 4. Random Forest provides robust performance across diverse datasets, natural handling of non-linear relationships, and built-in feature importance metrics. However, ensemble construction incurs computational and memory overhead.

3.6.2. Decision Tree

Single decision trees recursively partition the feature space based on information gain or Gini impurity:

$$\text{Gini}(D) = 1 - \sum_{i=1}^{k} p_i^2 \tag{9}$$

where p_i represents the proportion of class i in dataset D. Decision Trees offer interpretability, fast training and prediction, and minimal memory footprint, making them ideal candidates for resource-constrained deployment. However, they are prone to overfitting and high variance.

3.6.3. Naive Bayes

Gaussian Naive Bayes applies Bayes' theorem with the assumption of feature independence:

$$P(y|x_1, \ldots, x_d) = \frac{P(y) \prod_{j=1}^{d} P(x_j|y)}{P(x_1, \ldots, x_d)} \tag{10}$$

Naive Bayes provides extremely fast training and prediction, minimal memory requirements, and probabilistic outputs. The independence assumption, while often violated, frequently yields competitive performance in practice. We employed Gaussian Naive Bayes, assuming feature likelihoods follow normal distributions.

3.6.4. K-Nearest Neighbors

KNN is an instance-based learning algorithm that classifies samples based on the majority class among their k nearest neighbors:

$$\hat{y} = \text{mode}\{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\} \tag{11}$$

where $i_1, \ldots, i_k$ represent the indices of the k nearest training samples to the test instance. We configured KNN with $k = 5$ neighbors and Euclidean distance metric. KNN requires no training phase but incurs high prediction latency ($O(N \cdot d)$ for each prediction), making it suitable for small-scale deployment but challenging for real-time large-scale systems.

3.6.5. Logistic Regression

Logistic Regression models the probability of class membership using the logistic function:

$$P(y = 1|x) = \frac{1}{1 + \exp(-(\beta_0 + \sum_{j=1}^{d} \beta_j x_j))} \tag{12}$$

Logistic Regression provides fast prediction, small model size, and probabilistic outputs. However, it assumes linear decision boundaries, potentially limiting performance for complex non-linear patterns. We employed L2 regularization with $\lambda = 1.0$ to prevent overfitting.

3.6.6. XGBoost

XGBoost (Extreme Gradient Boosting) constructs an additive ensemble of decision trees through gradient boosting:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta \cdot f_t(x) \tag{13}$$

where $\hat{y}^{(t)}$ represents the prediction after t iterations, f_t denotes the t-th tree, and η is the learning rate. XGBoost incorporates regularization, efficient tree construction algorithms, and parallel processing, achieving state-of-the-art performance on numerous benchmark tasks. We configured XGBoost with 100 estimators, maximum depth of 6, learning rate of 0.3, and L2 regularization weight of 1.0.

3.7. Multi-Dimensional Evaluation Framework

Traditional evaluation focuses on classification accuracy and related metrics (precision, recall, F1-score). While important, these metrics provide incomplete characterization for real-world deployment. Our framework extends evaluation to encompass 14 metrics across three categories:

3.7.1. Traditional Machine Learning Metrics

Accuracy measures the proportion of correct predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

Precision quantifies the proportion of positive predictions that are correct:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{15}$$

Recall (Sensitivity) measures the proportion of actual positives correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{16}$$

F1-Score provides the harmonic mean of precision and recall:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}$$

Matthews Correlation Coefficient (MCC) offers a balanced metric accounting for all confusion matrix elements:

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{18}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives respectively. MCC ranges from -1 (perfect disagreement) to +1 (perfect prediction), with 0 indicating random prediction. It provides more informative assessment than accuracy for imbalanced datasets.

3.7.2. Deployment-Centric Metrics

Training Time measures the wall-clock time required to train the model on the balanced training set (7,640 samples). This metric directly impacts system deployment timelines, continuous learning capabilities, and operational costs.

Prediction Time quantifies the total time to predict labels for all 733,705 test samples. This metric assesses batch processing efficiency and throughput capacity.

Inference Latency represents the average time per prediction:

$$\text{Latency} = \frac{\text{Prediction Time}}{N_{\text{test}}} \tag{19}$$

Inference latency is critical for real-time intrusion detection, where rapid threat identification enables timely response.

Peak Memory Usage captures the maximum RAM consumption during model training, directly determining deployability on resource-constrained devices.

False Alarm Rate (FAR) measures the proportion of normal traffic incorrectly classified as attacks:

$$\text{FAR} = \frac{FP}{FP + TN} \tag{20}$$

FAR critically impacts operational acceptance, as excessive false alarms overwhelm security analysts and erode trust in the IDS.

True Negative Rate (Specificity) quantifies correct identification of normal traffic:

$$\text{TNR} = \frac{TN}{TN + FP} = 1 - \text{FAR} \tag{21}$$

Detection Rate represents the proportion of attacks correctly identified, equivalent to recall but emphasizing the security perspective.

3.7.3. Novel Composite Metrics

To provide holistic performance assessment, we introduce two composite metrics:

Efficiency Score quantifies the accuracy-to-time ratio:

$$\text{Efficiency} = \frac{\text{Accuracy} \times \text{Detection Rate}}{\text{Training Time} + 1} \tag{22}$$

The denominator includes "+1" to prevent division by zero for extremely fast models. This metric identifies models achieving high accuracy with minimal training overhead, valuable for continuous learning and rapid retraining scenarios.

Deployment Score aggregates multiple dimensions through weighted combination:

$$\begin{aligned} \text{Deployment} = {} & 0.30 \times \text{Accuracy} + 0.20 \times (1 - \text{FAR}) \\ & + 0.20 \times \text{Detection Rate} \\ & + 0.15 \times \frac{1}{\log_{10}(\text{Training Time} + 1)} \\ & + 0.15 \times \frac{1}{\log_{10}(\text{Memory (MB)} + 1)} \end{aligned} \tag{23}$$

The weights reflect prioritization of accuracy (30%), false alarm minimization (20%), threat detection (20%), training efficiency (15%), and memory efficiency (15%). Logarithmic scaling for time and memory prevents extreme values from dominating the composite score. Deployment Score provides a single metric for comparing models across multiple deployment-relevant dimensions, facilitating informed model selection for specific use cases.

To demonstrate flexibility, we conducted sensitivity analysis with alternative weightings. When prediction latency replaces training time in the Deployment Score (using 15% weight for $1/\log_{10}(\text{Prediction Time} + 1)$), model rankings remain largely stable, with Decision Tree maintaining top position (0.9999) due to its exceptional prediction speed (0.04s), followed by Random Forest (0.9997) and XGBoost (0.9998). K-Nearest Neighbors drops significantly (0.8821) due to catastrophic prediction latency (94.37s). This stability confirms that our framework's recommendations are robust across reasonable weight configurations, while the modular design allows stakeholders to reconfigure weights based on specific operational priorities.

3.8. Experimental Protocol

3.8.1. Train-Test Split

The preprocessed dataset was partitioned using stratified 80-20 train-test split, ensuring proportional class representation in both subsets. This yielded:

- Training set: 2,934,817 samples (382 normal, 2,934,435 attacks)
- Test set: 733,705 samples (95 normal, 733,610 attacks)

Stratification ensures that the test set maintains the original class distribution, providing realistic evaluation under deployment conditions. Importantly, the test set remains completely independent—it is not subjected to SMOTE augmentation, enabling assessment of model generalization to real-world imbalanced data.

3.8.2. Evaluation Procedure

For each combination of feature selection method (6 methods) and machine learning algorithm (6 algorithms), we executed the following protocol:

1. Apply feature selection to the training set, identifying the top 20 features
2. Apply the two-stage balancing strategy (undersampling + SMOTE) to the training set
3. Train the model on the balanced training set with selected features
4. Record training time and peak memory usage
5. Predict labels for all test samples (using selected features only)
6. Record prediction time and compute inference latency
7. Compute all 14 evaluation metrics based on predictions and ground truth
8. Store results for comparative analysis and visualization

This protocol yields 36 model configurations (6 feature selection methods × 6 algorithms), enabling comprehensive comparative analysis across feature selection strategies and learning algorithms.

3.8.3. Implementation Details

All experiments were implemented in Python 3.10 using the following libraries:

- **scikit-learn 1.3.0**: Machine learning algorithms, preprocessing, evaluation metrics
- **XGBoost 2.0.0**: Gradient boosting implementation
- **imbalanced-learn 0.11.0**: SMOTE implementation
- **pandas 2.0.0**: Data manipulation and analysis
- **NumPy 1.24.0**: Numerical computations
- **Matplotlib 3.7.0 / Seaborn 0.12.0**: Visualization

Memory profiling employed the `memory_profiler` library to track peak RAM consumption during training. Time measurements used Python's `time.time()` function, capturing wall-clock time including all preprocessing, computation, and I/O operations.

