BILINGUAL PUBLISHING GROUP

Journal of Electronic & Information Systems
https://journals.bilpubgroup.com/index.php/jeis

ARTICLE

Comparative Evaluation of Ensemble and Tree-Based Machine Learning Algorithms for Network Intrusion Detection

Atakan Özçelebi , Vedat Marttin* [*]

Department of Computer Engineering, Faculty of Engineering, Bilecik Seyh Edebali University, Bilecik 11210, Türkiye

ABSTRACT

The increasing sophistication and scale of malicious network activities demand a fundamental shift from traditional signature-based intrusion detection systems toward adaptive, data-driven security architectures. Machine learning (ML) provides an effective paradigm for addressing this challenge by identifying intricate and non-linear patterns associated with cyber threats within complex, high-dimensional network data. This study presents a comprehensive comparative analysis of four widely used ensemble and tree-based ML algorithms Random Forest (RF), Decision Tree (DT), XGBoost, and LightGBM applied to the multi-class classification of contemporary network intrusions. Using the benchmark CIC-IDS-2017 dataset, a meticulous preprocessing framework was implemented to ensure data integrity, reproducibility, and methodological rigor. Model performance was evaluated through standard classification metrics, with macro-averaged F1-score prioritized to provide an equitable assessment across highly imbalanced class distributions. Experimental findings reveal substantial differences in performance among the examined algorithms. Although RF, DT, and LightGBM achieved overall accuracy levels exceeding 99.8%, XGBoost consistently demonstrated superior capability in identifying minority attack categories, achieving the highest overall accuracy of 99.89% and a macro-averaged F1-score of 0.8903. These results highlight XGBoost's enhanced generalization capacity and resilience to class imbalance, confirming its suitability for deployment in real-time cybersecurity environments. In conclusion, this research establishes a consistent methodological benchmark for evaluating ensemble-based intrusion detection algorithms. It underscores the critical importance of balanced model assessment in the context of skewed network traffic distributions. The findings suggest that XGBoost offers the

*CORRESPONDING AUTHOR:
Vedat Marttin, Department of Computer Engineering, Faculty of Engineering, Bilecik Seyh Edebali University, Bilecik 11210, Türkiye; Email: vedat.marttin@bilecik.edu.tr

ARTICLE INFO
Received: 2 August 2025 | Revised: 22 September 2025 | Accepted: 29 September 2025 | Published Online: 7 October 2025
DOI: https://doi.org/10.30564/jeis.v7i2.12299

CITATION
Özçelebi, A., Marttin, V., 2025. Comparative Evaluation of Ensemble and Tree-Based Machine Learning Algorithms for Network Intrusion Detection. Journal of Electronic & Information Systems. 7(2): 116–131. DOI: https://doi.org/10.30564/jeis.v7i2.12299

COPYRIGHT
Copyright © 2025 by the author(s). Published by Bilingual Publishing Group. This is an open access article under the Creative Commons Attribu tion-NonCommercial 4.0 International (CC BY-NC 4.0) License (https://creativecommons.org/licenses/by-nc/4.0/).

most reliable and balanced performance profile for practical implementation within modern Security Operations Centers (SOCs), providing a strong foundation for adaptive and intelligent intrusion detection frameworks.

Keywords: Cyber Security; Machine Learning; Intrusion Detection; Classification; Random Forest; CIC-IDS-2017; XGBoost; LightGBM

1. Introduction

The modern digital ecosystem is witnessing an unprecedented expansion in the number of interconnected devices and services, which, while fostering innovation and connectivity, has simultaneously widened the attack surface exploitable by malicious actors. Traditional network security frameworks—predominantly based on static rules and predefined signatures—are increasingly inadequate against polymorphic, zero-day, and advanced persistent threats (APTs) [1,2]. These conventional Intrusion Detection Systems (IDS) inherently lack the capacity to generalize from previously unseen patterns, rendering them ineffective in identifying novel or evolving attack vectors. Consequently, the cybersecurity landscape necessitates a paradigm shift toward proactive, adaptive, and intelligence-driven defense mechanisms capable of detecting and mitigating previously unknown malicious behaviors in real time.

Machine learning (ML) has emerged as a foundational technology in contemporary cybersecurity, offering data-driven methods capable of discerning complex, non-linear correlations in high-dimensional network traffic data. By autonomously learning from historical and real-time data streams, ML-based models can detect subtle deviations in network behavior indicative of intrusion attempts [3,4]. When integrated into Network Intrusion Detection Systems (NIDS), these approaches transcend the limitations of rule-based architectures, enabling adaptive and resilient threat detection strategies that evolve alongside the threat landscape.

This study focuses specifically on supervised learning methods, with an emphasis on ensemble and tree-based algorithms namely, Random Forest, Decision Tree, XGBoost, and LightGBM. The choice of these algorithms is substantiated by consistent findings in the literature that demonstrate their superior performance, scalability, and interpretability in intrusion detection tasks [5–7]. Ensemble-based algorithms combine the predictive strengths of multiple weak learners to enhance generalization and mitigate overfitting, while tree-based structures facilitate transparent decision-making processes. This interpretability is critical in cybersecurity contexts, where explainability supports forensic analysis, compliance auditing, and incident response [8]. Compared to deep learning models, which often operate as "black boxes" [9], tree-based methods offer a more interpretable and computationally efficient alternative—particularly valuable when processing large-scale tabular datasets such as network flow records generated by CICFlowMeter and similar tools [10,11].

Despite significant progress, the current literature still lacks a unified and methodologically consistent comparison of leading ensemble and tree-based algorithms for intrusion detection. Many prior studies rely on heterogeneous preprocessing pipelines, inconsistent validation strategies, or singular evaluation metrics such as overall accuracy, which can produce misleading impressions of performance—especially under conditions of severe class imbalance [12]. The inherent imbalance in network traffic data, where malicious instances represent only a minute fraction of total observations, exacerbates this issue by inflating performance scores for majority classes while obscuring deficiencies in minority-class detection.

Addressing this critical gap, the present study provides a rigorous, reproducible benchmark analysis of four prominent algorithms within a unified experimental framework. The CIC-IDS-2017 dataset, widely recognized for its comprehensive representation of modern attack categories, is employed to ensure comparability and replicability. A meticulous preprocessing pipeline encompassing feature normalization, encoding, and data partitioning is implemented to maintain methodological integrity across all models.

Performance evaluation extends beyond overall accuracy to incorporate macro-averaged precision, recall, and F1-scores—metrics that treat each class with equal importance regardless of frequency. This evaluation paradigm ensures an unbiased representation of model behavior, yielding insights that are both statistically robust and operationally relevant. By prioritizing class-level performance balance, the study establishes a more realistic understanding of how machine learning models operate in practical intrusion detection settings, where the ability to detect rare but severe threats is paramount.

The highlights of the study are as follows:

- Using Macro F1-Score: Evaluating unbalanced datasets using macro-averaged F1 instead of classical accuracy offers a strong contribution compared to the literature.
- Methodological Consistency: All algorithms were tested with the same preprocessing pipeline and the same hyperparameter settings, providing a fair comparison.
- Balanced Performance of XGBoost: The superiority of XGBoost in rare attack classes is demonstrated in detail.
- Open Benchmark Contribution: The study provides a methodological reference that can be used in security operations centers.
- Detailed Results Presentation: A transparent analysis was conducted using a confusion matrix, macro metrics, and comparison tables.

