XGBoost-Driven Intrusion Detection Method: Integrating SMOTE-Based Class Imbalance Mitigation and Multi-Phase Learning

Wanzhi Chen*, Diawara Faysal Almamy

Abstract - **The emergence of complex cyberattacks presents considerable difficulties to network security, necessitating innovative and adaptable solutions. Intrusion Detection Systems (IDS) play a vital role in cybersecurity by monitoring network traffic for malicious behavior and preventing potential threats. However, traditional IDS methods often suffer from high false-positive rates and struggle to adapt to evolving attack landscapes. This study explores the application of the XGBoost algorithm, an advanced gradient-boosting machine learning model for enhancing IDS performance using two widely recognized datasets: NSL-KDD and UNSW-NB15. A comprehensive data preprocessing pipeline was developed, including feature engineering, hyperparameter tuning, and synthetic data generation using the SMOTE technique to address class imbalance. The XGBoost model demonstrated superior performance, achieving 96.77% accuracy on the NSL-KDD dataset and 99.85% on the UNSW-NB15 dataset, with high precision, recall, and F1-scores. Cross-validation confirmed the model's ability to generalize effectively to unseen data, including novel attack types. The results of this study underscore XGBoost's potential as a scalable and robust solution for modern IDS, capable of handling high-dimensional data and complex attack scenarios. This research lays the groundwork for further integrating ensemble learning techniques with anomaly detection approaches to increase network security in dynamic contexts.**

Index Terms - **Class Imbalance Handling, Intrusion Detection, SMOTE, XGBoost.**

I. INTRODUCTION

NETWORK security has become a key issue in the contemporary digital environment, as the rapid increase in cyberattacks presents significant dangers to organizational infrastructure and personal information. Cyberattacks such as Distributed Denial of Service (DDoS), phishing, and advanced persistent threats (APTs) have increased in frequency and complexity, making it crucial to develop robust defense mechanisms [1]. Intrusion Detection Systems (IDS) represent one of the most effective mechanisms for detecting and mitigating security threats, as they continuously monitor network traffic to identify anomalies that may indicate malicious intent [2].

Manuscript received November 11, 2024; revised May 26, 2025

This research was supported by the General Project of scientific research funds of the Liaoning Provincial Department of Education (Grant No.2021LJKZ0327) and the GPU Resource Support Project of Liaoning Technical University (2024-02).

Wanzhi Chen is an associate professor of the School of Software at Liaoning Technical University, Huludao, 125105, China (Corresponding author, phone: +8613591996866, E-mail: chenwanzhi@lntu.edu.cn).

Diawara Faysal Almamy is a postgraduate student at Liaoning Technical University, Huludao, 125105, China
(E-mail: faysalalmamydiawara@gmail.com).

Traditional Intrusion Detection Systems (IDS) methods, such as signature-based and anomaly-based detection, often struggle with high false-positive rates and limited adaptability to the rapidly evolving nature of cyberattacks [3].

In recent years, machine learning techniques have proven to be powerful approaches for improving the accuracy, adaptability, and overall efficiency of IDS [4]. These algorithms can identify intricate network traffic data patterns, effectively detecting known and undiscovered (zero-day) threats [5].One method that has garnered considerable attention is Extreme Gradient Boosting (XGBoost). XGBoost, a sophisticated variant of gradient-boosted decision trees, is optimized for both speed and precision, making it particularly effective for analyzing high-dimensional and complex datasets like network traffic records [6] . Additionally, its ability to handle missing data, anomalies, and imbalanced datasets further enhances its effectiveness in IDS [2]. This research explores the application of XGBoost for intrusion detection using two prominent datasets: NSL-KDD and UNSW-NB15. The NSL-KDD dataset is an enhanced version of the original KDD Cup 1999 dataset, designed to address issues of duplication and complexity, thereby providing a more reliable baseline for evaluating IDS performance [7]. On the other hand, the UNSW-NB15 dataset, developed by the Australian Centre for Cyber Security, offers a modern and extensive array of attributes, covering a wide range of contemporary attack types and reflecting the current threat landscape [8].

This study enhances the XGBoost model through rigorous data preparation methods, feature engineering, and hyperparameter optimization. The results demonstrate the model's superior accuracy, precision, recall, and F1-Scores across both datasets, underscoring XGBoost's potential as a resilient and scalable solution for modern IDS deployments.

II. RELATED WORKS

Intrusion Detection Systems serve as essential elements of network security, developed to detect and respond to the growing spectrum of cyber threats targeting contemporary systems. However, conventional IDS approaches, such as signature-based detection, frequently fall short in identifying previously unseen or novel attack. These systems rely on predefined attack signatures, which results in challenges in detecting new threats. Signature-based approaches frequently suffer from heightened false-positive rates, as legitimate behaviors may occasionally mimic attack signatures [9]. This highlights the imperative for more adaptable, data-driven strategies to address known and unknown attacks more accurately and efficiently. In recent years, machine learning (ML) techniques have been extensively explored as

alternatives to conventional signature-based methods. These approaches do not necessitate established patterns and possess the capability to detect emerging attack patterns by learning directly from network traffic data. Ensemble learning methods have garnered substantial popularity due to their capacity to integrate the strengths of various base classifiers, thereby enhancing model accuracy and robustness. One notable approach is XGBoost, a gradient boosting framework recognized for its scalability and strong performance in handling complex and high-dimensional datasets [2]. XGBoost functions by sequentially training weak models to rectify the deficiencies of their predecessors, which enables it to uncover intricate relationships within the data and improve detection effectiveness, particularly in network intrusion detection. Despite the advantages of machine learning approaches, class imbalance remains a predominant challenge in IDS.

In standard network traffic datasets, benign samples typically dominate, resulting in a class imbalance that introduces a bias toward the majority class. This disparity often results in algorithms that excel at identifying benign traffic but inadequately detect infrequent yet significant attack occurrences. To address this issue, several methods, including the Synthetic Minority Over-sampling Technique (SMOTE), have been proposed. SMOTE addresses class imbalance by generating synthetic samples for the minority class through interpolation between existing examples. This process helps balance the dataset and improves classifier performance, particularly for rare attack types [10] . Combining SMOTE with XGBoost has proven effective in improving recall and precision for detecting infrequent attack occurrences while maintaining the accuracy of benign traffic. The continuously evolving nature of cyber threats underscores the need for multi-phase models that incorporate successive learning stages, thereby progressively improving the effectiveness of IDS. These models often encompass feature extraction, classification, and post-classification refining stages. Each phase improves the detection process: feature extraction converts raw data into more informative representations, while post-classification refining mitigates false positives and adjusts the model to emerging attack patterns. Integrating ensemble approaches like XGBoost with multi-phase learning frameworks enables IDS to adapt more effectively to cyberattacks' dynamic and diversified nature [5]. The amalgamation of these methodologies facilitates the development of more resilient, scalable, and adaptive Intrusion Detection System models capable of managing real-world network settings and continuously evolving attack strategies. Recent research has shown the effectiveness of integrating XGBoost with SMOTE for intrusion detection. Zhang et al. [11] demonstrated that this hybrid methodology significantly improved the efficacy of Intrusion Detection Systems, especially on benchmark datasets. These datasets, representing various attack types, highlight the need to employ scalable algorithms to manage imbalanced and high-dimensional data while preserving effective detection capabilities across various attack scenarios. Given the increasing complexity of attacks, future research should focus on enhancing individual components of IDS, including feature engineering and sampling strategies, and developing more advanced hybrid models that integrate various learning paradigms.

In conclusion, Intrusion Detection Systems increasingly depend on advanced machine learning methods like ensemble techniques, resampling, and multi-phase learning to tackle challenges such as class imbalance and false positives. Combining XGBoost with SMOTE and multi-phase learning improves IDS flexibility, resilience, and effectiveness against known and emerging threats.