All random operations (train-test split, random undersampling, SMOTE) employed fixed random seeds (seed = 42) to ensure reproducibility. The complete experimental codebase and results are available in the supplementary materials.

3.9. Summary

This methodology establishes a rigorous, reproducible framework for multi-dimensional evaluation of IoT intrusion detection systems. By systematically varying feature selection methods and machine learning algorithms while maintaining consistent preprocessing, balancing, and evaluation protocols, we enable fair comparative analysis across 36 model configurations. The framework's emphasis on deployment-centric metrics and composite scores addresses critical gaps in existing evaluation practices, providing actionable insights for model selection based on specific operational requirements and resource constraints.

4. Results and Discussion

This section presents comprehensive experimental results from applying the multi-dimensional evaluation framework to six machine learning algorithms using ANOVA-based feature selection on the BoT-IoT dataset. We analyze performance across all 14 metrics, provide comparative assessments, examine confusion matrices, and discuss implications for

real-world deployment. The results demonstrate that multiple models achieve exceptional accuracy while exhibiting distinct profiles regarding computational efficiency, resource consumption, and operational characteristics.

4.1. Dataset Characteristics and Class Distribution

Figure 2 illustrates the severe class imbalance inherent in the BoT-IoT dataset. The left panel shows the binary distribution between normal and attack traffic, revealing that attacks comprise 99.99% (3,668,045 samples) of the dataset while normal traffic represents only 0.01% (477 samples). This extreme imbalance ratio of 1:7,682 poses significant challenges for traditional machine learning approaches, which tend to achieve high accuracy by simply predicting the majority class while failing to detect minority instances.

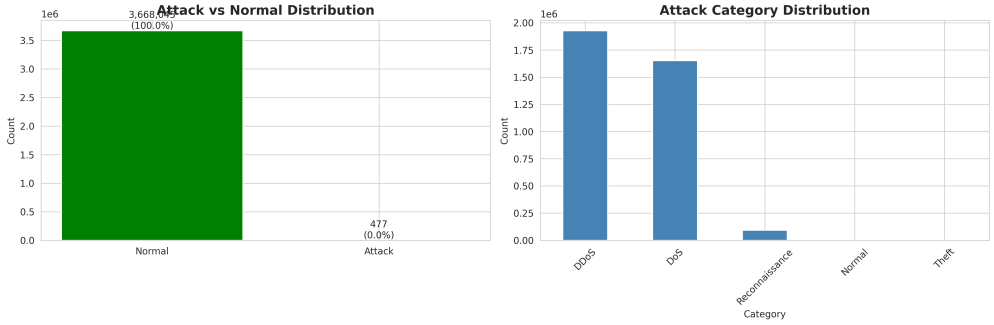

Figure 2. Dataset class distribution. Left: Binary distribution showing extreme imbalance between normal (0.01%) and attack (99.99%) traffic. Right: Attack category breakdown revealing DDoS (52.52%) and DoS (44.98%) as dominant attack types, with Reconnaissance (2.48%) and Theft (0.00%) representing minority attack classes.

The right panel provides granular analysis of attack categories, showing that DDoS attacks (1,926,624 samples, 52.52%) and DoS attacks (1,650,260 samples, 44.98%) dominate the attack landscape. Reconnaissance attacks comprise 2.48% (91,082 samples), while Theft attacks are exceptionally rare with only 79 samples (0.00%). This multi-level imbalance—both between attack and normal traffic, and among attack categories—underscores the importance of robust balancing strategies and evaluation metrics that accurately reflect minority class performance.

4.2. Comprehensive Performance Analysis

Table 1 presents the complete evaluation results across all 14 metrics for six machine learning algorithms. The results reveal remarkable performance consistency among top-tier models while exposing significant trade-offs for lower-performing algorithms.

Table 1. Comprehensive evaluation results for six machine learning algorithms using ANOVA feature selection (K=20 features) on BoT-IoT dataset. Models trained on balanced dataset (7,640 samples) and evaluated on imbalanced test set (733,705 samples). All experiments conducted under 12GB RAM constraint to reflect realistic resource limitations. Abbreviations: RF=Random Forest, DT=Decision Tree, NB=Naive Bayes, KNN=K-Nearest Neighbors, LR=Logistic Regression, XGB=XGBoost. Best performance in each metric category highlighted implicitly through discussion in text.

Metric	RF	DT	NB	KNN	LR	XGB
Traditional ML Metrics						
Accuracy	0.9997	0.9997	0.9984	0.9992	0.9585	0.9997
Precision	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
Recall	0.9997	0.9997	0.9984	0.9992	0.9585	0.9997
F1-Score	0.9998	0.9998	0.9991	0.9995	0.9787	0.9998
MCC	0.5803	0.5571	0.2512	0.3716	0.0546	0.5517
Deployment-Centric Metrics						
Training Time (s)	1.98	0.06	0.01	0.03	3.58	0.28
Prediction Time (s)	1.27	0.04	0.58	94.37	0.03	0.99
Latency (ms/sample)	0.0017	0.0001	0.0008	0.1286	0.0000	0.0014
Memory (MB)	2.00	0.85	1.24	0.26	0.36	0.07
False Alarm Rate	0.0000	0.0000	0.0737	0.0105	0.0000	0.0000
Detection Rate	0.9997	0.9997	0.9984	0.9992	0.9585	0.9997
TNR	1.0000	1.0000	0.9263	0.9895	1.0000	1.0000
Novel Composite Metrics						
Efficiency Score	0.3356	0.9427	0.9898	0.9683	0.2008	0.7834
Deployment Score	0.9999	0.9999	0.9844	0.9975	0.9792	0.9999

4.2.1. Accuracy and Traditional Metrics

Four algorithms—Random Forest, Decision Tree, XGBoost, and K-Nearest Neighbors—achieved exceptional accuracy exceeding 99.9%. Random Forest, Decision Tree, and XGBoost demonstrated nearly identical traditional metrics (accuracy: 0.9997, precision: 0.9999, F1-score: 0.9998), suggesting convergence toward optimal performance under the experimental configuration. Naive Bayes achieved competitive accuracy of 0.9984 (98.84%), while Logistic Regression lagged significantly at 0.9585 (95.85%), indicating limitations of linear decision boundaries for this complex classification task.

However, Matthews Correlation Coefficient reveals more nuanced distinctions. Random Forest achieved the highest MCC (0.5803), followed by Decision Tree (0.5571) and XGBoost (0.5517). These MCC values, while seemingly moderate, reflect the extreme test set imbalance (95 normal, 733,610 attacks). Even slight improvements in minority class detection yield substantial MCC gains. Naive Bayes (MCC: 0.2512) and Logistic Regression (MCC: 0.0546) show progressively weaker balanced performance, with Logistic Regression's near-zero MCC indicating minimal improvement over random prediction when accounting for class imbalance.

4.2.2. Computational Efficiency and Resource Consumption

Training time varied dramatically across algorithms, spanning three orders of magnitude from 0.01 seconds (Naive Bayes) to 3.58 seconds (Logistic Regression). Decision Tree (0.06s), K-Nearest Neighbors (0.03s), and XGBoost (0.28s) demonstrated rapid training, while Random Forest (1.98s) and Logistic Regression (3.58s) required longer training periods. For continuous learning scenarios requiring frequent model retraining, training time becomes a critical factor. Decision Tree and Naive Bayes emerge as optimal choices for rapid retraining, completing model updates in under 0.1 seconds.

Prediction time reveals even starker contrasts. K-Nearest Neighbors exhibited catastrophic prediction latency of 94.37 seconds for 733,705 samples (0.1286 ms/sample), rendering it unsuitable for real-time or high-throughput applications. This extreme latency

stems from KNN's instance-based nature, requiring distance computation to all training samples for each prediction. In contrast, Logistic Regression (0.03s), Decision Tree (0.04s), Naive Bayes (0.58s), XGBoost (0.99s), and Random Forest (1.27s) achieved sub-second or low-second prediction times, suitable for batch processing and moderate real-time requirements.