The findings presented herein aim to serve both the academic and professional cybersecurity communities. For researchers, the work contributes a standardized methodological baseline against which future studies can be benchmarked. For practitioners, it provides empirical guidance for selecting algorithms that offer the optimal trade-off between interpretability, computational efficiency, and detection performance—qualities essential for deployment in real-world Security Operations Centers (SOCs).

The remainder of this paper is structured as follows: Section Literature Review provides an expanded review of related literature, establishing the theoretical and empirical context for the present research. Section 2 describes the dataset, detailing the preprocessing techniques, model training procedures, and evaluation methodology. Section 3 presents the experimental findings, including detailed performance metrics and confusion matrices for each algorithm. Finally, Section 4 synthesizes the results, discusses their implications for real-world cybersecurity operations, outlines key contributions, and concludes with recommendations for future research directions.

Literature Review

A comprehensive review of the literature reveals several dominant research dimensions shaping the evolution of machine learning (ML)-based intrusion detection systems (IDS). One of the fundamental prerequisites for empirical research in this domain is the availability of reliable and representative datasets for both model training and performance evaluation. Among the most widely recognized benchmarks are the CIC-IDS-2017 [13] and UNSW-NB15 [14] datasets, each offering a diverse and contemporary range of attack scenarios that reflect real-world network conditions. These datasets have become cornerstones of academic experimentation, enabling researchers to systematically evaluate various ML techniques under standardized conditions and to compare their relative efficacy in detecting complex intrusions.

Early studies in this field primarily investigated conventional ML algorithms such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) [15]. While these algorithms achieved satisfactory results in smaller-scale experiments, their effectiveness diminished when confronted with the high dimensionality and volume characteristic of modern network traffic. The computational overhead required for feature extraction, training, and inference often rendered these methods impractical for real-time intrusion detection in large-scale environments [5].

In parallel, the emergence of deep learning (DL) methods, particularly those employing Convolutional Neural Networks (CNNs) and other neural architectures [16–18], introduced a new paradigm by enabling automatic hierarchical feature learning directly from raw input data. These models demonstrated strong capability in capturing spatial and temporal dependencies within network traffic flows, significantly advancing the state of anomaly detection and threat classification [1]. However, despite their promising accuracy, deep neural models exhibit two major limitations that constrain their practical deployment in cybersecurity operations. First, their inherent "black-box" [9] nature hinders interpretability, making it difficult for analysts to trace the rationale behind classification decisions—an essential requirement in forensic analysis and incident response. Second, the substantial computational resources required for both training and inference stages pose scalability challenges, particularly in environments with limited hardware or energy constraints.

Recent scholarly discourse indicates a pronounced shift toward ensemble and tree-based algorithms, which consistently deliver a favorable balance between performance, interpretability, and computational efficiency. These models, characterized by structured decision hierarchies and collective learning mechanisms, have emerged as reliable solutions for contemporary IDS applications. Decision tree algorithms, in their simplest form, are valued for their clarity and transparent rule-based structure, which allows practitioners to readily understand and validate the decision-making process[19]. However, individual decision trees tend to overfit the training data, thereby limiting their generalization capability. To overcome this limitation, ensemble techniques such as bagging and boosting have gained prominence as de facto standards in modern ML-based intrusion detection research.

Among ensemble approaches, Random Forest (RF) has been widely recognized for its robustness and consistent performance across heterogeneous datasets. By constructing multiple decorrelated trees through bootstrap aggregation, RF effectively mitigates overfitting while maintaining high predictive accuracy. Numerous studies have reported that RF achieves superior results on benchmark datasets like CIC-IDS-2017, making it one of the most dependable algorithms in academic and applied contexts[6].

Beyond RF, gradient boosting frameworks such as XGBoost have redefined performance benchmarks in this domain. XGBoost introduces regularization mechanisms and optimized gradient computations that enhance both learning efficiency and model generalization. Its ability to handle high-dimensional, imbalanced, and noisy data has positioned it as a dominant method, frequently outperforming traditional ensemble models in both speed and predictive accuracy[7]. The remarkable success of XGBoost and its successors (e.g., LightGBM and CatBoost) has further consolidated ensemble and tree-based algorithms as the methodological cornerstone for intrusion detection research.

Moreover, these algorithms have proven highly adaptable to specialized application contexts. For instance, researchers have successfully applied ensemble-based IDS models to complex environments such as the Internet of Things (IoT)[10], where lightweight yet accurate detection mechanisms are required, and to specific threat domains such as botnet detection[15]. Their consistent reliability across diverse network infrastructures underscores their suitability for real-world deployment.

Collectively, the reviewed literature underscores a clear methodological trajectory: from early reliance on traditional ML algorithms to the growing dominance of ensemble and tree-based methods. This progression reflects the field's increasing emphasis on achieving not only high predictive accuracy but also operational interpretability, scalability, and efficiency. **Table 1** provides a summary of key academic studies forming the empirical and methodological foundation of the present research. It highlights the evolution of algorithmic preferences and methodological rigor within ML-based intrusion detection, illustrating how ensemble and tree-based models have come to represent the most practical and effective paradigm for modern cybersecurity defense.

Table 1. Summary of Foundational Academic Studies.

Author (Year)	Topic	Methodology	Key Finding	Relevance to Project
Ahmad et al. (2018)[5]	Algorithm Comparison	RF, K-NN, SVM on NSL-KDD	RF achieved the highest accuracy among the tested models.	Justifies the inclusion of RF as a strong candidate.
Hindy et al. (2020)[6]	Comprehensive Benchmark	RF, SVM, DT on CIC-IDS-2017	RF exhibited balanced and high-performance results.	Supports the primary hypothesis on dataset.
Mahbooba et al. (2021)[8]	Explainable AI (XAI)	RF + SHAP/LIME	Enables interpretation of model decisions.	Emphasizes importance of future XAI work.
Sharafaldin et al. (2018)[13]	Benchmark Dataset	Network traffic analysis	Introduced the CIC-IDS-2017 dataset.	Provides the data foundation for the project.

2. Materials and Methods

The methodology adopted in this study was developed to ensure systematic execution, transparency, and full reproducibility of results. This section provides a detailed overview of the employed dataset, the architectural framework of the experimental system, and the multi-phase data preprocessing pipeline implemented to ensure data integrity and consistency. Additionally, it outlines the theoretical principles and operational mechanisms underlying the selected classification algorithms, establishing the methodological foundation for the subsequent experimental analysis. Together, these components form a coherent framework that guarantees methodological rigor and facilitates reproducible research in machine learning-based intrusion detection.

2.1. Dataset

This study employs the CIC-IDS-2017 dataset[7,13], a widely acknowledged benchmark[20] for evaluating machine learning algorithms in network intrusion detection research. The dataset captures both benign (normal) network traffic and a diverse range of attack types that accurately represent the complexity of the contemporary cyber threat landscape. It was generated under realistic network conditions and includes features extracted from packet-level data using tools such as CICFlowMeter, thereby ensuring high-quality and reproducible data for experimental analysis.