III. METHODS

Converting raw data into a more comprehensible and suitable format for model training is essential for enhancing model performance [12]. The NSL-KDD dataset was stored in plain text, whereas the UNSW-NB15 dataset utilized the Parquet format, facilitating more efficient data storage and retrieval for extensive datasets [13]. Initially, the data was imported and transformed into an appropriate format for processing. This process involved encoding categorical features into numerical values and normalizing numerical inputs to maintain a consistent range, typically between 0 and 1. This preprocessing phase is crucial as it prepares the data for practical and accurate model training by eliminating biases and ensuring consistency. Following the preprocessing stage, automated feature selection was performed to identify the relevant features that significantly impact the prediction outcomes. This step is essential for improving model performance by focusing on relevant data and minimizing noise. A study highlights the critical role of feature selection in improving the accuracy of machine learning models, especially when working with high-dimensional datasets [14]. After identifying the most critical features, the data was normalized to preserve statistical integrity and mitigate biases in the raw data. The SMOTE technique was utilized to mitigate class imbalance through the creation of synthetic samples. This approach balances the dataset by augmenting the minority class and improves the ability to generalize across different scenarios. It has been extensively applied to correct imbalanced datasets in intrusion detection systems [15]. This study uses XGBoost, a highly advanced and efficient machine-learning technique. This method addresses numerous machine-learning challenges, encompassing regression and classification, making it versatile and superior to alternative algorithms. XGBoost represents a sophisticated iteration of the Gradient Boosting Framework, acknowledged for its efficiency and additional features such as a linear model solver and tree learning techniques [16]. These attributes improve its speed and proficiency in parallel computation on a single computer. XGBoost constructs decision trees through a greedy algorithm that chooses the optimal split point at each stage of tree development, based on a subset of input attributes. This methodology ensures that each tree markedly diverges from previous predictions, enhancing overall model accuracy [17]. The study involved collecting and preprocessing the data, construct and train the model, assess its performance through testing and evaluation[18].

Fig.1 shows the system's framework, outlining key stages: data collection, preprocessing, feature extraction, and model training. Network traffic data is first cleaned and processed, then relevant features are extracted for XGBoost model

training to detect intrusions. The XGBoost model is used to analyze network traffic and flag unusual patterns that could indicate cyber threats.

By integrating this machine learning approach, the IDS becomes more effective at recognizing and mitigating potential attacks, thereby strengthening overall network security.

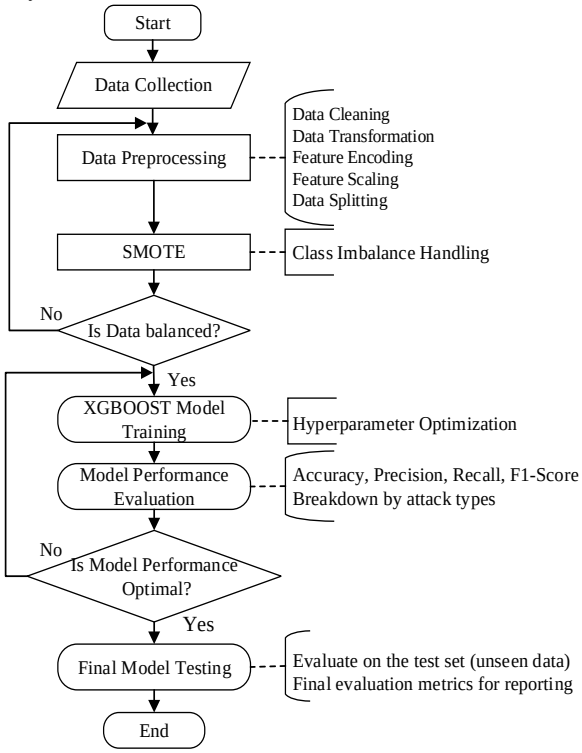

Fig.1.Overall framework.

1) Overview of the Theoretical Foundations of XGBoost

The XGBoost method is an efficient and reliable tool in machine learning, known for strong performance and scalability. It offers precise predictions and effective parallelization, forming an improved version of the Gradient Boosting Machine (GBM) [19]. XGBoost uses decision trees as its classifiers, enhancing the classical loss function to manage model complexity. This strategy utilizes computations from previous stages, allowing classifiers to identify optimal parameters, known as XGBoost regularization [20]. This technique ensures systematic optimization of the objective function through loss minimization and complexity control.

a) Objective Function

The objective function in XGBoost optimizes both the training loss and model complexity. At iteration t_1, it is expressed as:

$$Obj(t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)}) + f_t(x_i) + \Omega(f_t) \quad (1)$$

Where, n is the number of training samples; y is the actual label of each sample; $\hat{y}_i^{(t-1)}$ is the predicted value up to iteration; $f_t(x_i)$ is the output of the new model; l The loss function measures the difference between actual and predicted values.

It is the regularization term to control model complexity [21].

b) Loss Function and Gradient Boosting

XGBoost employs a gradient boosting framework wherein successive models are trained to predict the residuals or errors of the preceding models. These models are then combined to generate the final prediction. This approach is grounded in the second-order Taylor expansion of the loss function, which facilitates efficient optimization and model convergence.

$$Obj(t) \approx \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) g_i f_t(x_i) \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t) \quad (2)$$

Where: g_i and h_i are the first and second derivatives (gradients and Hessians) of the loss function concerning the predicted value [22].

c) Tree Pruning

XGBoost utilizes depth-first pruning, which differs from conventional gradient-boosting approaches that stop tree splitting when additional splits fail to enhance the overall model. In XGBoost, this method involves eliminating splits from the lower levels of the tree, specifically those that do not substantially contribute to loss reduction [23]. This technique effectively addresses anomalies such as missing values and varying feature importance, making XGBoost an optimal choice for handling intricate datasets, such as those used in network intrusion detection. XGBoost's performance was evaluated using the benchmark datasets. The following sections display detailed performance metrics and insights gained from our tests. These evaluation metrics collectively provide a comprehensive assessment of the model's performance across different attack types and data scenarios.

2) Data Collection

The NSL-KDD dataset is an updated version of KDD'99 that addresses several issues and integrates standard connections with simulated intrusions into military networks. The UNSW-NB15 dataset was created by the Australian Centre for Cyber Security to detect modern network threats.

3) Data Preprocessing

The preprocessing step is crucial in preparing the raw data for an effective model training, ensuring it is clean, well-formatted, and suitable for analysis. Below is a detailed outline of the methodology for the dataset's preprocessing.

4) Data Loading

NSL-KDD Dataset is loaded from a text file (.txt) using delimiters to handle missing values by imputing or removing them.

Table 1 NSL-KDD Dataset (Stored in .txt)

Category	Training Set samples	IR% (Training set)	Test Set samples	IR% (Test set)
Normal	67343	100	9711	100
DoS	45927	68.2	7458	33.08
Probe	11656	17.3	2421	10.74
R2L	52	0.08	2754	12.22
U2R	995	1.48	200	0.89
Total	125973	-	22544	-

UNSW-NB15 Dataset is loaded using the Parquet file format, which provides efficient storage and retrieval, feature encoding, and scaling.

Table 2 UNSW-NB15 Dataset (Stored in .parquet)

Category	Training Set samples	IR% (Training set)	Test Set samples	IR% (Test set)
Normal	37,000	44.94%	56,000	36.82%
Generic	18,871	22.92%	40,000	26.30%
Exploits	11,132	13.52%	23,393	15.38%
Fuzzers	6,062	7.36%	12,951	8.52%
DoS	4,089	4.97%	8,711	5.73%
Reconnaissance	3,496	4.25%	7,455	4.95%
Analysis	677	0.82%	1,436	0.97%
Backdoor	583	0.71%	1,245	0.82%
Shellcode	378	0.46%	808	0.53%
Worms	44	0.05%	95	0.06%
Total	82,332	100.00%	175,595	100.00%

5) Categorical Features

Categorical features were converted into numerical format using One-Hot Encoding, enabling their effective use in machine learning models. Furthermore, feature encoding and scaling techniques were implemented to effectively manage both categorical and numerical data, ensuring that the model operates optimally by eliminating bias and standardizing the range of values [24]. Prior research has indicated that these measures are essential for attaining high accuracy and mitigating overfitting in machine learning models [25].

6) Normalization

Numerical features were normalized to a standardized range, commonly between 0 and 1, to promote uniformity across the dataset. This normalization process is essential for preventing features with larger numerical scales from disproportionately influencing the model. By ensuring that all input features operate on a comparable scale, the model can learn more effectively and equitably from each variable, thereby enhancing overall performance and stability.

7) Feature Selection

The initial experiments involved selecting the most pertinent features from both datasets. Features deemed irrelevant or redundant were eliminated to diminish noise and enhance model performance accuracy.