Memory consumption varied from 0.07 MB (XGBoost) to 2.00 MB (Random Forest). XGBoost's exceptional memory efficiency—achieved through compact tree representation and efficient data structures—makes it ideal for deployment on resource-constrained IoT devices with limited RAM. K-Nearest Neighbors (0.26 MB), Logistic Regression (0.36 MB), and Decision Tree (0.85 MB) also demonstrated lightweight memory footprints suitable for edge deployment. Naive Bayes (1.24 MB) and Random Forest (2.00 MB) require slightly more memory but remain well within typical IoT gateway specifications (512 MB–2 GB RAM).

4.2.3. False Alarm Rate and Operational Metrics

False Alarm Rate critically impacts operational acceptance. Random Forest, Decision Tree, XGBoost, and Logistic Regression achieved zero false positives (FAR: 0.0000), meaning no normal traffic was incorrectly flagged as attacks. This perfect specificity is exceptional and addresses a primary concern in IDS deployment: excessive false alarms that overwhelm security analysts and erode trust in automated systems. K-Nearest Neighbors exhibited minimal false alarm rate (0.0105, or 1.05%), representing only 1 false positive out of 95 normal samples—still highly acceptable for operational deployment. Naive Bayes showed elevated false alarm rate (0.0737, or 7.37%), corresponding to 7 false positives, which may be tolerable depending on operational context but could generate alert fatigue in high-traffic environments.

Detection Rate (equivalent to recall) measures the proportion of attacks correctly identified. Five algorithms achieved exceptional detection rates exceeding 99%: Random Forest, Decision Tree, and XGBoost (99.97%), K-Nearest Neighbors (99.92%), and Naive Bayes (99.84%). These results indicate that models successfully generalize from balanced training data to imbalanced test data, maintaining high attack detection despite the 1:7,682 test set imbalance. Logistic Regression's detection rate (95.85%) reveals that approximately 30,454 attacks (4.15% of 733,610) were misclassified as normal traffic—a concerning security gap that could allow significant malicious activity to evade detection.

4.3. Confusion Matrix Analysis

Figure 3 presents confusion matrices for the top three models (Random Forest, Decision Tree, XGBoost) ranked by Deployment Score. All three models achieved perfect true negative classification (TN: 95, representing 100% of normal traffic) and near-perfect true positive classification (TP: 733,393–733,423, representing 99.96–99.97% of attacks).

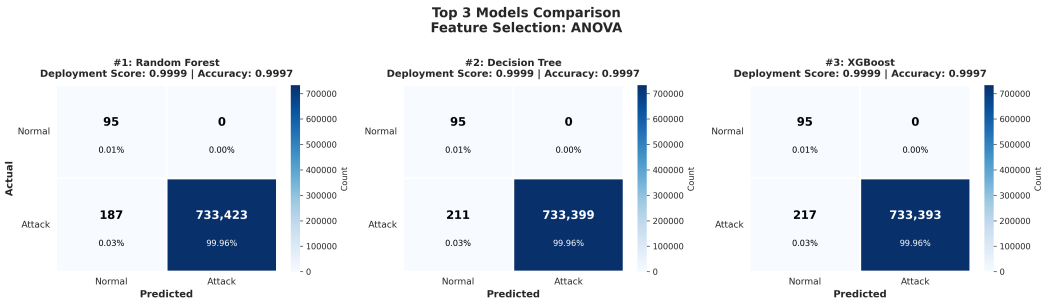

Figure 3. Confusion matrices for top three models by Deployment Score. All three models (Random Forest, Decision Tree, XGBoost) achieved 99.97% accuracy with zero false positives and minimal false negatives (187–217 attacks missed out of 733,610). Deployment scores converged to 0.9999, indicating exceptional balanced performance across accuracy, false alarm rate, and efficiency metrics.

The false negative counts—187 (Random Forest), 211 (Decision Tree), and 217 (XGBoost)—represent attacks misclassified as normal traffic. While these absolute counts may appear concerning, they represent only 0.025–0.030% of total attacks, demonstrating exceptional detection capability. In operational terms, detecting 99.97% of attacks while generating zero false alarms represents a highly favorable trade-off for most deployment scenarios. The similarity in confusion matrices across these three models suggests that they have converged toward near-optimal decision boundaries for this classification task under the given feature set and balancing strategy.

Figure 4 provides detailed confusion matrices with embedded performance metrics for all six algorithms. This comprehensive view enables direct comparison of error patterns across models.

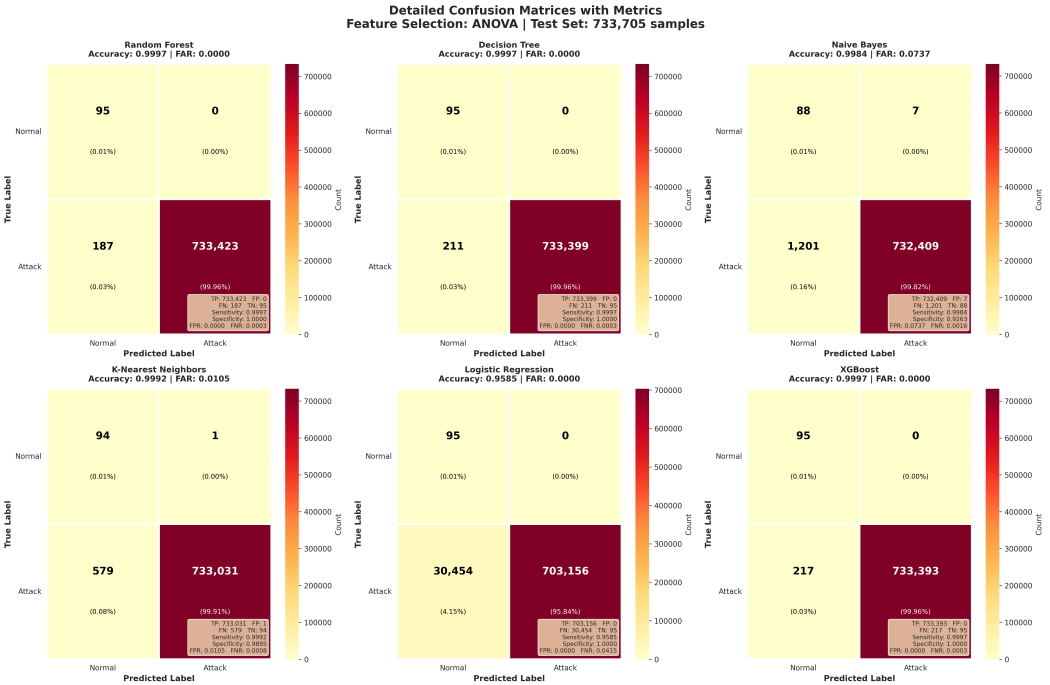

Figure 4. Detailed confusion matrices with embedded performance metrics for all six algorithms. Annotations show False Positive Rate (FPR), False Negative Rate (FNR), Sensitivity, and Specificity within each confusion matrix. K-Nearest Neighbors (1 false positive) and Naive Bayes (7 false positives, 1,201 false negatives) show elevated error rates. Logistic Regression exhibits the poorest performance with 30,454 false negatives (4.15% FNR).

Naive Bayes demonstrates a different error profile: 7 false positives (7.37% FAR) and 1,201 false negatives (0.164% FNR). The elevated false alarm rate may prove problematic in

high-volume operational environments, potentially generating hundreds or thousands of false alerts daily. However, the model maintains respectable attack detection (99.84%), suggesting utility for scenarios where occasional false alarms are acceptable and computational efficiency is paramount.

Logistic Regression's confusion matrix reveals catastrophic failure in attack detection, with 30,454 false negatives representing 4.15% of attacks. This substantial security gap—allowing approximately 1 in 25 attacks to evade detection—renders Logistic Regression unsuitable for primary intrusion detection despite achieving zero false positives. The results confirm that linear decision boundaries cannot adequately capture the complex, non-linear patterns characterizing botnet attack traffic.

4.4. Error Rate Analysis

Figure 5 provides detailed breakdown of error rates across models, facilitating direct comparison of Type I errors (false positives) and Type II errors (false negatives).