Following an extensive preprocessing procedure and the exclusion of statistically insignificant minority classes, the final dataset used for model training and evaluation comprised the class distribution presented in **Table 2**. The substantial imbalance between benign and malicious traffic categories particularly between the majority benign class and minority attack types highlights the necessity of employing evaluation metrics that remain robust under skewed class distributions. Accordingly, this study emphasizes the use of macro-averaged performance indicators such as the F1-score, which assign equal importance to each class irrespective of its frequency.

Table 2. Distribution of Classes in the Preprocessed Dataset.

Class Label (Attack Type)	Number of Samples
BENIGN	2,273,097
DoS Hulk	231,073
PortScan	158,930
DDoS	128,027
DoS GoldenEye	10,293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack – Brute Force	1507
Web Attack – XSS	652
Infiltration	36
Web Attack – Sql Injection	21
Total	2,830,732

2.1.1. Attack Categories and Descriptions

A comprehensive understanding of the dataset and its experimental implications to facilitate, the major attack categories and their corresponding characteristics are summarized below. Each category reflects distinct adversarial tactics and network behaviors used to evaluate model generalization and sensitivity to varied intrusion types.

BENIGN (Normal Traffic): This category represents legitimate, non-malicious network activity that exhibits no indicators of compromise. Benign flows serve as the baseline reference for distinguishing normal system behavior from anomalous or hostile patterns. Accurate classification of benign traffic is essential to minimize false-positive alerts in operational intrusion detection systems.

DoS/DDoS (Denial of Service Attacks): Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks aim to exhaust the computational or bandwidth resources of a target system or service by flooding it with superfluous traffic. The objective is to render legitimate services inaccessible to authorized users. While DoS attacks originate from a single source, DDoS attacks involve multiple compromised devices (often part of a botnet) acting in concert to amplify the attack's scale and impact. Common examples include Hulk, GoldenEye, Slowloris, and Slowhttptest—each employing unique techniques to overwhelm target systems.

Port Scanning: Port scanning is a reconnaissance technique used by attackers to identify open ports and vulnerable services within a network. Analogous to checking entry points in a building, this activity enables attackers to map the system's accessible interfaces before executing a more targeted intrusion. While not inherently destructive, port scanning often constitutes the initial phase of a coordinated cyberattack.

Botnet (Bot): A botnet refers to a network of compromised computers—often described as a "zombie army"—that are remotely controlled by malicious actors. Once infected, these systems can be leveraged to execute large-scale operations such as DDoS attacks, phishing campaigns, or malware dissemination. Within the CIC-IDS-2017 dataset, botnet traffic serves as a representative example of coordinated, automated attack behavior that is particularly challenging to detect due to its distributed and stealthy nature.

Infiltration: Infiltration attacks involve unauthorized access to a network or system through compromised hosts or backdoors. Once the attacker successfully penetrates the network perimeter, the focus shifts to internal reconnaissance, data exfiltration, and lateral movement within the compromised environment. In the dataset, infiltration traffic captures the distinct signatures of early-stage network compromise, allowing models to assess their ability to detect subtle intrusion attempts.

Web Attacks: Web-based attacks target web servers, applications, and underlying databases, exploiting vulnerabilities in input validation, session management, or authentication mechanisms. These attacks often aim to compromise user data, alter web content, or gain unauthorized administrative access. Three common subtypes are included in the CIC-IDS-2017 dataset:

Brute Force Attacks: These attacks attempt to gain unauthorized access by systematically trying numerous username-password combinations until the correct credentials are found. Such methods exploit weak or default authentication mechanisms and are often automated.

Cross-Site Scripting (XSS): XSS attacks inject malicious scripts into otherwise trusted web applications. When executed in the browser of unsuspecting users, these scripts can hijack sessions, steal cookies, or redirect users to malicious websites. XSS attacks are particularly dangerous due to their ability to exploit client-side vulnerabilities without requiring direct access to the server.

SQL Injection (SQLi): SQL Injection attacks manipulate database queries through unsanitized user inputs. By injecting malicious SQL commands into web forms or URLs, attackers can gain unauthorized access to sensitive information such as credentials or financial data. This attack type exemplifies the risks of inadequate input validation and poor database security practices.

Patator (FTP and SSH Brute-Force Attacks): The FTP-Patator and SSH-Patator attacks represent specialized brute-force strategies directed at the File Transfer Protocol (FTP) and Secure Shell (SSH) services, respectively. These attacks continuously attempt multiple credential combinations to gain unauthorized access to remote systems. Their persistence and automation make them particularly effective against improperly configured or weakly secured authentication systems.

Collectively, these diverse attack types provide a comprehensive foundation for evaluating the robustness and adaptability of machine learning models in intrusion detection. The inclusion of both volumetric (e.g., DDoS) and stealth-oriented (e.g., Infiltration, XSS) attacks ensures that models are tested across the full spectrum of cyber threat scenarios encountered in modern networks.

The pronounced class imbalance in the CIC-IDS-2017 dataset mirrors real-world network environments, where malicious traffic constitutes only a small fraction of total network activity. This characteristic presents a significant challenge for model development, as traditional accuracy metrics can obscure poor performance on minority attack classes. Consequently, the use of macro-averaged evaluation metrics particularly the F1-score is critical for achieving a balanced and realistic assessment of model performance.

By integrating these detailed attack categories and class distributions, this study establishes a rigorous empirical foundation for assessing the generalization capability of ensemble and tree-based algorithms in detecting diverse and complex intrusion behaviors within large-scale network datasets.

2.1.2. Data Splitting and Balancing Techniques

To ensure a robust and unbiased evaluation, the pre-processed dataset was first split into two subsets: 80% for training and 20% for testing. The test set was kept separate and used only for the final performance evaluation.

Given the significant class imbalance shown in **Table 2**, which can bias machine learning models towards the majority class ("BENIGN"), various balancing techniques were applied exclusively to the 80% training data. The test data remained in its original, imbalanced state to reflect a realistic deployment scenario. The following balancing techniques were employed on the training set:

Original (Unbalanced): The original 80% training set was used as a baseline.

SMOTE (Synthetic Minority Over-sampling TEchnique): This oversampling method creates synthetic samples for the minority classes.

RandomOverSampler: This method randomly duplicates samples from the minority classes.

RandomUnderSampler: This method randomly removes samples from the majority class.

Table 3 presents the distribution of the training dataset after these techniques were applied.

Table 3. Distribution of Training Data after Balancing Techniques.

Balancing Technique	Data Distribution (Samples, Features)	Class Distribution (Class 0, Class 1...)
Original (Unbalanced)	(2,264,585, 77)	BENIGN (1,818,477), Bot (1573), DDoS (102,421), DoS GoldenEye (8234), DoS Hulk(184,858), DoS Slowhttptest (4399), DoS slowloris (4637), FTP-Patator (6350), Infiltration (29), PortScan (127,144), SSH-Patator (4718), Web Attack- Brute Force (1206), Web Attack- Sql Injection (17), Web Attack- XSS (522)
SMOTE	-	-
RandomOverSampler	-	-
RandomUnderSampler		BENIGN (17), Bot (17), DDoS (17), DoS GoldenEye (17), DoS Hulk (17), DoS Slowhttptest (17), DoS slowloris (17), FTP-Patator (17), Infiltration (17), PortScan (17), SSH-Patator (17), Web Attack-Brute Force (17), Web Attack-Sql Injection (17), Web Attack-XSS (17)

2.2. System Architecture and Methodology

The proposed system adopts a modular architecture, illustrated in **Figure 1**, specifically designed to efficiently process and analyze large-scale, complex network datasets. The first core component, the Training Module, performs all computationally intensive tasks including data preprocessing, model training, and hyperparameter optimization—only once. Upon completion, the trained models, data scalers, and associated configuration files are permanently stored on disk, ensuring their availability for subsequent analyses without re-execution of the entire workflow.