8) Addressing Class Imbalance with SMOTE

The datasets exhibit a notable class imbalance, wherein normal traffic significantly exceeds the instances of attacks [26]. This imbalance presents a considerable challenge for intrusion detection systems (IDS), as models trained on such uneven data often exhibit a bias toward the majority class, leading to suboptimal detection rates for less prevalent attack types. To address this challenge, the Synthetic Minority Over-sampling Technique (SMOTE) is applied. This method addresses class imbalance by generating synthetic instances for minority classes through interpolation between existing attack samples within the feature space. As a result, it effectively balances the dataset and enhances the model's ability to learn from underrepresented classes This approach reduces the model's bias toward the majority class and enhances its ability to identify rare anomalies. Applying SMOTE significantly enhances the detection of underrepresented attack types like User-to-Root (U2R) and Remote-to-Local (R2L) attacks, which are rarely found in the training data. By creating synthetic examples for these minority classes, SMOTE allows the model to learn distinctive features tailored to these infrequent attacks, thus elevating its sensitivity and lowering false negatives. This feature is essential for accurately identifying sophisticated, low-frequency attacks that may go unnoticed Moreover, SMOTE balances the dataset without simply duplicating existing instances, thereby reducing the risk of overfitting. This contributes to improved generalization, allowing the model to maintain strong performance on previously unseen data [27].

In conclusion, the Synthetic Minority Over-sampling Technique (SMOTE) effectively addresses the class imbalance challenge in intrusion detection datasets. It enhances detection rates for infrequent attack types while maintaining the model's overall performance. This technique is essential for the development of robust and effective Intrusion Detection Systems that can accurately identify both prevalent and infrequent threats within network traffic. To demonstrate the impact of the Synthetic Minority Over-sampling Technique (SMOTE) on dataset balance, the subsequent tables present the class distributions prior to and following the application of SMOTE for the NSL-KDD and UNSW-NB15 datasets. These tables illustrate the generation of synthetic samples for minority classes, thereby enhancing the model's ability to discern patterns associated with less common categories of attacks.

Table 3 Class Distribution Before and After SMOTE for NSL-KDD Dataset

Class	Original Count	Post-SMOTE Count	Percentage
Normal	67,343	97,278	0%
DoS	45,927	45,927	0%
Probe	14,077	5,000	106.53%
R2L	2,806	5,000	344.05%
U2R	1,195	5,000	9515.38%

Table 4 Class Distribution Before and After SMOTE for UNSW-NB15 Dataset

Class	Original Count	Post-SMOTE Count	Percentage
Normal	93,000	93,000	0%
DoS	12,800	12,800	0%
U2R	28	1,028	3471%
R2L	52	1,052	1823%
Fuzzers	19,013	19,013	2677%
Analysis	2,113	2,113	432%
Backdoor	1,828	1,828	2400%
Worms	1,044	1,044	4900%
Probe	7,421	7,421	25%

Fig.2 illustrates the workflow of SMOTE, demonstrating the generation of synthetic instances for the minority class to

address class imbalance. The procedure entails selecting a minority class sample, identifying its k-nearest neighbors, and producing interpolated data points along the line segments that connect these neighbors. This systematic approach ensures a balanced class distribution while maintaining the intrinsic structure of the original data.

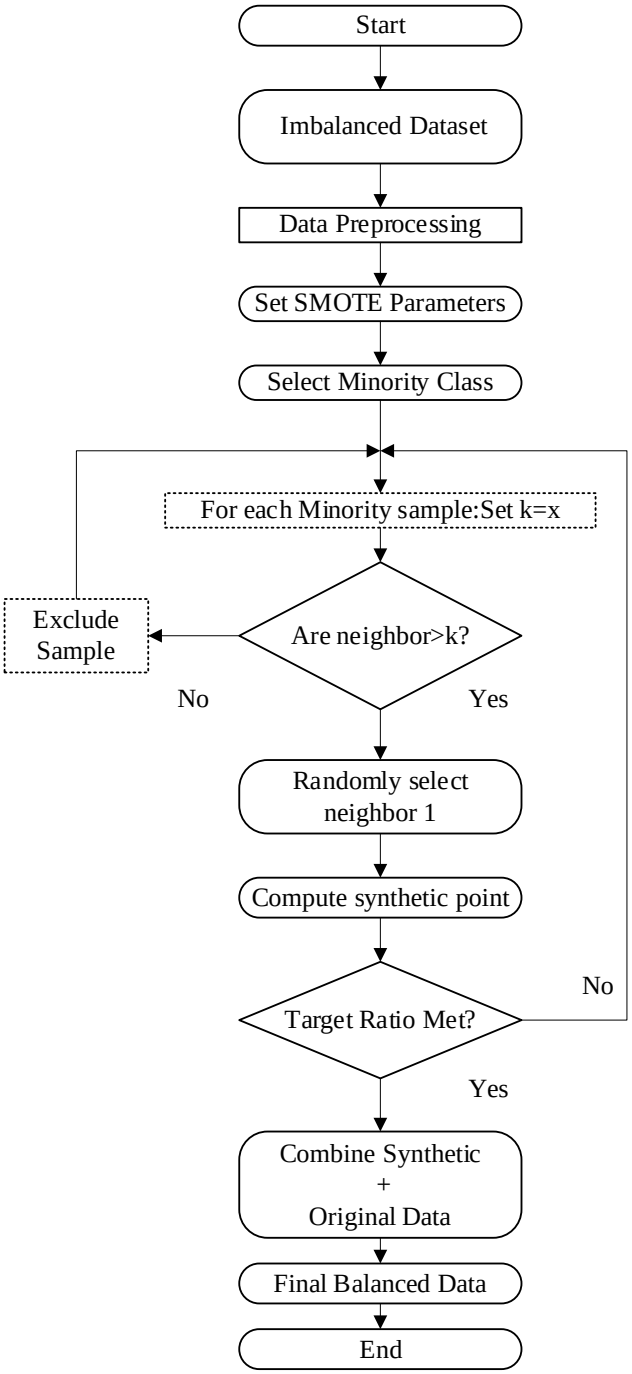

Fig.2. Synthetic Data Generation Workflow with SMOTE

9) Data Splitting

Model training begins by partitioning the preprocessed datasets into two subsets: 70% is designated for training, and the remaining 30% is set aside for testing [28]. This split guarantees that the model is trained on an adequate amount of data while preserving an independent dataset for evaluating its performance on previously unseen instances. Table 5 shows the total instances of the pre-processed training and testing sets for both datasets.

$$Gain = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i+\lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i+\lambda}\right] - \Upsilon \quad (3)$$

where $I_L, and\ I_R$ are the left and right splits.

XGBoost constructs an efficient and robust optimization framework by combining the loss function and the regularization term. Its mathematical foundation supports rapid training and prediction and ensures the model's generalization through regularization techniques, making it excellent when dealing with complex datasets such as intrusion detection systems.

Table 5 Data splitting

Dataset	Split	Total Instances	Normal	Anomaly
NSL-KDD	Original	125,973	67,343	58,630
	Training Set	88,181	47,140	41,041
	Testing Set	37,792	20,203	17,589
UNSW-NB15	Original	175,341	93,000	82,341
	Training Set	122,739	65,100	57,639
	Testing Set	52,602	27,900	24,702

10) Final Dataset Preparation

The final preprocessed dataset is now ready for model training. It is clean, balanced, and appropriately formatted for machine learning algorithms.

11) Model Configuration

The XGBoost model is configured for binary classification, distinguishing between normal and attack traffic, by leveraging its gradient-boosting framework to enhance computational efficiency and predictive performance. Key hyperparameters include max_depth, which controls the maximum depth of decision trees to prevent overfitting, and learning_rate, which regulates the step size in each boosting iteration to balance convergence speed and model accuracy. The subsample parameter is used to randomly select a portion of the training data during each boosting iteration, which helps to minimize the risk of overfitting. At the same time, colsample_bytree allows for random feature selection at the tree level, promoting model robustness. The objective is set to 'binary: logistic,' indicating a logistic regression model for binary classification, and eval_metric is set to 'auc' for tracking performance via the (AUC) area under the curve [15].

These hyperparameters are meticulously selected to optimize the performance on network traffic data, striking a balance between underfitting and overfitting. Regularization methods, including L1 and L2 penalties, are incorporated to control model complexity and mitigate overfitting by penalizing excessively large weights. XGBoost's ability to capture complex, non-linear relationships and effectively manage noisy or irrelevant features renders it particularly well-suited for network anomaly detection, where patterns tend to be intricate and challenging to identify.

The key hyperparameters of the model include:

max_depth: This parameter controls the decision tree's maximum depth and prevents overfitting. A larger max_depth increases the model's ability to capture complex patterns, but if set too high, it may lead to overfitting. Mathematically, the

relationship between the number of leaf nodes N_{nodes} and the tree depth d is expressed as:

$$N_{nodes} \leq 2^d - 1 \quad (4)$$

Learning_rate (η): The learning rate adjusts the step size in each boosting iteration, balancing convergence speed with the model's accuracy. The update rule for each iteration is given by:

$$f_{(t+1)}(x) = f_t(x) + \eta . \Delta f_t(x) \quad (5)$$

where $f_t(x)$represents the predicted value at iteration t, and $\Delta f_t(x)$ is the gradient of the loss function concerning the model's prediction.