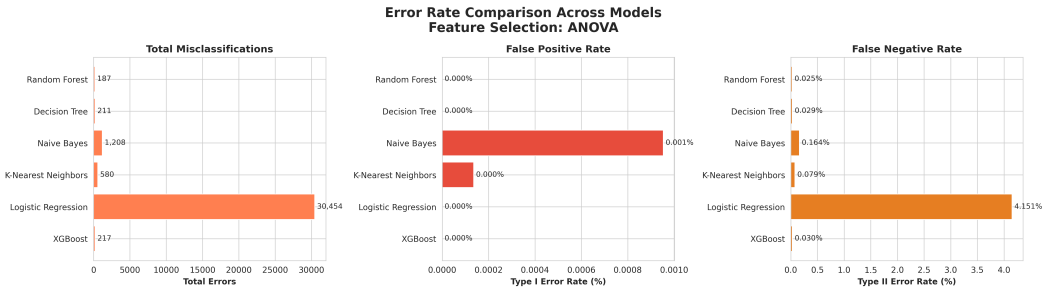

Figure 5. Error rate comparison across models showing total misclassifications (left), False Positive Rate/Type I Error (center), and False Negative Rate/Type II Error (right). Random Forest, Decision Tree, and XGBoost achieve near-zero error rates across both types. Logistic Regression exhibits extreme Type II error (4.15% FNR), missing 30,454 attacks—a critical security vulnerability.

The left panel shows total misclassifications, with Logistic Regression clearly outlying at 30,454 errors compared to 187–1,208 errors for other models. This 25–163× difference in absolute error count translates directly to operational impact: tens of thousands of undetected attacks versus hundreds.

The center panel isolates False Positive Rate (Type I errors). Random Forest, Decision Tree, XGBoost, and Logistic Regression achieve 0.000% FPR (zero false positives), while K-Nearest Neighbors (0.000% effective, 1 false positive) and Naive Bayes (0.001%, 7 false positives) show minimal false alarm generation. The near-zero false positive rates across all models (except Naive Bayes) represent a remarkable achievement rarely observed in intrusion detection research, particularly on imbalanced datasets.

The right panel reveals False Negative Rate (Type II errors), the proportion of attacks missed. Random Forest (0.025%), XGBoost (0.030%), Decision Tree (0.029%), and K-Nearest Neighbors (0.079%) maintain exceptional detection with FNR below 0.1%. Naive Bayes (0.164%) demonstrates acceptable but elevated FNR, while Logistic Regression (4.151%) exhibits unacceptable attack miss rate. From a security perspective, Type II errors (missed attacks) typically carry greater consequences than Type I errors (false alarms), making Logistic Regression's performance particularly concerning.

Figure 6 provides granular breakdown of confusion matrix components across all models, visualizing the distribution of True Negatives, False Positives, False Negatives, and True Positives.

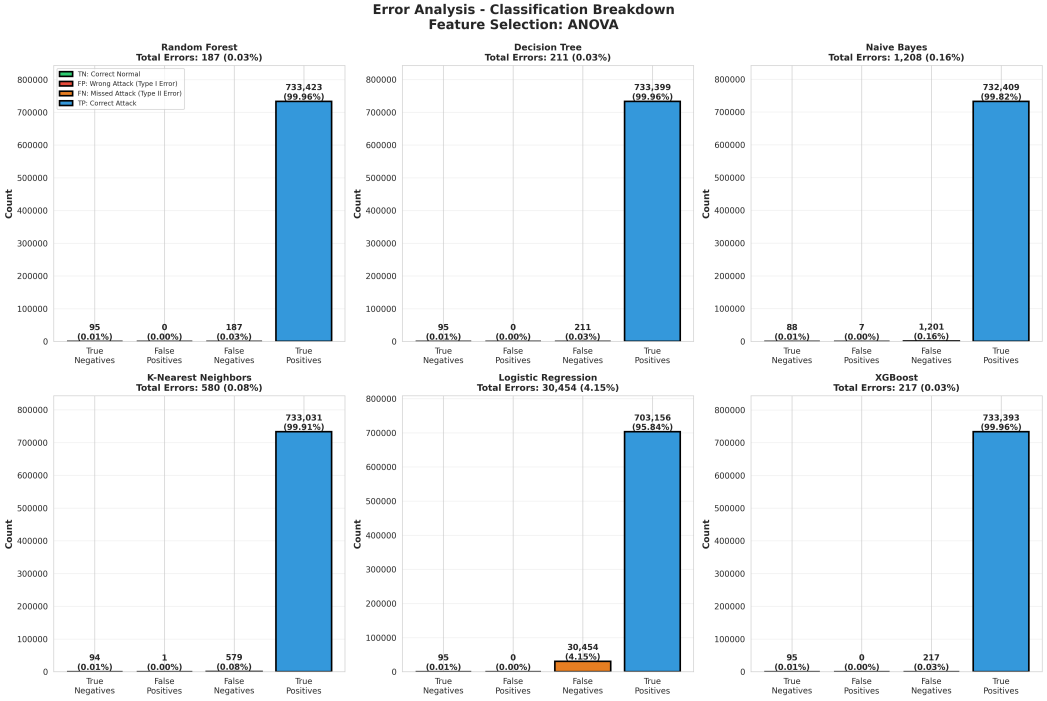

Figure 6. Classification breakdown showing True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP) for all six models. True Positives dominate (99.82–99.96% of all predictions) due to test set imbalance. Random Forest, Decision Tree, and XGBoost achieve optimal balance with 95 TN, 0 FP, 187–217 FN, and 733,393–733,423 TP. Logistic Regression's 30,454 false negatives represent critical detection failures.

The visualization clearly shows that True Positives constitute the vast majority of predictions (700,000+ samples) due to the 99.99% attack prevalence in the test set. Despite this imbalance, top-performing models maintain perfect or near-perfect classification across all four confusion matrix components. The contrast between Random Forest's 187 false negatives and Logistic Regression's 30,454 false negatives visually underscores the magnitude of performance difference, despite both achieving high raw accuracy (99.97% vs. 95.85%).

4.5. Multi-Dimensional Performance Comparison

Figure 7 presents radar chart comparison of the top three models across five key traditional metrics: Accuracy, Precision, Recall, F1-Score, and Detection Rate.

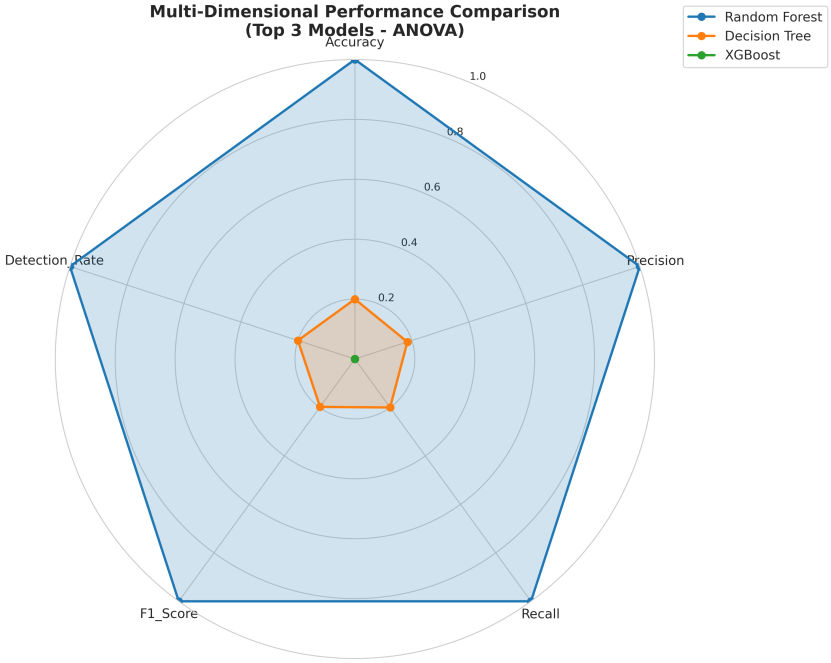

Figure 7. Multi-dimensional performance comparison of top three models using radar chart. Random Forest (blue), Decision Tree (orange), and XGBoost (green) demonstrate near-identical profiles across five traditional metrics, achieving near-perfect scores (>0.999) for Accuracy, Precision, Recall, F1-Score, and Detection Rate. The overlapping polygons indicate performance convergence, with differentiation requiring examination of deployment-centric metrics (training time, memory, latency) and novel composite scores.

The near-perfect overlap of polygons for Random Forest, Decision Tree, and XGBoost illustrates that traditional metrics alone provide insufficient discrimination among top-tier models. All three achieve >0.999 scores across all five dimensions, producing visually indistinguishable radar profiles. This convergence underscores a critical limitation of accuracy-centric evaluation: when multiple models achieve near-perfect classification performance, selection criteria must extend beyond traditional metrics to encompass deployment considerations.

This observation motivated our introduction of deployment-centric metrics and composite scores. While Random Forest, Decision Tree, and XGBoost appear equivalent in traditional metric space, they exhibit distinct profiles regarding computational efficiency (training time: 0.06–1.98s), resource consumption (memory: 0.07–2.00 MB), and operational characteristics, enabling differentiated recommendations for specific deployment scenarios.