The second component, the Analysis and Prediction Module, leverages these pre-trained and stored models to perform rapid performance evaluations and generate detailed analytical reports within seconds. This approach substantially reduces computational overhead and eliminates the need for repeated, time-consuming training cycles.

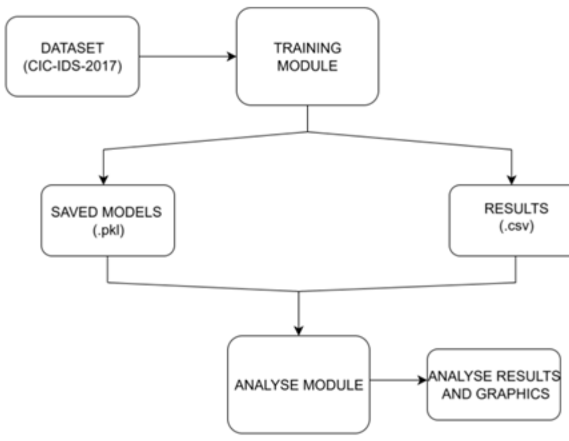

Figure 1. General architecture of the developed system, separating the training and analysis modules.

To ensure the accuracy and robustness of the developed models, the raw dataset was subjected to a comprehensive, multi-phase preprocessing pipeline, as illustrated in **Figure 2**.

This pipeline encompassed data cleaning, normalization, encoding, and partitioning procedures designed to preserve data integrity and consistency across experimental iterations. Collectively, this modular design enables efficient experimentation, reproducibility, and scalability-key requirements for practical deployment in real-world cybersecurity and machine learning research environments.

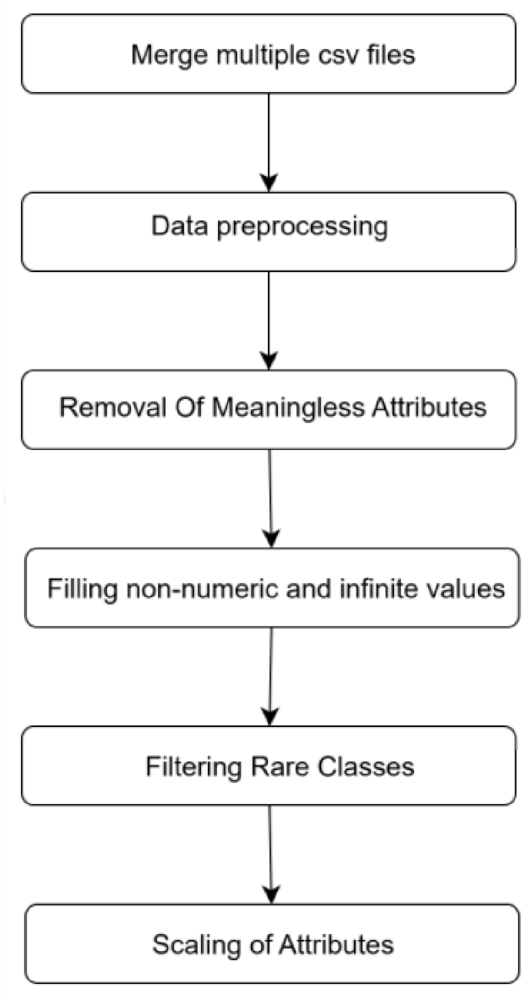

Figure 2. The sequential stages of the data preprocessing pipeline.

Since the raw dataset was distributed across multiple CSV files, all files were initially merged into a unified data frame to enable holistic analysis. To enhance model performance and reliability, a comprehensive preprocessing pipeline was applied, as summarized in **Figure 2**. This pipeline included several key steps: removing irrelevant or redundant features, imputing missing and infinite values using the SimpleImputer technique, filtering out statistically insignificant minority classes, and standardizing all numerical features with StandardScaler. These procedures ensured data consistency, reduced noise, and improved the overall quality of inputs for subsequent model training and evaluation.

2.3. Classification Algorithms

Four prominent tree- and ensemble-based models were selected for comparative analysis. The theoretical foundations of these algorithms are well-established in statistical learning literature[21,22].

Decision Tree (DT): DT algorithm is widely employed for both regression and classification problems[23]. It constructs a hierarchical, tree-structured model by recursively dividing the dataset according to feature values, thereby allowing the model to make decisions at each branching point. A notable advantage of DTs lies in their transparency, as they emulate human-like reasoning processes and can be easily represented through flowchart-style diagrams.

A decision tree is composed of three main elements: the root node, which represents the full dataset and initiates the partitioning procedure; internal nodes, where decision rules are applied based on chosen attributes; and leaf nodes, which provide the final class assignment or predicted output.

The effectiveness of a split at any node is evaluated using impurity measures. Entropy is mathematically expressed in Equation (1)[24]:

$$H(S) = -\sum \rho_i \, log2\left(\rho_i\right) \tag{1}$$

where $H(S)$ is the entropy of the dataset S, and ρ_i is the proportion of samples belonging to class i.

Random Forest (RF): RF algorithm generates an ensemble of multiple decision trees (DTs) and integrates their predictions to enhance overall predictive accuracy[25]. It is grounded in the principle of bootstrap aggregation (bagging), whereby each tree is trained on a distinct subset of the data obtained through sampling with replacement. By aggregating the outputs of numerous trees, RF mitigates overfitting and achieves greater robustness compared to an individual DT.

$$y_{pred} = \text{majority\_vote}\left\{h_b(x)\right\}_{b=1}^{B} \tag{2}$$

In Equation (2), y_{pred} is the final predicted class, $h_b(x)$ is the prediction of the b-th tree, and B is the total number of trees in the forest.

Beyond employing bootstrap samples, RF introduces additional randomness by selecting a random subset of features at each node split, rather than evaluating the entire feature set. This dual-source randomness—arising from both data sampling and feature selection effectively lowers model variance and enhances generalization performance on unseen data.

Due to its stability, ease of implementation, and high predictive capability, RF is extensively applied in both classification and regression tasks, particularly in scenarios involving high-dimensional or noisy datasets.

XGBoost (Extreme Gradient Boosting): XGBoost represents an enhanced implementation of the gradient boosting framework, designed with a strong emphasis on computational efficiency, scalability, and regularization[26].

It extends conventional gradient boosting by incorporating mechanisms such as L1/L2 regularization, parallelized training, and effective strategies for managing missing values. These advancements establish XGBoost as one of the most powerful algorithms for structured data analysis in modern machine learning.

An optimized implementation of gradient boosting. It builds trees sequentially, where each new tree is trained to correct the errors of its predecessors. It minimizes a regularized objective function $L(\phi)$ in Equation (3):

$$L(\phi) = \sum\nolimits_{i=1}^{n} l\left(y_i, y_{pred}\right) + \sum\nolimits_{k=1}^{K} \Omega\left(f_k\right) \tag{3}$$

where l is the loss function measuring the difference between the actual value y_i and the predicted value y_{pred} and Ω is a regularization term that penalizes the complexity of the model'ss trees f_k.