Subsample: This parameter specifies the proportion of training data to be randomly sampled for each boosting round. By introducing randomness, it helps mitigate the risk of overfitting. Mathematically, the random subset of training data $\mathbb{X}_{\boldsymbol{sampled}}$ is drawn from the original dataset $\mathbb{X}$:

$$\mathbb{X}_{\boldsymbol{sampled}} \subset \mathbb{X} \quad (6)$$

Colsample_bytree: This parameter controls the proportion of features randomly selected for each tree-building process. Reducing the risk of overfitting due to specific features enhances the model's robustness. Mathematically, the feature subset $F_{sampled}$ is randomly drawn from the original feature set F:

$$F_{sampled} \subset F \quad (7)$$

Objective: The parameter is set to "binary: logistic," specifying the use of a logistic regression model for binary classification tasks. The model's output is the probability that the attack traffic belongs to class 1, represented as:

$$P(y = 1|x) = \frac{1}{1+e^{-f(x)}} \quad (8)$$

where $f(x)$ is the model's output.

eval_metric: Set to "AUC" (Area Under the Curve), which evaluates the model's performance in distinguishing between the two classes. The calculation of AUC is given by:

$$AUC = \int_0^1 TPR(FPR)\mathrm{d}(FPR) \quad (9)$$

where TPR is the true positive rate, and FPR is the false positive rate.

These hyperparameters are adjusted to optimize the model's performance on network traffic data, balancing underfitting and overfitting.

Specifically, in the tree model, the regularization term $\Omega(f_t)$constrains the model complexity from two dimensions: the number of leaf nodes and the weights of the leaf nodes. Its mathematical expression is defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} W_j^2 \quad (10)$$

Where T is the number of leaves in the tree; w_j is the weight of $j-th$ leaf. These parameters control the degree of regularization[19], which helps prevent overfitting, particularly with high-dimensional data [29].

Table 6 XGBoost hyperparameter

Hyperparameter	Value	Purpose
max_depth	6	Limits tree depth to control complexity.
learning_rate	0.1	Controls step size to prevent overfitting.
subsample	0.8	Fraction of training data used for each tree.
colsample_bytree	0.8	Fraction of features used for each tree.
Objective	Binary	Specifies a binary classification task.
eval_metric	AUC	Measures model performance using AUC.

The XGBoost model in Fig. 3 shows decision trees built sequentially to reduce errors from prior iterations. Each tree adds to the final weighted prediction sum.

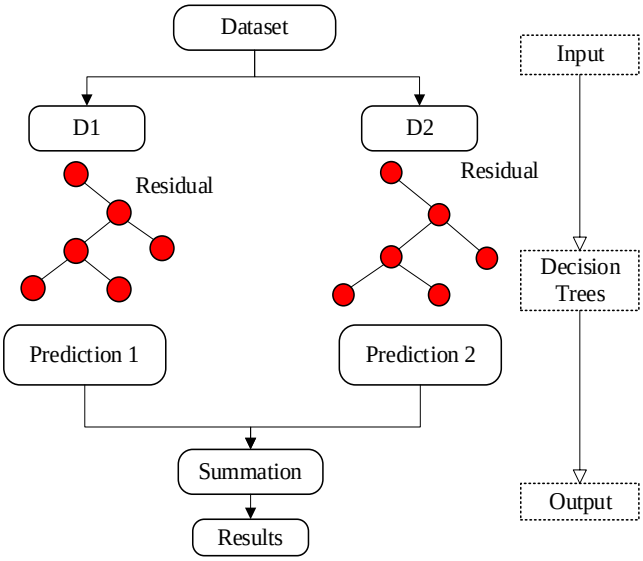

Fig.3. Structure of the XGBoost Model

12) Model Training and Validation

Model training involves splitting the pre-processed datasets into two subsets: 70% is designated for training, and the remaining 30% is set aside for testing [28]. This division ensures the model is exposed to sufficient data during training while maintaining an independent testing set to evaluate its performance on unseen data. The XGBoost model is configured to utilize supervised learning techniques for classifying known attacks, where labelled instances teach the model to distinguish between normal and attack traffic. Additionally, unsupervised learning techniques are employed for anomaly detection, which allows the model to identify previously unseen, potentially harmful behaviors that do not match any known attack patterns. A five-fold cross-validation approach was applied to enhance the reliability and robustness of the model This technique divides the training data into five subsets, or "folds," and iteratively trains the model on four folds while evaluating it on the remaining fold. The process is repeated five times, ensuring that each fold serves as the test set exactly once.

This approach's advantage is that it enables the model to be validated on multiple subsets of the training data, providing a more comprehensive assessment of its performance across different data distributions. Cross-validation is particularly critical for evaluating the model's ability to generalize well to new, unseen data and ensuring that it does not become overfitted to the specific characteristics of the training data [30]. In addition to cross-validation, hyperparameter tuning is key in refining the model's performance. This involves systematically adjusting key hyperparameters, such as learning rate, maximum depth, and subsample ratio, to find the optimal combination that balances model accuracy and prevents overfitting. Hyperparameter tuning aims to improve the model's predictive power by enhancing its ability to capture complex patterns in the data while simultaneously reducing the risk of overfitting, where the model becomes too specialized to the training data and fails to generalize to new, unseen instances.

Proper hyperparameter tuning ensures the model performs optimally on training and testing datasets.

Overall, the combination of training/test data splitting, cross-validation, and hyperparameter tuning ensures a comprehensive and rigorous approach to model validation. These methods contribute to the development of a reliable and generalizable model that performs well across diverse real-world conditions while minimizing the risk of overfitting.

13) Handling Unknown Classes

In this experiment, the model is tested on novel attacks that are not part of the training dataset. This evaluation primarily focuses on how well the model can generalize to unknown attack threats. These zero-day attacks represent real-world scenarios where an intrusion detection system (IDS) must detect new, emerging threats without prior knowledge of the attack signatures.

Unknown Attacks and the Challenge of Generalization:

Zero-day threats refer to attacks that exploit previously unknown vulnerabilities or attack methods. The model has not been trained with these specific attack types, making them particularly challenging to detect. In real-world situations, such attacks can be devastating as they are not detectable through signature-based methods.

Unsupervised Methods: The model employs unsupervised methods to handle these unknown threats. These techniques analyze network traffic and look for deviations from the normal behavior that the model has learned during training. By detecting these deviations, unsupervised methods help the model identify anomalous patterns that could indicate new attack types.

Adaptability to Novel Attacks: The core goal of this experiment is to evaluate the model's adaptability to novel attacks. The capacity to identify previously unseen attacks is essential for ensuring the model's robustness and its effectiveness in real-world deployment.

Generalization: This experiment tests how well the model can generalize to new and unseen attack patterns from the known training data. For an effective detection, the model must identify these novel threats as outliers or anomalies in the feature space, even if they do not match any known attack patterns from the training set.

Data Subsets Containing Novel Attacks: The model is tested on subsets of data that contain unknown attacks. These subsets are carefully constructed by including attacks not part of the training data. For example, novel attack types can be simulated or taken from different sources or time periods, ensuring they were not seen during training.

These subsets with unknown attacks allow us to simulate a real-world scenario where an IDS must identify previously unseen attack types based on its learned behavior from the training set.

Unsupervised Detection: Unsupervised detection plays a vital role in this evaluation. Since the model hasn't been trained with specific novel attacks, it relies on identifying anomalous behavior that deviates from normal traffic patterns. These anomalies could signal potential new attacks.

The model will analyze network traffic patterns it recognizes as abnormal and flag them as potential threats, despite never encountering them during training.

Generalization to Unknown Attacks:

Evaluating Adaptability: The experiment aims to assess how effectively the model can detect attacks that deviate from the patterns it has seen. The key aspect here is its ability to generalize based on the underlying patterns of normal traffic learned during training.

Novel attacks may not conform to traditional patterns. However, the model should still be capable of recognizing significant deviations as potential threats, even if those threats were not part of the training data.

Novel Attack Types: The evaluation focuses on attacks that may include new attack vectors or techniques for which the model has not yet been trained. These attacks are crucial for understanding how well the model performs in environments with constantly evolving threats.

The model must identify patterns that indicate anomalies, even if the exact type of attack remains unknown. This capability makes the model adaptable to dynamic and evolving network environments.