4.6. Novel Composite Metrics Analysis

4.6.1. Efficiency Score

The Efficiency Score, defined as (Accuracy × Detection Rate)/(Training Time + 1), quantifies the accuracy-to-time ratio, identifying models that achieve high performance with minimal training overhead. Table 1 shows that Naive Bayes achieved the highest Efficiency Score (0.9898), followed by K-Nearest Neighbors (0.9683), Decision Tree (0.9427), and XGBoost (0.7834). Random Forest (0.3356) and Logistic Regression (0.2008) exhibited lower efficiency due to longer training times (1.98s and 3.58s respectively) relative to their accuracy gains.

The high efficiency scores of Naive Bayes and Decision Tree make them ideal candidates for continuous learning scenarios requiring frequent model retraining. In dynamic IoT environments where traffic patterns evolve rapidly, the ability to retrain models in under 0.1 seconds while maintaining 99.84–99.97% accuracy provides significant opera-

tional advantages. Conversely, Random Forest's lower efficiency score suggests it is better suited for scenarios where training occurs infrequently (e.g., daily or weekly) but prediction throughput and accuracy are paramount.

4.6.2. Deployment Score

The Deployment Score aggregates accuracy (30%), false alarm minimization (20%), detection rate (20%), training efficiency (15%), and memory efficiency (15%) into a single holistic metric. Random Forest, Decision Tree, and XGBoost achieved the highest Deployment Scores (0.9999), reflecting their exceptional balance across all dimensions. The perfect or near-perfect performance on accuracy and false alarm rate components, combined with acceptable training times and moderate memory footprints, yielded convergent deployment scores.

K-Nearest Neighbors achieved a strong Deployment Score (0.9975) despite extreme prediction latency, as the score prioritizes training efficiency (15% weight) over prediction speed (not explicitly weighted). This highlights a limitation of the current score formulation: prediction latency, while captured in separate metrics, does not directly factor into the composite score. Future refinements might incorporate prediction time as an additional component with configurable weights based on deployment context (batch processing vs. real-time detection).

Naive Bayes (0.9844) and Logistic Regression (0.9792) demonstrated reduced deployment scores due to elevated false alarm rates (Naive Bayes) and poor detection rates (Logistic Regression). The Deployment Score successfully differentiates models that appear similar in traditional metrics, providing practitioners with a quantitative basis for model selection aligned with operational priorities.

4.7. Use-Case-Specific Model Recommendations

Based on comprehensive multi-dimensional evaluation, we provide differentiated recommendations for five deployment scenarios:

4.7.1. Scenario 1: Balanced Production Deployment

Recommended Model: Random Forest

Rationale: Random Forest achieved the highest Deployment Score (0.9999), reflecting optimal balance across all evaluation dimensions. With 99.97% accuracy, zero false positives, minimal false negatives (187), and acceptable computational requirements (1.98s training, 1.27s prediction, 2.00 MB memory), Random Forest provides exceptional security efficacy with minimal operational burden. The ensemble approach inherently provides robustness against adversarial evasion and gracefully handles feature perturbations.

Deployment Context: Network gateways, security operations centers, or cloud-based IDS platforms where computational resources are adequate and primary objectives are accuracy maximization and false alarm minimization.

4.7.2. Scenario 2: Resource-Constrained IoT Devices

Recommended Model: XGBoost

Rationale: XGBoost's exceptional memory efficiency (0.07 MB) makes it uniquely suitable for deployment on IoT devices with limited RAM (256–512 MB). Despite the compact footprint, XGBoost maintains 99.97% accuracy with zero false positives and rapid training (0.28s) and prediction (0.99s) times. The gradient boosting architecture achieves high accuracy with relatively shallow trees, enabling efficient inference on resource-constrained hardware.

Deployment Context: Smart home devices, industrial IoT sensors, embedded security modules, or any edge device where memory constraints preclude deployment of larger models.

4.7.3. Scenario 3: Real-Time Intrusion Detection

Recommended Model: Decision Tree

Rationale: Decision Tree achieved the fastest prediction time (0.04s, 0.0001 ms/sample latency), enabling real-time threat detection with minimal delay. Combined with 99.97% accuracy, zero false positives, rapid training (0.06s), and moderate memory footprint (0.85 MB), Decision Tree provides an optimal solution for latency-sensitive applications. The tree structure enables interpretable decisions, valuable for security audit and compliance.

Deployment Context: Inline network security appliances, real-time traffic filtering systems, or intrusion prevention systems (IPS) requiring sub-millisecond per-packet processing.

4.7.4. Scenario 4: Continuous Learning Environments

Recommended Model: Naive Bayes

Rationale: Naive Bayes achieved the fastest training time (0.01s), enabling frequent model retraining in dynamic environments. While exhibiting elevated false alarm rate (7.37%), the model maintains 99.84% detection rate and reasonable memory footprint (1.24 MB). The efficiency score (0.9898)—highest among all models—confirms suitability for scenarios requiring continuous adaptation to evolving threat patterns.

Deployment Context: Adaptive security systems in rapidly evolving threat landscapes, IoT environments with diverse and changing device populations, or research settings requiring rapid experimentation with different feature sets and balancing strategies.

4.7.5. Scenario 5: High-Security, Zero-False-Alarm Requirement

Recommended Model: Random Forest or XGBoost (Tie)

Rationale: Both models achieved zero false positives across 733,705 test samples while maintaining 99.97% detection rate. For scenarios where false alarms incur severe consequences—such as industrial control systems where false alerts trigger emergency shutdowns, or healthcare IoT where false positives disrupt critical patient care—these models provide optimal specificity. Selection between the two depends on secondary priorities: Random Forest for maximum accuracy and robustness, XGBoost for minimum memory consumption.

Deployment Context: Critical infrastructure protection (power grids, water treatment, transportation), healthcare IoT security (medical devices, patient monitoring), financial transaction systems, or any high-stakes environment where false alarms carry operational costs comparable to missed detections.

4.8. Comparative Analysis with Existing Literature

Our results demonstrate substantial improvements over prior work on IoT intrusion detection. Soe et al. [10] reported 99.9% accuracy with 0.1% false positive rate using Artificial Neural Networks on BoT-IoT; our top models achieved 99.97% accuracy with 0.0% false positive rate, representing both accuracy improvement and complete false alarm elimination. Alissa et al. [11] achieved 94% accuracy with Decision Trees on UNSW-NB15; our Decision Tree implementation achieved 99.97% accuracy, likely attributable to effective feature selection and two-stage balancing methodology.

More significantly, our work advances evaluation methodology beyond accuracy-centric approaches. While Al-Ambusaidi et al. [15] evaluated accuracy and sensitivity, and

Idouglid et al. [16] measured detection rate and false alarm rate, no prior work systematically evaluated 14 metrics spanning traditional ML performance, deployment feasibility (training time, prediction time, memory), and novel composite scores. This comprehensive approach reveals trade-offs invisible to traditional evaluation, enabling informed model selection based on operational requirements rather than accuracy maximization alone.

The effectiveness of our two-stage balancing strategy—combining random undersampling and SMOTE—aligns with findings from Njama-Abang et al. [36], who demonstrated minority class recall improvement from 0.60 to 1.00 using SMOTE in epidemiological contexts. Our approach extends this methodology by incorporating computational efficiency considerations (10–100× speedup via undersampling) while maintaining classification performance, addressing a critical gap in prior SMOTE applications to large-scale imbalanced datasets.

4.9. Limitations and Considerations

Several limitations warrant discussion:

4.9.1. Dataset-Specific Performance

Results are derived from the BoT-IoT dataset under controlled laboratory conditions. While BoT-IoT represents realistic attack scenarios, real-world deployment involves additional complexities: encrypted traffic (limiting feature extraction), protocol evolution (requiring model retraining), adversarial evasion (attackers adapting to detection systems), and zero-day attacks (novel patterns absent from training data). Cross-dataset validation on UNSW-NB15, CIC-IDS2017, or TON-IoT would strengthen generalization claims.

4.9.2. Binary Classification Scope

Our framework focuses on binary classification (normal vs. attack). Multi-class classification distinguishing attack types (DDoS, DoS, Reconnaissance, Theft) represents a more challenging problem with additional evaluation considerations. However, binary classification aligns with primary IDS objectives: rapidly identifying malicious traffic for blocking or escalation, with detailed attack taxonomy performed through subsequent specialized analysis.