LightGBM: LightGBM is a gradient boosting framework developed by Microsoft, designed with a focus on efficiency, scalability, and computational performance[27]. It is particularly advantageous when working with large-scale datasets and high-dimensional feature spaces.

Compared with conventional gradient boosting methods, LightGBM achieves superior training speed, reduced

memory consumption, and higher predictive accuracy through a set of algorithmic innovations.

Like other boosting-based algorithms, LightGBM optimizes a regularized objective function that combines a loss term with a complexity penalty in Equation (4):

$$\mathcal{L}(\phi) = \sum\nolimits_{i=1}^{n} l(y_i, \widehat{y}_i) + \sum\nolimits_{k=1}^{K} \Omega\left(f_k\right) \tag{4}$$

In this formulation, $l(y_i, \hat{y}_i)$ represents a differentiable loss function (e.g., logistic loss or squared error), $\hat{y}_i$ is the predicted value for the *i*-th sample, f_k corresponds to the tree generated at the *k*-th iteration, and $\Omega(f_k)$ denotes a regularization term that penalizes overly complex trees.

A key distinction between LightGBM and earlier boosting approaches lies in its tree construction strategy. Unlike level-wise growth, which expands all nodes at the same depth simultaneously, LightGBM utilizes a leaf-wise growth approach.

This strategy expands the leaf with the greatest capacity to minimize loss, resulting in deeper, more specialized trees that typically achieve lower training error and improved efficiency.

2.4. Technical Infrastructure and Hyperparameters

All experiments were conducted in a Python 3 environment, leveraging a suite of open-source libraries for data manipulation, machine learning, and visualization. The key software components are detailed in **Table 4**.

Table 4. Software and Libraries Used for Implementation.

Component	Purpose
Python 3	Core Programming Language
Pandas, NumPy	Data manipulation & numerical computation
Scikit-learn	ML framework for modeling & evaluation
XGBoost, LightGBM	Libraries for boosting algorithms
Matplotlib, Seaborn	Data visualization

The experiments were conducted on a system with the hardware and software specifications detailed in **Table 5**.

Table 5. Experimental Environment Specifications.

Component	Specification
CPU	Intel(R) Core(TM) i7-11800H 2.30 GHz, 2304 Mhz, 8 Core
GPU	NVIDIA GeForce RTX 3050 Laptop GPU
RAM	16.0 GB
Operating System	Microsoft Windows 11 Home
Python Version	3.13

The machine learning models were trained using the specific hyperparameter configurations listed in **Table 6**. These parameters, derived from the Scikit-learn, XGBoost, and LightGBM libraries, were kept consistent across all experiments to ensure a fair and direct comparison. To ensure reproducibility, all models were initialized with random_state = 42. To assess the stability and consistency of the models, the experiments were replicated using a different random seed (random_state = 100). The results showed negligible deviation from the primary findings, confirming the robustness of the reported outcomes.

Table 6. Hyperparameter Configuration for Trained Models.

Model	Hyperparameters
Decision Tree	random_state = 42
Random Forest	n_estimators = 100, random_state = 42, n_jobs = −1
XGBoost	random_state = 42, use_label_encoder = False, eval_metric = ‘smlogloss’, n_jobs = −1
LightGBM	random_state = 42, n_jobs = −1

3. Results

The empirical performance of the four trained models was rigorously evaluated on the held-out 20% test set. The models were trained on the four different variations of the training data (Original, SMOTE, RandomOverSampler, RandomUnderSampler).

During the data balancing phase, both SMOTE and RandomOverSampler failed to generate a dataset large enough to match millions of samples, most of which were in the ‘sBENIGN’ class, resulting in a “Memory Error (None-N/A)” error on the hardware used.

The consolidated results, including performance metrics and computational cost (training time), are presented in **Table 7**. This comprehensive table allows for a direct comparison of how each model performs under different data balancing conditions.

Table 7. Consolidated Model Performance Metrics.

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1-Score (Macro)	Training Time (s)
XGBoost (Original)	0.9989	0.9180	0.8850	0.8903	183.24
XGBoost (RandomUnderSampler)	0.5901	0.2672	0.8421	0.3013	0.87
Decision Tree (Original)	0.9987	0.8685	0.8255	0.8326	126.16
Decision Tree (RandomUnderSampler)	0.5456	0.2498	0.8184	0.2841	0.16
Random Forest (Original)	0.9986	0.8599	0.8176	0.8323	154.62
Random Forest (RandomUnderSampler)	0.5862	0.2946	0.8070	0.3192	1.91
LightGBM (Original)	0.9542	0.4823	0.3891	0.4181	119.76
LightGBM (RandomUnderSampler)	0.6001	0.2999	0.8452	0.3237	5.39

While the top three models achieve high accuracy, the macro-averaged metrics reveal a clearer performance hierarchy. **Figure 3** provides a visual comparison of these key performance indicators. A detailed breakdown of the class-wise performance for each model is provided by the confusion matrices in **Figures 4–7**.

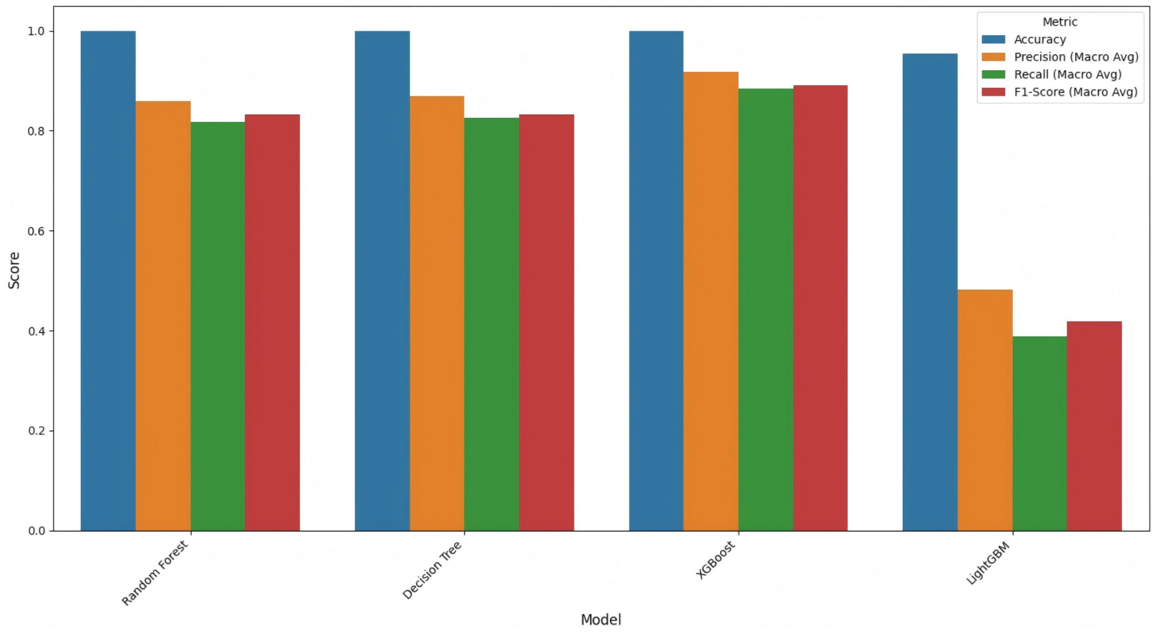

Figure 3. Detailed performance metrics across all models.