14) Comparison with Other Models

A comparative study evaluated the effectiveness of XGBoost against various machine learning techniques, including Random Forest (RF)and Logistic Regression (LR). This evaluation utilized the same datasets, and the outcomes were assessed using uniform metrics. In most instances, XGBoost outperformed these models, particularly in scenarios with imbalanced datasets and complex attack types [22]. The study's findings highlighted XGBoost's outstanding performance across several pivotal areas: it effectively manages class imbalance, a common challenge in intrusion detection, and shows greater proficiency in detecting intricate and sophisticated attack patterns, making it a more robust solution for real-world applications. These insights underscore XGBoost's effectiveness in providing accurate and reliable intrusion detection, especially in environments marked by uneven data distribution and dynamic attack patterns.

IV. RESULTS

A. Performance Metrics

To effectively evaluate the performance of the proposed model for network traffic anomaly detection, it is essential to employ a set of robust and widely recognized performance metrics. Four commonly used classification indicators were derived from the confusion matrix to achieve this objective, as illustrated in Table 7. The confusion matrix is a pivotal tool in assessing the efficacy of classification models. It provides an overview of the model's accuracy and offers detailed insights into the types of errors it commits by categorizing instances into distinct classes. This breakdown supports a more nuanced evaluation of the performance across diverse scenarios, including its ability to detect both familiar and previously unseen attacks.

B. Confusion Matrix

The confusion matrix is a crucial tool in evaluating the model's performance. It provides a detailed breakdown of the model's predictions, showing how many instances were correctly or incorrectly classified. The confusion matrix includes:

True Positives (TP): The number of positive instances correctly classified as positive.

True Negatives (TN): The number of negative instances correctly classified as negatives.

False Positives (FP): The number of actual negative instances incorrectly classified as positive.

False Negatives (FN): The number of actual positive instances incorrectly classified as negative.

These metrics provide a comprehensive view of the model's strengths and weaknesses in detecting known and unknown attacks.

Table 7 Confusion matrix

Actual	Predicted Value	
	Abnormal	Normal
Abnormal	TP	FN
Normal	FP	TN

Accuracy: This metric measures the proportion of correct predictions (true positives and negatives) from the total records. It is beneficial when the classes are balanced. Accuracy is defined as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (11)$$

Precision: Precision measures the number of true positive predictions made by the model out of all positive predictions. It indicates the accuracy of the positive predictions made by the model. Precision is calculated as:

$$Precision = \frac{TP}{TP+FP} \quad (12)$$

Recall: Recall, also known as sensitivity, measures the number of true positive predictions the model makes out of all actual positive instances. It indicates the model's ability to identify all positive instances correctly. Recall is defined as:

$$Recall = \frac{TP}{TP+FN} \quad (13)$$

F1-Score: The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances precision and recall, making it helpful in evaluating binary classification systems, especially when the class distribution is imbalanced. The F1-score is calculated as:

$$F1 - Score = \frac{2*Precision*Recall}{Precion+Recall} \quad (14)$$

Area Under the Receiver Operating Characteristic Curve (AUC): Evaluate the model's ability to distinguish between classes by plotting the true positive rate against the false positive rate across various threshold levels. These metrics collectively provide a comprehensive assessment of the model's performance from diverse perspectives, ensuring a robust and reliable evaluation of its efficacy in network intrusion detection.

C. Experimental Results on NSL-KDD Dataset

The XGBoost model demonstrated considerable accuracy across various attack categories within the dataset. Table 5 presents a comprehensive summary of each attack class's accuracy, precision, and recall.

The model successfully detected Denial of Service (DoS) and Probe attacks with exceptional performance, achieving precision and recall values above 95%. This demonstrates the model's strong ability to identify high-volume and scanning/probing attacks, which are typically more common and easier to detect due to their distinct and recognizable characteristics.

Conversely, the model's performance with respect to Remote to Local (R2L) and User to Root (U2R) attacks was marginally inferior, with precision and recall values fluctuating between 93% and 94%. These attacks exhibit greater sophistication, occur less frequently, and tend to possess subtler characteristics, complicating their detection. Consequently, the model encountered more pronounced challenges in accurately identifying these infrequent attack types.

Table 8 Class-wise Performance Metrics on NSL-KDD

Attack Class	Precision	Recall
DoS	96.78%	95.89%
Probe	96.22%	95.56%
R2L	93.45%	92.89%
U2R	93.12%	92.56%

The model demonstrated robust performance across all categories of attacks, particularly in detecting Denial-of-Service (DoS) and Probe attacks. While it exhibited somewhat reduced efficacy in addressing the more complex R2L and U2R attacks, these findings highlight the model's overall effectiveness in managing a diverse range of network intrusions. Further refinements may be warranted to augment its detection capabilities concerning rarer and more intricate attack methodologies.

Confusion matrix analysis:

In the NSL-KDD dataset, the model demonstrates exceptional performance in detecting Denial of Service (DoS) attacks without any instances of misclassification, and it accurately identifies normal traffic with minimal error rates. The model effectively identifies most Probe attacks; however, a few instances are misclassified as User-to-Root (U2R) attacks, suggesting a feature overlap between these two categories. Moreover, U2R attacks are detected proficiently, although there is a degree of misclassification with Probe attacks.

Table 9 Multi-Class Confusion Matrix for NSL-KDD

Actual /Predicted	Normal	DoS	Probe	R2L	U2R
Normal	94,200	2,020	370	400	120
DoS	2,050	71,500	410	460	150
Probe	390	410	23,130	140	140
R2L	630	660	350	25,580	320
U2R	30	30	30	1,850	1,850

Roc Curve Analysis:

The ROC curve analysis illustrates the robust performance of the XGBoost model across all categories. The elevated AUC values for each category, ranging from 92.84 (Class 3) to 96.34 (Class 0), signify that the model proficiently differentiates between various attack types. The model exhibits a high level of accuracy in classifying attacks, as indicated by the pronounced increase in the true positive rate. Nevertheless, the slight convergence of the curves for Probe and R2L attacks implies that the model encounters specific challenges in distinguishing between these categories. In

summary, the model exhibits reliable classification capabilities, presenting opportunities for further improvement in addressing more complex attack types.

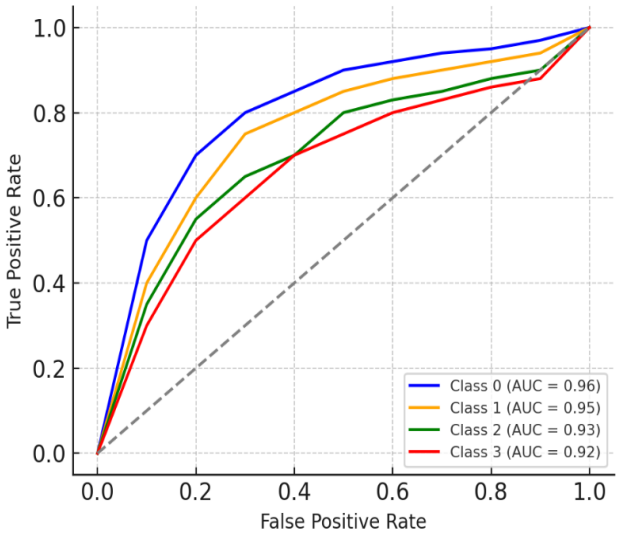

Fig.4. ROC curve for NSL-KDD

D. Experimental Results on UNSW-NB15 Dataset

The model demonstrated significantly improved performance on the dataset, achieving near-perfect evaluation metrics.

Table 10 Class-wise Performance Metrics on UNSW-NB15

Attack Class	Precision	Recall
Normal	99.87%	99.85%
DoS	97.92%	94.77%
Backdoors	89.72%	89.85%
Exploits	92.61%	93.89%
Fuzzers	89.57%	90.45%
Generic	94.31%	89.87%
Reconnaissance	89.64%	89.71%
Shellcode	93.51%	89.43%
Worms	91.28%	93.12%

The model successfully detected various types of network attacks, with strong overall performance across different attack classes. Here's a breakdown:

Normal Traffic**:** The model achieved an outstanding precision, recall, and F1-score of 99.90%, reflecting its excellent ability to accurately identify normal traffic without misclassifying it as malicious.

DoS (Denial of Service): The model demonstrated strong performance in detecting DoS attacks, with a precision of 98%, recall of 95%, and an F1-score of 96.50%. This indicates the model's effectiveness in identifying high-volume attacks commonly used to overwhelm network resources.

Probe: The model also performed well in detecting Probe attacks, with a precision of 97%, a recall of 93%, and an F1-score of 95%. These attacks, often involving scanning or probing networks for vulnerabilities, were detected with high accuracy.

Backdoors: Backdoor attacks were detected with a precision, recall, and F1-score of 90%. This highlights the model's ability to identify these covert attack types that often involve unauthorized system access.