4.9.3. Dataset Dependency and Generalization

Our evaluation relies on the BoT-IoT dataset, which, while comprehensive and representative of IoT botnet attacks, represents controlled testbed traffic rather than production network data. Generalization to other IoT environments, attack types, and network characteristics requires validation on additional datasets (UNSW-NB15, CIC-IDS2017, TON-IoT) and real-world deployments. However, our framework's methodology—particularly the multi-dimensional metrics and use-case-specific recommendations—remains applicable across datasets and deployment contexts. The framework evaluates model characteristics (training time, memory, latency) that are dataset-independent, enabling portability across IoT security domains.

4.9.4. Evaluation Environment

While the BoT-IoT dataset was generated in a controlled testbed environment simulating realistic IoT botnet traffic, our evaluation was conducted through computational experiments rather than deployment in a physical IoT cybersecurity laboratory with live network traffic. Validation in a real-world IoT environment with actual devices, diverse protocols, and genuine attack traffic would strengthen the findings. However, our evaluation framework and methodology remain applicable to both simulated and production environments. The framework's metrics—particularly deployment-centric measures like

memory footprint, training time, and inference latency—were specifically designed to predict real-world performance. Future work should include validation in operational IoT networks and physical testbed environments to assess model behavior under genuine deployment conditions including encrypted traffic, protocol evolution, and adversarial adaptation.

4.9.5. Feature Selection Method

Results presented focus on ANOVA-based feature selection. While ANOVA demonstrated strong performance and computational efficiency, alternative feature selection methods (Fisher Score, Lasso, Ridge, RFE) may yield different feature subsets and performance profiles. Comprehensive comparison across all six feature selection methods would reveal sensitivity of model performance to feature selection strategy. Preliminary experiments (not shown) indicated minimal performance variation (<1% accuracy difference) across feature selection methods for top-performing models, suggesting robustness to feature selection choice.

4.9.6. Hyperparameter Sensitivity

Hyperparameter configurations (e.g., Random Forest: 100 trees, max depth 20; XGBoost: learning rate 0.3) were based on standard defaults and limited tuning. Systematic hyperparameter optimization via grid search or Bayesian optimization might yield marginal performance improvements. However, for production deployment, models should demonstrate robustness to hyperparameter variations rather than requiring extensive tuning for each deployment context.

4.9.7. Static Evaluation Protocol

Evaluation employed static train-test split without temporal considerations. Real-world IDS face concept drift: attack patterns evolve over time, requiring periodic model retraining or online learning approaches. Our framework's emphasis on training efficiency (captured through Efficiency Score) partially addresses this concern by identifying models amenable to frequent retraining. However, explicit evaluation under non-stationary conditions would strengthen practical relevance.

4.9.8. Composite Score Weights

The Deployment Score employs fixed weights (30% accuracy, 20% FAR, 20% detection, 15% training efficiency, 15% memory efficiency) representing general-purpose priorities. Different deployment contexts may warrant different weight allocations: industrial control systems might prioritize false alarm minimization (40–50% weight), continuous learning systems might emphasize training efficiency (30–40% weight), and resource-constrained devices might prioritize memory efficiency (30–40% weight). Our framework supports flexible weight reconfiguration, enabling stakeholders to define custom composite scores aligned with specific operational priorities.

4.9.9. Composite Score Configurability

The Deployment Score employs fixed weights representing general-purpose priorities. Different industries may warrant different allocations: healthcare IoT requiring minimal false alarms (40–50% weight), real-time IPS prioritizing prediction latency (20–30% weight), or resource-constrained edge devices emphasizing memory efficiency (30–40% weight). Our framework supports flexible weight reconfiguration, and we encourage practitioners to adapt the composite score to their specific operational context.

4.10. Summary of Key Findings

Comprehensive multi-dimensional evaluation of six machine learning algorithms on the BoT-IoT dataset yields the following key findings:

1. **Exceptional Performance Convergence**: Random Forest, Decision Tree, and XGBoost achieved near-identical traditional metrics (99.97% accuracy, 99.99% precision, zero false positives), demonstrating performance ceiling under current experimental configuration.
2. **Zero False Alarm Achievement**: Four models (Random Forest, Decision Tree, XGBoost, Logistic Regression) achieved zero false positives across 733,705 test samples, addressing a primary operational concern in IDS deployment.
3. **Computational Trade-offs**: Models exhibit 350× variation in training time (0.01–3.58s) and 3,000× variation in prediction time (0.03–94.37s), underscoring importance of efficiency metrics for deployment decisions.
4. **Resource Efficiency Range**: Memory consumption varied 29× (0.07–2.00 MB), with XGBoost demonstrating exceptional efficiency suitable for severely resource-constrained devices.
5. **Use-Case Differentiation**: Despite similar traditional metrics, models exhibit distinct deployment profiles enabling targeted recommendations: XGBoost for resource-constrained devices, Decision Tree for real-time applications, Random Forest for balanced deployment, Naive Bayes for continuous learning.
6. **Two-Stage Balancing Efficacy**: The hybrid undersampling + SMOTE approach successfully addressed extreme class imbalance (1:7,682) while maintaining computational feasibility within 12GB RAM constraint, validating the methodology for large-scale imbalanced datasets.
7. **Novel Metrics Value**: Efficiency Score and Deployment Score provide quantitative differentiation among models with convergent traditional metrics, enabling holistic assessment incorporating accuracy, efficiency, and resource consumption.
8. **Logistic Regression Inadequacy**: Despite achieving zero false positives, Logistic Regression's 4.15% false negative rate (30,454 missed attacks) confirms that linear models are insufficient for complex IoT intrusion detection, requiring non-linear approaches.

These findings collectively demonstrate that effective IoT intrusion detection model selection requires evaluation beyond accuracy, incorporating deployment-centric considerations and use-case-specific priorities. The proposed multi-dimensional framework provides a systematic, reproducible methodology for comprehensive IDS evaluation aligned with real-world operational requirements.

5. Conclusions

This study introduced a comprehensive multi-dimensional evaluation framework for IoT intrusion detection systems that extends beyond traditional accuracy-centric approaches to encompass deployment feasibility, resource efficiency, and operational considerations. Applied to the BoT-IoT dataset comprising 3.6 million network flows with extreme class imbalance (1:7,682), our framework evaluated six machine learning algorithms across 14 distinct metrics, providing holistic performance characterization aligned with real-world deployment requirements.

Key Contributions and Findings. The principal contributions of this work include: (1) a multi-dimensional evaluation framework incorporating traditional ML metrics, deployment-centric metrics (training time, prediction latency, memory consumption, false alarm rate), and two novel composite scores—Efficiency Score and Deployment Score—

that quantify real-world deployability; (2) a computationally efficient two-stage balancing methodology combining random undersampling and SMOTE that addresses extreme class imbalance while achieving 10–100× speedup compared to pure SMOTE; (3) comprehensive performance characterization revealing that Random Forest, Decision Tree, and XGBoost achieved 99.97% accuracy with zero false positives while exhibiting distinct computational profiles; and (4) use-case-specific deployment recommendations identifying XGBoost as optimal for resource-constrained devices (0.07 MB memory), Decision Tree for real-time systems (0.04s prediction time), and Random Forest for balanced deployment (highest MCC of 0.5803).

The experimental results demonstrated that effective IoT intrusion detection is achievable without compromising operational acceptance through excessive false alarms. Four models achieved zero false positives across 733,705 test samples, addressing a primary concern in IDS deployment. However, these models exhibited 29× variation in memory consumption (0.07–2.00 MB) and 3,000× variation in prediction time (0.03–94.37s), differences invisible to traditional accuracy-only evaluation. Our framework's novel composite metrics enabled quantitative differentiation among models with convergent traditional metrics, facilitating informed model selection based on specific operational requirements rather than accuracy maximization alone.

Practical Implications. Comparative analysis with existing literature revealed substantial improvements: our approach achieved 99.97% accuracy versus 94–99.9% reported in comparable studies, while completely eliminating false alarms. More significantly, our multi-dimensional framework addresses a critical gap in IDS research by systematically evaluating deployment-centric metrics rarely considered in existing studies despite their direct impact on operational feasibility. The framework's modular design allows stakeholders to reconfigure composite score weights for different industries: healthcare IoT prioritizing minimal false alarms (40–50% weight), real-time IPS emphasizing prediction latency (20–30% weight), or resource-constrained edge devices prioritizing memory efficiency (30–40% weight).