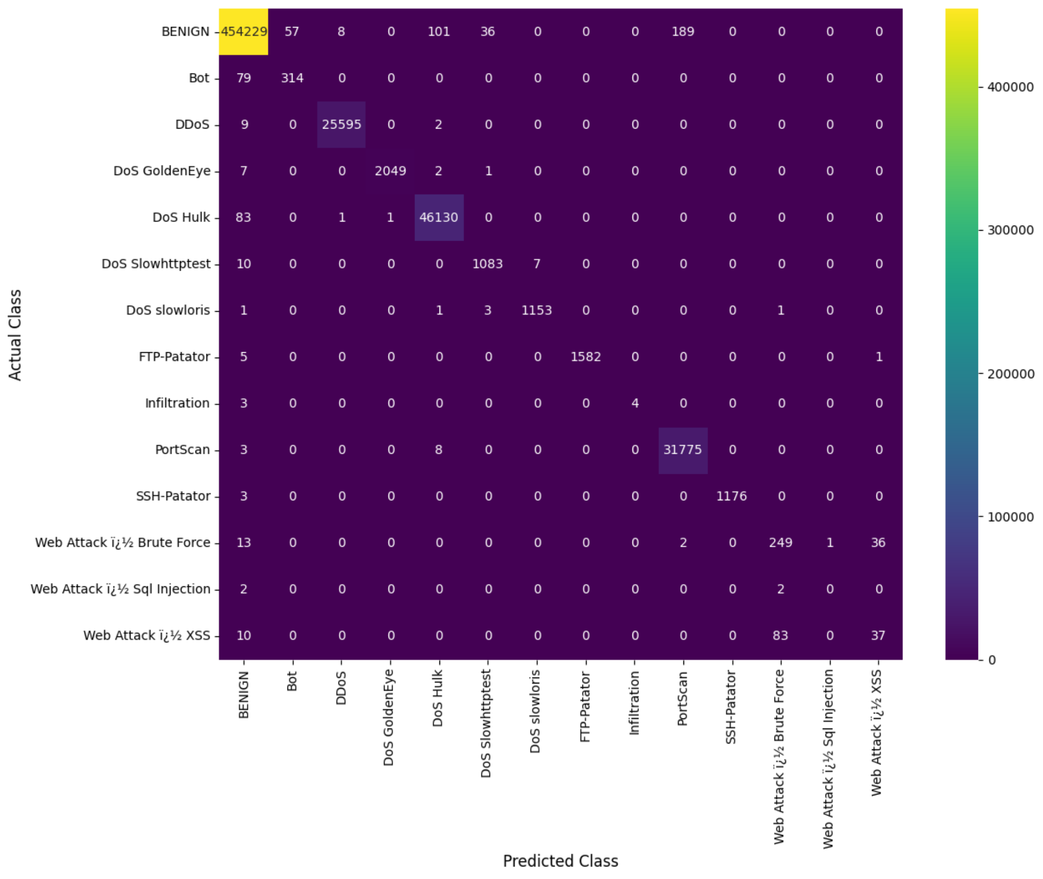

Figure 4. Confusion Matrix for the Random Forest Model.

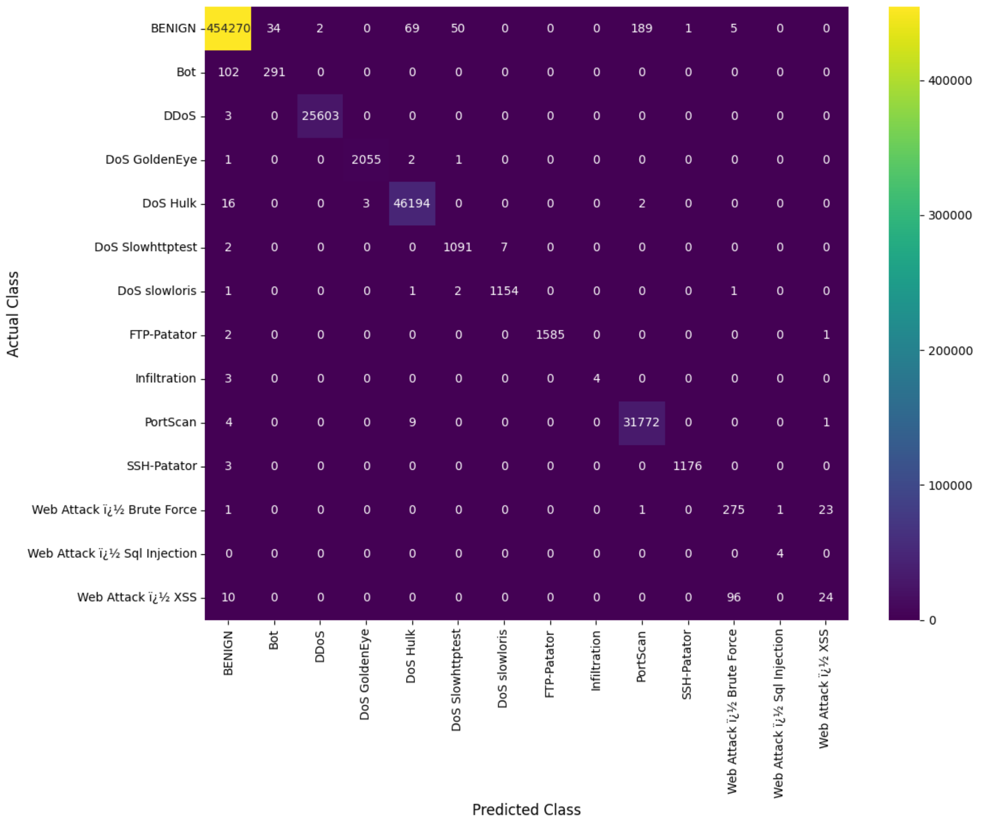

Figure 5. Confusion Matrix for the XGBoost Model.