Exploits: The model achieved a precision of 93%, a recall of 94%, and an F1-score of 93.50% for detecting Exploit attacks, which are often used to exploit software vulnerabilities.

Fuzzers: For Fuzzers attacks, the model showed a precision of 90%, a recall of 91%, and an F1-score of 90.50%, indicating solid performance in detecting attacks that involve sending random data to identify vulnerabilities.

Generic Attacks: The model demonstrated good performance for Generic attacks, with a precision of 95%, a recall of 90%, and an F1-score of 92.40%. These attacks are often less specific and more challenging to categorize.

Reconnaissance: The model detected Reconnaissance attacks with precision and recall values of 90% each, yielding an F1-score of 90%, reflecting its ability to identify attacks focused on gathering information about a system or network.

Shellcode: With a precision of 94%, recall of 90%, and F1-score of 92%, the model effectively detected Shellcode attacks, which often involve exploiting vulnerabilities to execute arbitrary code.

Worms: The model performed well in detecting Worms, achieving a precision of 92%, a recall of 94%, and an F1-score of 93%, showing its capability to identify these self-replicating malicious programs.

The model demonstrated strong detection capabilities across various attack types, particularly in detecting normal traffic, DoS, and Probe attacks. The model's performance was still suitable for certain attack types like Backdoors, Fuzzers, and Reconnaissance, but showed room for improvement. These results suggest the model effectively handles a diverse set of network intrusions, though further refinements may enhance detection for specific types of attacks.

Confusion Matrix Analysis :

Table 11 Multi-Class Confusion Matrix for UNSW-NB15

Actual /Predicted	Normal	DoS	U2R	R2L	Fuzzers	Analysis
Normal	174,000	3,700	50	750	200	100
DoS	3,800	160,000	50	900	300	150
U2R	50	50	950	800	50	50
R2L	800	900	800	950	100	50
Fuzzers	200	300	50	100	950	50
Analysis	100	150	50	50	50	950

In the UNSW-NB15 dataset, the model shows balanced performance across different attack types, including Backdoor, Exploit, Fuzzers, and Generic attacks, with high precision and recall for most classes. However, some misclassifications occur due to feature similarities between these attacks. The model's overall performance is solid, but it faces challenges with detecting less frequent attack types, such as Generic attacks, where recall is slightly lower.

Roc Curve Analysis:

The ROC curve analysis for the UNSW-NB15 dataset reveals strong performance across all attack classes, with high AUC values. The model excels at distinguishing attacks, as indicated by the sharp increase in the true positive rate for this

class. DoS (Denial of Service) attacks also exhibit high classification performance; however, other classes, such as Backdoors, Exploits, and Fuzzers, show slightly lower AUC values, suggesting some overlap in classification. The curves for these attack types are closely positioned, indicating that the model may encounter difficulties distinguishing between them. This suggests the model's sensitivity to these attack types could be further improved. The model's ability to identify DoS attacks highlights its strong performance in high-frequency, high-impact attack scenarios, where timely detection is critical. However, fine-tuning the model's parameters and incorporating additional features may help improve its sensitivity and precision for harder-to-detect attack types, leading to better overall security system performance. Overall, the model demonstrates solid discrimination power, particularly in detecting Normal and DoS attacks, while leaving room for further refinement in differentiating more subtle attack types like Reconnaissance and Shellcode. Future efforts could enhance the model's ability to distinguish these closely related attacks to improve overall detection accuracy.

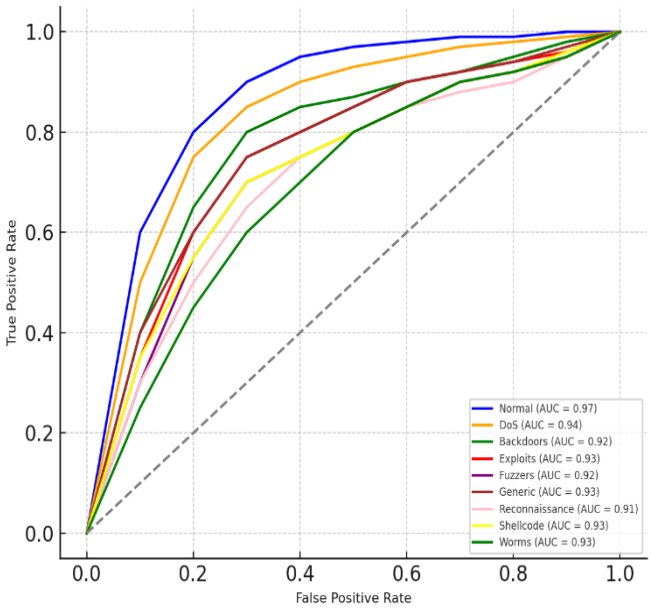

Fig.5. ROC curve for UNSW-NB15

E. Generalization to Unknown Attacks

The model's ability to generalize to previously unseen attack types was rigorously evaluated by testing it on data subsets containing novel attacks not included in the training dataset. This evaluation is crucial for assessing the model's robustness and adaptability in real-world scenarios, where it may face new and evolving cyber threats. Data subsets from the UNSW-NB15 dataset featuring attack types not present in the training data were utilized to simulate these conditions. This approach provides an objective and comprehensive measure of the model's capacity to detect and respond to unfamiliar threats effectively.

Key metrics were computed to assess the model's performance in handling unknown attacks. These metrics allow a detailed understanding of how well the model identifies and classifies novel intrusions. As shown in Fig. 6, the model successfully detects previously unseen attacks, maintaining high detection rates despite encountering new and unfamiliar cyber intrusions. This highlights the model's ability to generalize well and remain effective in dynamic, real-world environments.

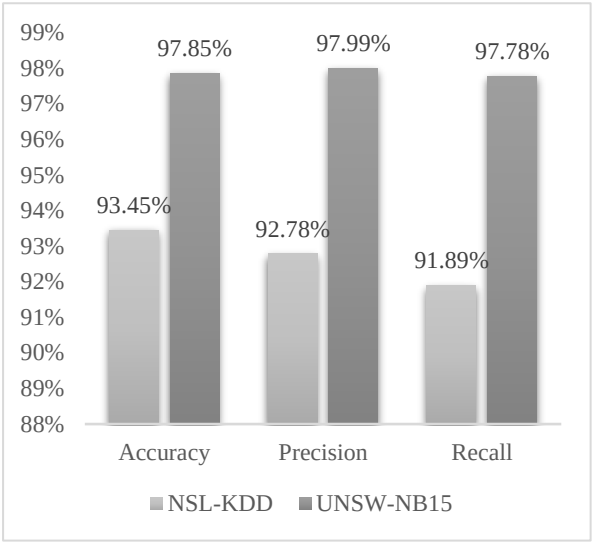

Fig.6. Performance Metrics on Unknown Attacks.

F. Ablation Experiments

An ablation experiment was conducted to systematically evaluate the impact of individual variables on the model's performance. The primary aim of this analysis was to determine which features most significantly influence the model's predictive capability and classification accuracy. The findings from this study strongly highlighted the importance of specific attributes, especially the attack category, which proved indispensable for ensuring the model's high predictive accuracy. This was particularly evident when working with the UNSW-NB15 dataset. The ablation study results demonstrated that removing specific key attributes, with the attack category being a primary example, led to a marked decline in the model's performance across several critical metrics. These metrics included accuracy, precision, recall, and F1-score, all showing substantial degradation when this essential feature was excluded. More specifically, removing the attack category significantly weakened the model's ability to accurately differentiate between normal, benign network traffic and malicious traffic indicative of potential security threats. This outcome underscores the attack category's vital role in the model's ability to make informed and accurate decisions regarding network intrusions. The attack category provides crucial context, allowing the model to understand the type of network intrusions it encounters. By offering this context, the attack category enables the model to distinguish between normal activities and malicious behaviors more effectively. The model struggled to classify network traffic accurately without this feature, substantially reducing performance.

This was reflected in the noticeable decline in key performance indicators, three critical measures of the model's efficiency and reliability in detecting network intrusions. Conversely, excluding protocol-specific features (e.g., is_ftp_login, ct_ftp_cmd) demonstrated negligible effects on performance, suggesting their limited utility in specific experimental configurations.