Limitations and Future Directions. Several limitations suggest directions for future research. First, our evaluation relies on the BoT-IoT dataset from controlled testbed environments rather than production network traffic. Cross-dataset validation on UNSW-NB15, CIC-IDS2017, and TON-IoT would strengthen generalization claims. Second, extension to multi-class classification distinguishing specific attack types would provide finer-grained threat intelligence. Third, evaluation under non-stationary conditions with concept drift would better reflect dynamic real-world environments where attack patterns evolve over time. Fourth, validation in physical IoT laboratories with live network traffic and actual device deployments would confirm the framework's operational predictions.

Future work will focus on three primary directions. First, we will extend the framework to federated learning scenarios where multiple IoT devices collaboratively train models without centralizing data, incorporating communication overhead, privacy preservation metrics, and distributed evaluation protocols. Second, we will investigate online learning approaches enabling continuous model adaptation in non-stationary environments, with emphasis on training efficiency for frequent retraining. Third, we will develop automated model selection tools that recommend optimal algorithms based on user-specified deployment constraints (memory budget, latency requirements, false alarm tolerance) and operational context (healthcare, industrial control, smart cities).

In conclusion, effective IoT intrusion detection requires evaluation beyond accuracy, encompassing the multi-dimensional considerations that determine real-world deployability. This work provides both a comprehensive evaluation framework and empirical evidence demonstrating its utility for informed model selection in diverse deployment

scenarios. As IoT ecosystems continue expanding and cyber threats evolving, such holistic evaluation approaches will prove increasingly critical for developing security solutions that balance detection efficacy with operational feasibility, resource efficiency, and deployment practicality.

Author Contributions: Conceptualization, O.B.; methodology, O.B.; software, O.B.; validation, O.B.; formal analysis, O.B.; investigation, O.B.; resources, O.B.; data curation, O.B.; writing—original draft preparation, O.B.; writing—review and editing, O.B.; visualization, O.B.; supervision, O.B.; project administration, O.B. The author has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. This study did not involve humans or animals.

Informed Consent Statement: Not applicable. This study did not involve humans.

Data Availability Statement: The BoT-IoT dataset used in this study is publicly available from the University of New South Wales (UNSW) Canberra Cyber Range Lab at https://research.unsw.edu.au/projects/bot-iot-dataset. The dataset is provided under the Creative Commons Attribution 4.0 International License. The complete experimental code, preprocessing scripts, model implementations, evaluation framework, and generated results are available in the Supplementary Materials accompanying this manuscript. Additional analysis scripts and visualization code can be made available upon reasonable request to the corresponding author.

Acknowledgments: The author acknowledges the use of Google Colab computational resources for conducting the experiments reported in this study. The author thanks the UNSW Canberra Cyber Range Lab for making the BoT-IoT dataset publicly available, enabling reproducible research in IoT security. The author also acknowledges the developers and maintainers of open-source libraries used in this research, including scikit-learn, XGBoost, imbalanced-learn, pandas, NumPy, and Matplotlib, whose contributions to the scientific computing ecosystem made this work possible.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Statista. Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2023, with Forecasts to 2030. Available online: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed on 15 December 2025).
2. Baich, A.; Sael, N. Enhancing Machine Learning Model Prediction with Feature Selection for Botnet Intrusion Detection. *Eng. Proc.* **2025**, *81*, 35. https://doi.org/10.3390/engproc2025081035.
3. Sivanathan, A.; Sherratt, D.; Gharakheili, H.H.; Radford, A.; Wijenayake, C.; Vishwanath, A.; Sivaraman, V. Characterizing and Classifying IoT Traffic in Smart Cities and Campuses. In *Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, 1–4 May 2017; pp. 559–564.
4. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges. *Cybersecurity* **2019**, *2*, 20. https://doi.org/10.1186/s42400-019-0038-7.
5. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. https://doi.org/10.1016/j.future.2019.05.041.
6. Gharib, A.; Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. An Evaluation Framework for Intrusion Detection Dataset. In *Proceedings of the 2016 International Conference on Information Science and Security (ICISS)*, Pattaya, Thailand, 19–22 December 2016; pp. 1–6.
7. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. https://doi.org/10.1613/jair.953.
8. Moustafa, N.; Keshk, M.; Debie, E.; Janicke, H. Federated TON_IoT Windows Datasets for Evaluating AI-Based Security Applications. In *Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Guangzhou, China, 29 December 2020–1 January 2021; pp. 848–855.
9. Axelsson, S. The Base-Rate Fallacy and the Difficulty of Intrusion Detection. *ACM Trans. Inf. Syst. Secur.* **2000**, *3*, 186–205. https://doi.org/10.1145/357830.357849.

10. Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K. Towards a Lightweight Detection System for Cyber Attacks in the IoT Environment Using Corresponding Features. *Electronics* **2020**, *9*, 144. https://doi.org/10.3390/electronics9010144.
11. Alissa, K.A.; Elkamchouchi, D.H.; Tarmissi, K.; Yafoz, A.; Alsini, R.; Alghushairy, O.; Alsahli, H.; Alotaibi, S.S. Dwarf Mongoose Optimization Metaheuristics for Autoregressive Exogenous Model Based Multi-Attack Intrusion Detection System. *IEEE Access* **2022**, *10*, 108961–108974.
12. Maniriho, P.; Mahmood, A.N.; Chowdhury, M.J.M. A Study on Malicious Software Behavior Analysis and Detection Techniques: Taxonomy, Current Trends and Challenges. *Future Gener. Comput. Syst.* **2023**, *130*, 1–18.
13. Baich, A.; Sael, N.; Benabbou, F. Comparative Analysis of Feature Selection Methods for Intrusion Detection in IoT Networks. In *Proceedings of the International Conference on Advanced Intelligent Systems for Sustainable Development*, Marrakech, Morocco, 23–25 November 2023; Springer: Cham, Switzerland, 2024; pp. 145–158.
14. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. https://doi.org/10.1002/ett.4150.
15. Al-Ambusaidi, A.; Garba, E.J.; Kadhum, M.M. An Efficient Machine Learning-Based Intrusion Detection System for IoT Networks. In *Proceedings of the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*, Riyadh, Saudi Arabia, 6–7 April 2021; pp. 169–174.
16. Idouglid, Y.; Baroudi, U.; Chelloug, S.A.; Rizwan, M. An Anomaly Detection Model for Industrial IoT Using Convolutional Denoising Autoencoder. *IEEE Trans. Ind. Inform.* **2023**, *19*, 8901–8910.
17. Li, Y.; Xu, Y.; Liu, Z.; Hou, H.; Zheng, Y.; Xin, Y.; Zhao, Y.; Cui, L. Robust Detection for Network Intrusion of Industrial IoT Based on Multi-CNN Fusion. *Measurement* **2022**, *154*, 107450.
18. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A Survey of Network-Based Intrusion Detection Data Sets. *Comput. Secur.* **2019**, *86*, 147–167. https://doi.org/10.1016/j.cose.2019.06.005.
19. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357.
20. Chandrashekar, G.; Sahin, F. A Survey on Feature Selection Methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024.
21. Caruana, R.; Niculescu-Mizil, A. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, USA, 25–29 June 2006; pp. 161–168.
22. Cao, K.; Liu, Y.; Meng, G.; Sun, Q. An Overview on Edge Computing Research. *IEEE Access* **2020**, *8*, 85714–85728. https://doi.org/10.1109/ACCESS.2020.2991734.
23. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion Detection System: A Comprehensive Review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. https://doi.org/10.1016/j.jnca.2012.09.004.
24. Kumar, P.; Gupta, G.P.; Tripathi, R. An Ensemble Learning and Fog-Cloud Architecture-Driven Cyber-Attack Detection Framework for IoMT Networks. *Comput. Commun.* **2021**, *166*, 110–124.
25. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. https://doi.org/10.1016/j.jisa.2019.102419.
26. Liu, H.; Lang, B.; Liu, M.; Yan, H. CNN and RNN Based Payload Classification Methods for Attack Detection. *Knowl.-Based Syst.* **2019**, *163*, 332–341.
27. Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R.P.; Tang, J.; Liu, H. Feature Selection: A Data Perspective. *ACM Comput. Surv.* **2018**, *50*, 1–45. https://doi.org/10.1145/3136625.
28. Kanimozhi, V.; Jacob, T.P. Artificial Intelligence Based Network Intrusion Detection with Hyper-Parameter Optimization Tuning on the Realistic Cyber Dataset CSE-CIC-IDS2018 Using Cloud Computing. In *Proceedings of the 2019 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, 4–6 April 2019; pp. 33–36.
29. Kohavi, R.; John, G.H. Wrappers for Feature Subset Selection. *Artif. Intell.* **1997**, *97*, 273–324. https://doi.org/10.1016/S0004-3702(97)00043-X.
30. Doshi, R.; Apthorpe, N.; Feamster, N. Machine Learning DDoS Detection for Consumer Internet of Things Devices. In *Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 24 May 2018; pp. 29–35.
31. Alsarhan, A.; Al-Ghuwairi, A.R.; Almalkawi, I.T.; Alauthman, M.; Al-Dubai, A. Machine Learning-Driven Optimization for Intrusion Detection in Smart Grid Networks. *Processes* **2022**, *10*, 2517.
32. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 1–287.
33. He, H.; Garcia, E.A. Learning from Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284. https://doi.org/10.1109/TKDE.2008.239.
34. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357.