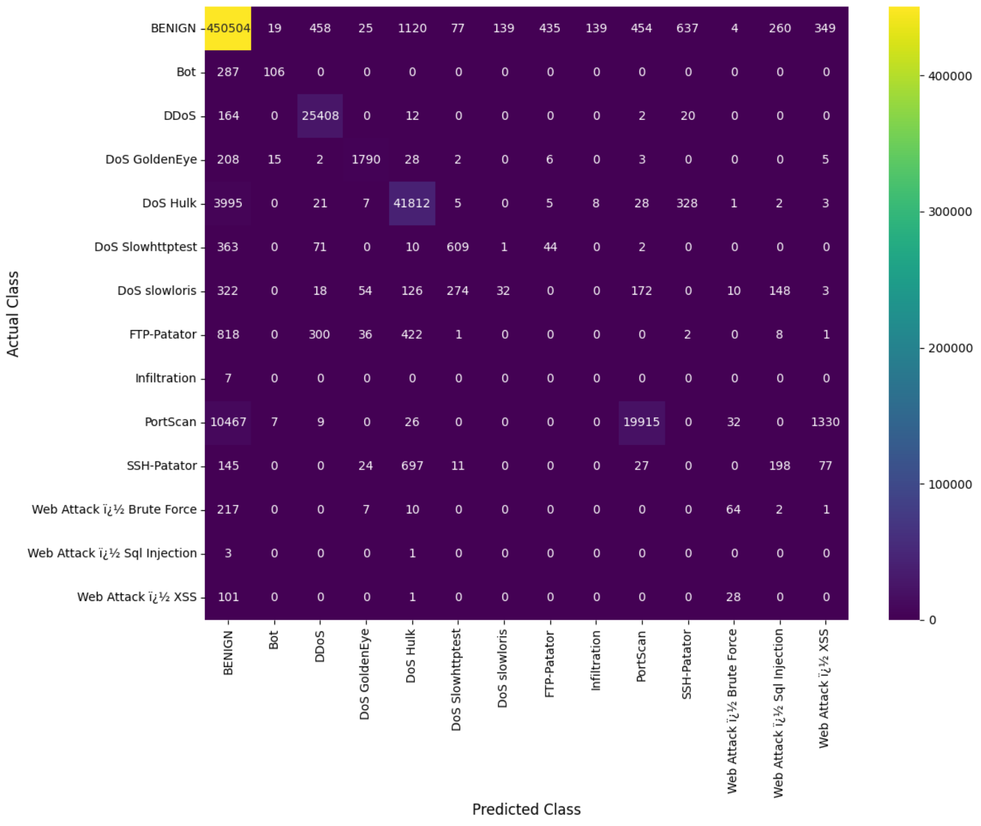

Figure 6. Confusion Matrix for the LightGBM Model.

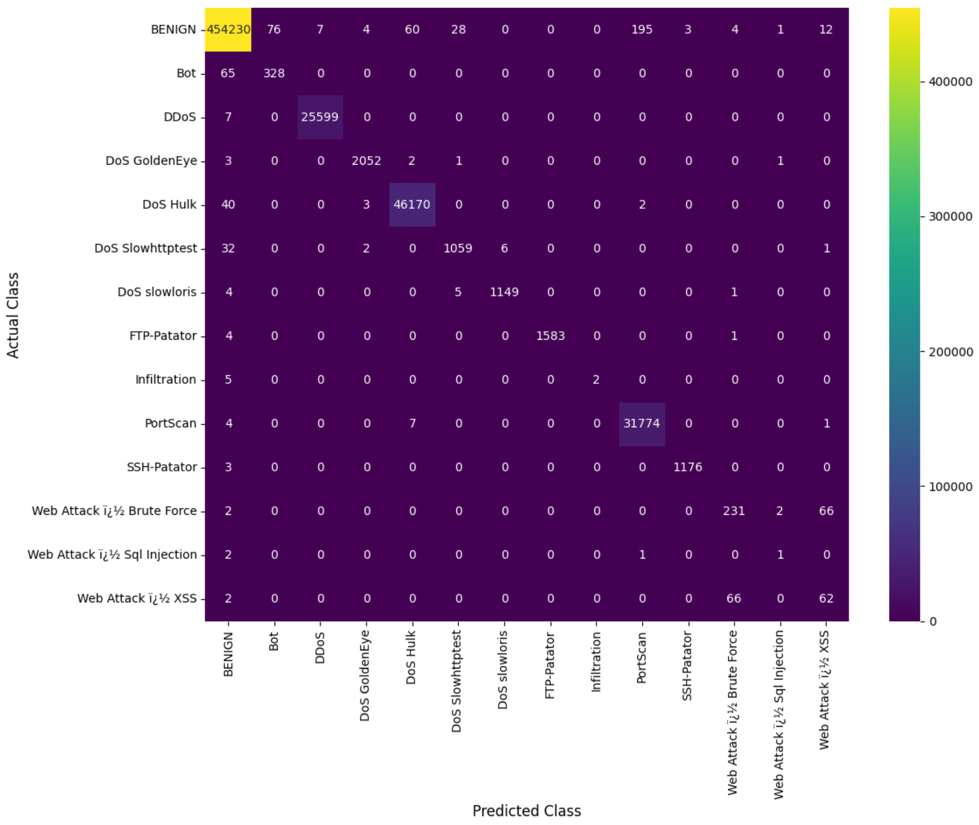

Figure 7. Confusion Matrix for the Decision Tree Model.

4. Discussion

The findings of this study provide several important insights into the practical effectiveness of ensemble and tree-based machine learning models in the context of network intrusion detection.

The superior performance of XGBoost can be attributed to its gradient boosting methodology, which sequentially trains learners to correct the errors of their predecessors, effectively focusing on difficult-to-classify instances like rare attack samples.

The notable underperformance of LightGBM with default parameters is an interesting finding, likely attributable to its leaf-wise growth strategy which may require specific tuning to prevent overfitting on this dataset.

As shown in **Table 7**, the application of data balancing techniques had a significant and varied impact on model performance. A critical finding was the computational infeasibility of oversampling methods for this large-scale dataset. Both SMOTE and RandomOverSampler failed to complete, resulting in a "Memory Error (N/A)" as they attempted to generate an impractically large dataset to match the millions of samples in the majority 'sBENIGN' class.

Conversely, RandomUnderSampler (RUS), while computationally viable, led to a drastic drop in performance for all models. For instance, the macro F1-score for Random Forest plummeted from 0.8323 (on the Original dataset) to 0.3192 (on the RUS dataset). This demonstrates that while RUS solves the computational problem, it does so by discarding the vast majority of 'sBENIGN' samples, leading to a massive loss of valuable information and a model that cannot effectively distinguish attacks.

The "Training Time" column in **Table 7** provides critical insights into the computational cost. As discussed, the oversampling methods were computationally infeasible. For the viable models, XGBoost, while being the most accurate (0.8903 Macro F1), also had the longest training time on the 'sOriginal' dataset (183.24 s).

In contrast, LightGBM was consistently the fastest model to train on the large, 'sOriginal' dataset (119.76 s), living up to its name, although its predictive performance was significantly lower (0.4181 Macro F1). The extremely fast training times on the RandomUnderSampler dataset (e.g., Decision Tree at 0.16 s) further illustrate this trade-off, as this speed was achieved on a much smaller, less informative dataset, rendering the models' predictions unreliable. This trade-off between accuracy and computational cost is a crucial consideration for real-world deployment.

5. Conclusions

This study presented a comprehensive comparative evaluation of four state-of-the-art machine learning (ML) algorithms for network intrusion detection, with a particular emphasis on ensemble and tree-based models. Using the imbalanced CIC-IDS-2017 dataset, the analysis employed macro-averaged performance metrics to ensure an equitable assessment across all attack categories. The results revealed that although multiple models achieved high overall accuracy, XGBoost consistently demonstrated the most balanced and reliable performance. Its superior capacity to accurately detect underrepresented attack classes, reflected in the highest macro-averaged F1-score, underscores its robustness and suitability for deployment in real-world cybersecurity systems, where the operational cost of false negatives is critically high.

This research establishes a transparent, data-driven benchmark that can assist both academics and practitioners in selecting and optimizing models for next-generation intrusion detection systems. The findings reinforce the importance of adopting evaluation frameworks that account for class imbalance—a pervasive challenge in network security datasets. By emphasizing macro-level metrics rather than aggregate accuracy, this study advances a more realistic understanding of model effectiveness in operational environments.