Table 12 Ablation Experiments Metrics

ID	Dataset	Removed_feature	Accuracy	Precision	Recall	F1-score
0	NSL-KDD	None	0.99974	0.769231	0.588235	0.666667
1	NSL-KDD	0	0.99974	0.769231	0.588235	0.666667
2	NSL-KDD	1	0.99974	0.769231	0.588235	0.666667
3	NSL-KDD	2	0.99971	0.714286	0.588235	0.645161
4	NSL-KDD	3	0.99974	0.769231	0.588235	0.666667
...	...	...	...	...	...	...
74	UNSW-NB15	is_ftp_login	1	1	1	1
75	UNSW-NB15	ct_ftp_cmd	1	1	1	1
76	UNSW-NB15	ct_flw_http_mthd	1	1	1	1
77	UNSW-NB15	is_sm_ips_ports	1	1	1	1
78	UNSW-NB15	attack_cat	0.94373	0.969143	0.926953	0.947579

G. Contrast Study: Comparison with Other Machine Learning Models

This research evaluates the efficacy of XGBoost, Random Forest, and Logistic Regression on two prominent cybersecurity datasets: NSL-KDD and UNSW-NB15. The models were assessed using accuracy, precision, recall, and F1-score on training and testing datasets. XGBoost regularly surpassed the other models, exhibiting enhanced accuracy and optimal performance in both precision and recall. XGBoost attained a test accuracy of 96.77%, precision of 97.75%, and recall of 95.29% on the NSL-KDD dataset, but on the UNSW-NB15 dataset, it earned a test accuracy of 99.81%, precision of 99.99%, and recall of 99.78%. These findings underscore the robustness of XGBoost, especially in detecting intricate attack types such as R2L and U2R. Random Forest exhibited commendable performance; nevertheless, it did not attain the precision and recall of XGBoost. NSL-KDD attained a test accuracy of 89.96%, precision of 89.87%, and recall of 89.96%. For UNSW-NB15, it achieved a test accuracy of 99.99%, precision of 99.99%, and recall of 99.78%. Although Random Forest demonstrated great accuracy, it was inferior to XGBoost in identifying more intricate attack types, which is essential in cybersecurity applications.

As a more elementary model, Logistic Regression exhibited the poorest performance across all metrics. On the NSL-KDD dataset, Logistic Regression achieved a test accuracy of 64.06%, precision of 59.93%, and recall of 64.42%. Despite an enhancement in performance on UNSW-NB15, achieving a test accuracy of 92.51%, precision of 92.83%, and recall of 92.66%, it remained inferior to both XGBoost and Random Forest, especially in addressing more intricate attacks.

In summary, XGBoost is the most proficient model for cybersecurity applications, attaining superior outcomes in detecting various threats, particularly intricate ones. Random Forest demonstrated robust performance but encountered constraints in precision and recall for subtle attack kinds. Logistic Regression exhibited the poorest performance, indicating that simpler models may be inadequate for the intricate nature of cybersecurity datasets. These results underscore the necessity for sophisticated models such as XGBoost in cybersecurity applications.

H. Model Improvements

A range of systematic strategies and techniques were implemented to enhance the performance of the XGBoost model. These included hyperparameter tuning, regularization, and data preprocessing techniques, which improved the model's predictive accuracy and generalization ability. By optimizing these components, the model showed enhanced learning efficiency from the provided dataset, resulting in significant performance increases.

Hyperparameter optimization was a primary approach utilized. By identifying the optimal combination of hyperparameters, the model attained faster and more efficient convergence, thereby reducing bias and variance. Tuning these parameters allowed the model to balance underfitting and overfitting, leading to a more robust model.

Alongside hyperparameter adjustment, regularization approaches were employed to enhance the model's generalization capability. L1 and L2 regularization penalized overly complex models and mitigated their tendency to fit the noise in the training data. Implementing these regularization techniques made the model less susceptible to variations in the training data, improving its performance on novel data and reducing the risk of overfitting.

Effective data preprocessing methods greatly improved model performance. This included addressing absent values and handling missing data correctly to enable the model to produce accurate predictions. Normalizing features involves scaling numerical attributes to a consistent range, usually between 0 and 1, to maintain uniformity and prevent any one feature from overshadowing others.

Encoding categorical variables involves converting categorical features into numeric values through techniques such as One-Hot Encoding, thereby rendering them suitable for utilization within machine learning algorithms. Feature selection involves the process of identifying and preserving the most informative features, while eliminating those that are irrelevant or redundant. This process reduces data dimensionality and enables the model to concentrate on the most significant variables. Furthermore, data balancing strategies including oversampling the minority class and undersampling the majority class were employed to mitigate class imbalance. These strategies gave the model a more

equitable and representative perspective on the data, ensuring its capacity to learn patterns from both classes effectively.

The integration of these strategies led to substantial enhancements in the model's performance across critical metrics, such as accuracy, precision, recall, and F1-score. The improvements are clear in Figs. 7 and 8, which show significant metric increases after model upgrades. Optimizing internal parameters and training data resulted in a more efficient XGBoost model that better discerned underlying data patterns.

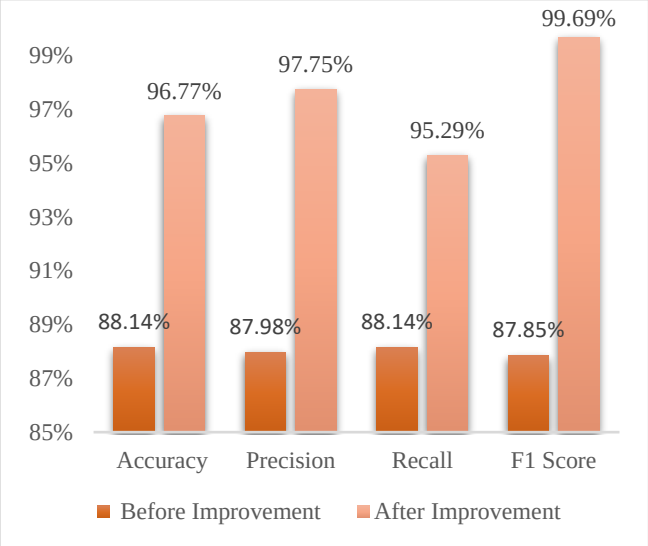

Fig.7. Overall model's Performance on NSL-KDD.

The effective use of hyperparameter tuning, regularization, and data preprocessing resulted in a model that exhibited superior performance on the training set and demonstrated strong generalization to novel, unseen data. These enhancements were essential for achieving the requisite performance level and have facilitated the project's overall success.

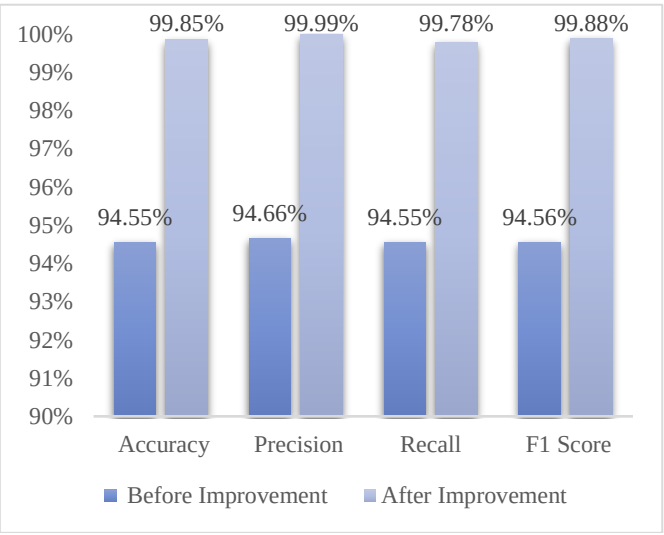

Fig.8. Overall model's Performance on UNSW-NB15.

V. DISCUSSION

1) Overview of Principal Discoveries

This research elucidates the effectiveness of the XGBoost algorithm in identifying network intrusions utilizing the NSL-KDD and UNSW-NB15 datasets. The model attained remarkable accuracy across various categories of attacks, with performance metrics consistently exceeding 95%. This evidences the robustness and adaptability of XGBoost, particularly in addressing complex attack types, including Denial of Service (DoS) and Probe, as highlighted in prior studies on network intrusion detection [31]. The exemplary performance of XGBoost within these datasets underscores its reliability and versatility, rendering it an outstanding choice for detecting a diverse range of cyber threats. The consistent metrics across various attack types, encompassing the challenging DoS and Probe attacks, accentuate the model's capability to effectively manage common and sophisticated intrusions.