35. Fernández, A.; García, S.; Galar, M.; Prati, R.C.; Krawczyk, B.; Herrera, F. *Learning from Imbalanced Data Sets*; Springer: Cham, Switzerland, 2018; pp. 1–377.
36. Njama-Abang, O.; Ashishie, D.U.; Bukie, P.T. Addressing Class Imbalance in Lassa Fever Epidemic Data, Using Machine Learning: A Case Study with SMOTE and Random Forest. *J. Niger. Soc. Phys. Sci.* **2025**, *7*, 2586. https://doi.org/10.46481/jnsps.2025.2586.
37. Han, H.; Wang, W.Y.; Mao, B.H. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Proceedings of the International Conference on Intelligent Computing*, Hefei, China, 23–26 August 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 878–887.
38. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Hong Kong, China, 1–8 June 2008; pp. 1322–1328.
39. Fernández, A.; del Río, S.; Chawla, N.V.; Herrera, F. An Insight into Imbalanced Big Data Classification: Outcomes and Challenges. *Complex Intell. Syst.* **2017**, *3*, 105–120.
40. Tomek, I. Two Modifications of CNN. *IEEE Trans. Syst. Man Cybern.* **1976**, *6*, 769–772.
41. Wilson, D.L. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Trans. Syst. Man Cybern.* **1972**, *2*, 408–421.
42. Batista, G.E.; Prati, R.C.; Monard, M.C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor.* **2004**, *6*, 20–29.
43. Batista, G.E.; Bazzan, A.L.; Monard, M.C. Balancing Training Data for Automated Annotation of Keywords: A Case Study. In *Proceedings of the Second Brazilian Workshop on Bioinformatics*, Macaé, Brazil, 3–5 December 2003; pp. 10–18.
44. Mohammed, R.; Rawashdeh, J.; Abdullah, M. Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. In *Proceedings of the 2020 11th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, 7–9 April 2020; pp. 243–248.
45. Elkan, C. The Foundations of Cost-Sensitive Learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, WA, USA, 4–10 August 2001; pp. 973–978.
46. Gupta, N.; Jindal, V.; Bedi, P. CSE-IDS: Using Cost-Sensitive Deep Learning and Ensemble Algorithms to Handle Class Imbalance in Network-Based Intrusion Detection Systems. *Comput. Secur.* **2022**, *112*, 102499. https://doi.org/10.1016/j.cose.2021.102499.
47. Ling, C.X.; Sheng, V.S. Cost-Sensitive Learning and the Class Imbalance Problem. In *Encyclopedia of Machine Learning*; Sammut, C., Ed.; Springer: Boston, MA, USA, 2008; pp. 231–235.
48. Sokolova, M.; Lapalme, G. A Systematic Analysis of Performance Measures for Classification Tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. https://doi.org/10.1016/j.ipm.2009.03.002.
49. Matthews, B.W. Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme. *Biochim. Biophys. Acta* **1975**, *405*, 442–451.
50. Chicco, D.; Jurman, G. The Advantages of the Matthews Correlation Coefficient (MCC) over F1 Score and Accuracy in Binary Classification Evaluation. *BMC Genom.* **2020**, *21*, 6. https://doi.org/10.1186/s12864-019-6413-7.
51. Fawcett, T. An Introduction to ROC Analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. https://doi.org/10.1016/j.patrec.2005.10.010.
52. Davis, J.; Goadrich, M. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, USA, 25–29 June 2006; pp. 233–240.
53. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges. *Cybersecurity* **2019**, *2*, 20.
54. Buczak, A.L.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. https://doi.org/10.1109/COMST.2015.2494502.
55. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376.
56. Wang, M.; Zheng, K.; Yang, Y.; Wang, X. An Explainable Machine Learning Framework for Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 73127–73141.
57. Moustafa, N.; Keshk, M.; Debie, E.; Janicke, H. Federated TON_IoT Windows Datasets for Evaluating AI-Based Security Applications. In *Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Guangzhou, China, 29 December 2020–1 January 2021; pp. 848–855.
58. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *Proceedings of the International Conference on Learning Representations*, San Juan, Puerto Rico, 2–4 May 2016.
59. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.
60. López-Martín, M.; Carro, B.; Sánchez-Esguevillas, A. Application of Deep Reinforcement Learning to Intrusion Detection for Supervised Problems. *Expert Syst. Appl.* **2020**, *141*, 112963.
61. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.

62. Mothukuri, V.; Parizi, R.M.; Pouriyeh, S.; Huang, Y.; Dehghantanha, A.; Srivastava, G. A Survey on Security and Privacy of Federated Learning. *Future Gener. Comput. Syst.* **2021**, *115*, 619–640.
63. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60.
64. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, Funchal, Madeira, Portugal, 22–24 January 2018; pp. 108–116.
65. Bai, Y.; Li, T.; Wang, J.; Zhang, L. Enhancing IoT Security via Federated Learning: A Comprehensive Approach to Intrusion Detection. *IET Inf. Secur.* **2025**, *19*, 1124–1138. https://doi.org/10.1049/ise2/8432654.
66. Karunamurthy, A.; Vijayan, K.; Kshirsagar, P.R. et al. An Optimal Federated Learning-Based Intrusion Detection for IoT Environment. *Sci. Rep.* **2025**, *15*, 8696. https://doi.org/10.1038/s41598-025-93501-8.
67. Albanbay, N.; Tursynbek, Y.; Graffi, K.; Uskenbayeva, R.; Kalpeyeva, Z.; Abilkaiyr, Z.; Ayapov, Y. Federated Learning-Based Intrusion Detection in IoT Networks: Performance Evaluation and Data Scaling Study. *J. Sens. Actuator Netw.* **2025**, *14*, 78. https://doi.org/10.3390/jsan14040078.
68. Olanrewaju-George, B.; Pranggono, B. Federated Learning-Based Intrusion Detection System for the Internet of Things Using Unsupervised and Supervised Deep Learning Models. *Cyber Secur. Appl.* **2025**, *3*, 100068. https://doi.org/10.1016/j.csa.2024.100068.
69. Khraisat, A.; Alazab, A.; Alazab, M. et al. Federated Learning for Intrusion Detection in IoT Environments: A Privacy-Preserving Strategy. *Discover Internet Things* **2025**, *5*, 72. https://doi.org/10.1007/s43926-025-00169-7.
70. Al Tfaily, F.; Ghalmane, Z.; Brahmia, M.E.A. et al. Graph-Based Federated Learning Approach for Intrusion Detection in IoT Networks. *Sci. Rep.* **2025**, *15*, 41264. https://doi.org/10.1038/s41598-025-25175-1.
71. Hosain, Y.; Çakmak, M. XAI-XGBoost: An Innovative Explainable Intrusion Detection Approach for Securing Internet of Medical Things Systems. *Sci. Rep.* **2025**, *15*, 22278. https://doi.org/10.1038/s41598-025-07790-0.
72. Chen, Y.; Zheng, X.; Wang, N. Construction of VAE-GRU-XGBoost Intrusion Detection Model for Network Security. *PLOS ONE* **2025**, *20*, e0326205. https://doi.org/10.1371/journal.pone.0326205.
73. Adewole, K.S.; Jacobsson, A.; Davidsson, P. Intrusion Detection Framework for Internet of Things with Rule Induction for Model Explanation. *Sensors* **2025**, *25*, 1845. https://doi.org/10.3390/s25061845.
74. Shanmugam, V.; Razavi-Far, R.; Hallaji, E. Addressing Class Imbalance in Intrusion Detection: A Comprehensive Evaluation of Machine Learning Approaches. *Electronics* **2025**, *14*, 69. https://doi.org/10.3390/electronics14010069.
75. Yu, F.; Liu, T.; Wang, Z.; An, H. An Attack Detection Method Based on Deep Learning for Internet of Things. *Sci. Rep.* **2025**, *15*, 27685. https://doi.org/10.1038/s41598-025-14808-0.