Furthermore, the results confirm the consistent superiority of ensemble-based methods in handling complex and heterogeneous data distributions. Among them, XGBoost's gradient boosting mechanism characterized by sequential error correction and adaptive weighting of weak learners emerges as the key factor behind its exceptional generalization and detection capability. This iterative refinement process enables XGBoost to focus on difficult-to-classify samples, thereby reducing misclassification rates and enhancing sensitivity to minority classes.

Beyond empirical validation, this study contributes methodologically to the field by demonstrating that the rigor of the experimental design and the appropriateness of evaluation criteria are as essential as algorithmic sophistication. A robust comparative framework ensures reproducibility, transparency, and the generation of actionable insights that extend beyond numerical performance metrics.

In conclusion, the research provides both an analytical and methodological reference for the development of reliable, scalable, and interpretable intrusion detection systems. The superior performance of XGBoost, validated under consistent experimental conditions, highlights its potential as a preferred algorithm for high-stakes cybersecurity operations. Ultimately, this work demonstrates that meaningful progress in network intrusion detection requires not only powerful models but also disciplined evaluation strategies that align with the complexities of real-world data.

Author Contributions

Conceptualization, V.M.; methodology, A.Ö.; software, A.Ö.; validation, V.M. and A.Ö.; formal analysis, A.Ö.; investigation, A.Ö. and V.M.; data curation, A.Ö.; writing—original draft preparation, A.Ö.; writing—review and editing, V.M.; visualization, A.Ö.; supervision, V.M. All authors have read and agreed to the published version of the manuscript.

Funding

These research received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

Canadian Institute for Cybersecurity. (2025). CICIDS2017 dataset. University of New Brunswick. Available online at: https://www.unb.ca/cic/datasets/ids-2017.html.

Conflicts of Interest

The authors declare no conflict of interest. During the preparation of this manuscript, the authors used an AI language model to assist with English translation, grammar correction, rephrasing of sentences for clarity, and formatting suggestions according to the journal'ss guidelines. The core concepts, experimental design, analysis, and interpretation of results are solely the work of the human authors.

References

[1] Thakkar, A., Lohiya, R., 2021. A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. Archives of Computational Methods in Engineering. 28, 3211–3243. DOI: https://doi.org/10.1007/s11831-020-09496-0

[2] Apruzzese, G., Colajanni, M., Ferretti, L., et al., 2018. On the effectiveness of machine and deep learning for cyber security. In Proceedings of the 10th International Conference on Cyber Conflict (CyCon), Tallinn, Estonia, 29 May–1 June 2018; pp. 371–390. DOI: https://doi.org/10.23919/CYCON.2018.8405026

[3] Genuario, F., Santoro, G., Giliberti, et al., 2024. Machine learning-based methodologies for cyber-attacks and network traffic monitoring: A review and insights. Information. 15(11), 741. DOI: https://doi.org/10.3390/info15110741

[4] Gu, J., Lu, S., 2021. An effective intrusion detection approach using SVM with naïve Bayes feature embedding. Computers & Security. 103, 102158. DOI: https://doi.org/10.1016/j.cose.2020.102158

[5] Ahmad, I., Basheri, M., Iqbal, M.J., et al., 2018. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. IEEE Access. 6, 33789–33795.

[6] Hindy, H., Brosset, D., Bayne, et al., 2020. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. IEEE Access. 8, 104650–104675. DOI: https://doi.org/10.1109/ACCESS.2020.3000179

[7] Canadian Institute for Cybersecurity, 2025. Intrusion detection evaluation dataset (CIC-IDS2017). Available from: https://www.unb.ca/cic/datasets/ids-2017.html (cited 13 July 2025).

[8] Mahbooba, B., Timilsina, M., Sahal, R., et al., 2021. Explainable artificial intelligence (XAI) to enhance trust management in intrusion detection systems using decision tree model. Complexity. 2021(1), 6634811. DOI: https://doi.org/10.1155/2021/6634811

[9] McCulloch, W.S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology. 5(1/2), 99–115.

[10] Ullah, I., Mahmoud, Q.H., 2022. Design and development of RNN anomaly detection model for IoT networks. IEEE Access. 10, 62722–62750. DOI: https://doi.org/10.1109/ACCESS.2022.3176317

[11] Panigrahi, R., Borah, S., 2018. A detailed analysis of CICIDS2017 dataset for designing network intrusion detection systems. International Journal of Engineering & Technology. 7(3), 479–482.

[12] Cieslak, D.A., Chawla, N.V., Striegel, A., 2006. Combating imbalance in network intrusion datasets. In Proceedings of the 2006 IEEE International Conference on Granular Computing, Atlanta, GA, USA, 10–12 May 2006.

[13] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. 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, Portugal, 22–24 January 2018; pp. 108–116. DOI: https://doi.org/10.5220/0006639801080116

[14] Moustafa, N., Slay, J., 2015. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6. DOI: https://doi.org/10.1109/MilCIS.2015.7348942

[15] Azeez, N.A., Odufuwa, O.E., Misra, S., et al., 2021. Windows PE malware detection using ensemble learning. Informatics. 8(1), 10. DOI: https://doi.org/10.3390/informatics8010010

[16] Azam, Z., Islam, M.M., Huda, M.N., 2023. Comparative analysis of intrusion detection systems and machine learning-based model analysis through decision tree. IEEE Access. 11, 80348–80391. DOI: https://doi.org/10.1109/ACCESS.2023.3296444

[17] James, G., Witten, D., Hastie, T., et al., 2013. An Introduction to Statistical Learning. Springer: New York, NY, USA. DOI: https://doi.org/10.1007/978-1-4614-7138-7

[18] Almuhanna, R., Dardouri, S., 2025. A deep learning/machine learning approach for anomaly-based network intrusion detection. Frontiers in Artificial Intelligence. 8, 1625891. DOI: https://doi.org/10.3389/frai.2025.1625891

[19] Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning. Springer: New York, NY, USA. DOI: https://doi.org/10.1007/978-0-387-84858-7

[20] Maseer, Z.K., Yusof, R., Bahaman, N., et al., 2021. Benchmarking of machine learning for anomaly-based intrusion detection systems in the CICIDS2017 dataset. IEEE Access. 9, 22351–22370. DOI: https://doi.org/10.1109/ACCESS.2021.3056614

[21] Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Springer: New York, NY, USA.

[22] Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press: Cambridge, MA, USA.

[23] Vinayakumar, R., Alazab, M., Soman, K.P., et al., 2019. Deep learning approach for intelligent intrusion detection system. IEEE Access. 7, 41525–41550. DOI: https://doi.org/10.1109/ACCESS.2019.2895334

[24] Aldweesh, A., Derhab, A., Emam, A.Z., 2020. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. Knowledge-Based Systems. 189, 105124. DOI: https://doi.org/10.1016/j.knosys.2019.105124

[25] Kim, J., Kim, J., Thu, H.L.T., et al., 2016. Long short term memory recurrent neural network classifier for intrusion detection. In Proceedings of the International Conference on Platform Technology and Service (PlatCon), Jeju, Republic of Korea, 15–17 February 2016; pp. 1–5.

[26] Buczak, A.L., Guven, E., 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials. 18(2), 1153–1176. DOI: https://doi.org/10.1109/COMST.2015.2494502

[27] Ke, G., Meng, Q., Finley, T., et al., 2017. LightGBM: A highly efficient gradient boosting decision tree. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 3149–3157.