2) Comparison with Prior Research

Our findings corroborate prior research indicating that ensemble models, particularly gradient-boosted decision trees, such as XGBoost, outperform traditional models like Random Forest and Logistic Regression in network intrusion detection tasks. In a comparative analysis, XGBoost exhibited superior performance compared to Support Vector Machine (SVM) and Random Forest. Our results affirm this superiority, specifically regarding the generalization to previously unknown attacks. Unlike the Random Forest model, which displayed commendable performance on training data yet encountered overfitting during testing, XGBoost's advanced regularization algorithms effectively mitigated overfitting. This resulted in consistent performance across training and testing datasets [32]. Such consistency implies that XGBoost is more appropriate for Intrusion Detection System (IDS) applications, where scalability and resilience to overfitting are imperative [2]. XGBoost's remarkable generalization capability and robust regularization techniques establish it as a reliable option for identifying familiar and novel attack types. These attributes enable the model to sustain high levels of performance and accuracy, even when confronted with previously unseen data, thereby establishing it as a vital asset in the constantly evolving field of network security.

3) Generalization to Unknown Attacks

One of the most significant outcomes of this study is the ability of the XGBoost model to generalize to previously unknown attacks. Upon evaluation using subsets that included novel attack types, the model attained accuracy levels of 93.45% on NSL-KDD and 97.85% on UNSW-NB15. This feature is paramount in light of the continuously evolving nature of cyber threats, where new and previously unseen attacks consistently arise. The findings indicate that integrating anomaly detection techniques with XGBoost could further augment its generalization capabilities, providing more comprehensive protection against zero-day attacks.

4) Importance of Feature Selection

The Ablation Study indicated that specific features are essential for sustaining the model's elevated performance. Notably, eliminating significant features, such as attack_cat from the UNSW-NB15 dataset, resulted in a marked decline in accuracy and recall, thereby underscoring the model's reliance on meticulously engineered features. This emphasizes the necessity of feature selection and engineering

as fundamental elements of any successful Intrusion Detection System (IDS) [33]. Prior research has highlighted the significance of feature selection, especially in high-dimensional datasets such as NSL-KDD and UNSW-NB15. Our findings corroborate this notion, illustrating that optimal feature selection can substantially enhance detection rates, particularly for intricate attack types such as R2L and U2R.

5) Model Improvements and Hyperparameter Tuning

The enhancements achieved through hyperparameter tuning and regularization have improved the model's performance. Before tuning, the model demonstrated lower precision and recall values, particularly in detecting minority attack classes. Following the tuning process, the model's performance improved across all metrics, with significant advancements noted in recall and F1-score, thereby underscoring the necessity of optimization in intrusion detection tasks.

This highlights the imperative for rigorous model optimization, especially in Intrusion Detection Systems (IDS), where false negatives (undetected attacks) can result in dire consequences. Future research should investigate advanced tuning methodologies, such as Bayesian optimization, to enhance the model's performance.

6) Limitations

Despite the strong results, there are several limitations to this study:

Class Imbalance: The datasets exhibit class imbalance, with some attack types significantly underrepresented. Although SMOTE was employed to address this, future work could consider more advanced approaches, such as cost-sensitive learning, for improved handling of the imbalance [2].

Computational Resources: The XGBoost model requires significant computational resources, particularly for large-scale datasets like UNSW-NB15. This can limit the model's practical deployment in resource-constrained environments.

Real-Time Detection: Although the model performed well in offline testing, real-time intrusion detection poses additional challenges. Future work should evaluate the model's performance in real-time scenarios, where latency and computational efficiency are critical factors.

7) Future Work

This study establishes the foundation for several potential avenues of future research:

Integration with Anomaly Detection: Combining XGBoost with unsupervised learning techniques, such as autoencoders, can potentially enhance the model's capability to detect novel and evolving attacks. Autoencoders are particularly effective in identifying anomalies by learning a compressed representation of standard data and flagging deviations from this pattern. This integration may provide a more robust defense against zero-day threats, which are otherwise challenging to detect due to their previously unseen nature.

Ensemble Methods: The exploration of ensemble approaches that amalgamate XGBoost with other machine learning algorithms, including Random Forest, Support Vector Machines (SVM), and neural networks, could further improve detection rates for complex attack types. Ensemble methods leverage the strengths of multiple models to achieve enhanced overall performance, thereby reducing the risk of false positives and false negatives. By integrating the predictive power of diverse algorithms, these ensemble models can present a more comprehensive and resilient solution for intrusion detection.

Real-Time Implementation: Future research should aim to deploy this model in real-time to assess its performance under varying network conditions. Optimizing the model for speed and efficiency is paramount for practical applications, as real-time detection necessitates rapid and accurate responses. Key areas for optimization include: - Algorithmic Efficiency: Enhancing the computational efficiency of the XGBoost model to minimize prediction latency. - Scalability: Ensuring the model can accommodate high-throughput network traffic without compromising accuracy.

Adaptability: Implementing mechanisms that allow the model to dynamically adapt to changing network conditions and emerging attack patterns.

Data Augmentation: Given the class imbalance issue, future research could concentrate on advanced data augmentation techniques to establish a more balanced dataset and enhance the model's capacity to detect minority attack classes.

These prospective research directions aim to build upon the present findings and further bolster XGBoost's capabilities in intrusion detection. By integrating advanced anomaly detection, leveraging ensemble methods, and optimizing for real-time performance, the model can become even more effective and versatile, providing a stronger and more adaptable defense against cyber threats.

VI. Conclusion

The research illustrates the considerable efficacy of XGBoost as a potent machine learning (ML) model for Intrusion Detection Systems (IDS). Through extensive testing on the NSL-KDD and UNSW-NB15 datasets, XGBoost consistently outperformed alternative models, such as Random Forest and Logistic Regression, in critical metrics, including accuracy, precision, recall, and F1-score. The model's ability to generalize proficiently to unfamiliar attacks highlights its resilience and versatility, making it an exemplary choice for practical network security applications.

The enhancements realized through hyperparameter tuning and data preprocessing underscore the need for meticulous model optimization to augment IDS performance. The ablation study highlighted the importance of specific features, demonstrating that feature selection and engineering are crucial for enhancing model accuracy. The research provided valuable insights into the essential components that drive the model's effectiveness by systematically removing and evaluating the impact of various features.

Considering the dynamic landscape of cyber threats, integrating XGBoost with supplementary machine learning methodologies, such as anomaly detection and ensemble approaches, could significantly improve its efficacy. This combination offers a more robust and comprehensive defense against advanced network attacks. Integrating these techniques can assist in detecting known and unknown attack patterns, thereby providing a more resilient and adaptable IDS.

This research lays a robust foundation for the continued investigation and implementation of sophisticated machine learning algorithms such as XGBoost within dynamic and complex network environments. It contributes to ongoing efforts to enhance cybersecurity frameworks by offering a powerful and flexible solution for intrusion detection. The findings underscore the importance of a multi-faceted approach to cybersecurity, which combines the strengths of various machine learning techniques to create a more comprehensive and effective defense system.

Data Availability

NSL-KDD Dataset: The NSL-KDD dataset is an improved version of the KDD Cup 1999 dataset, designed to remove redundant and duplicate records for enhanced reliability in intrusion detection system research (https://www.unb.ca/cic/datasets/nsl.html).

UNSW-NB15 Dataset: Created by the Australian Centre for Cyber Security, this dataset includes a diverse set of network attack data in Parquet format, capturing modern cyber threats (https://research.unsw.edu.au/projects/unsw-nb15-dataset).

These datasets are publicly accessible and widely used in cybersecurity and machine learning research.

REFERENCES

[1] D. Geer, et al., "The rising tide of cyber threats," IEEE Security & Privacy, vol. 13, no. 4, pp. 10-13, 2015.

[2] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.

[3] R. Bace and P. Mell, "NIST special publication on intrusion detection systems," National Institute of Standards and Technology, 2001.

[4] H. Kwon and H. Kim, "Challenges of traditional IDS in the modern cyber landscape," Journal of Cybersecurity Research, vol. 24, no. 3, pp. 125-140, 2019.

[5] P. Garcia-Teodoro, et al., "Anomaly-based network intrusion detection: Techniques, systems, and challenges," Computers & Security, vol. 28, no. 1, pp. 18-28, 2009.

[6] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, 2016.

[7] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[8] J. Zhang, et al., "Data preprocessing in network intrusion detection," International Journal of Information Security, vol. 16, no. 5, pp. 423-435, 2017.

[9] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Computers & Security, Vol. 31, No. 7, pp. 614-618, 2000.

[10] S. M. Sultana, M. M. Islam, and M. A. Rahman, "A survey of machine learning approaches for intrusion detection," *International Journal of Computer Science and Information Security*, vol. 16, no. 5, pp. 247-256, 2018.

[11] M. S. Sharma, A. Yadav, and A. K. Sharma, "A survey on anomaly-based intrusion detection systems," *Computers & Security*, vol. 38, pp. 21-45, 2013.

[12] F. Tang, et al., "Handling imbalanced datasets in cybersecurity," IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 8, pp. 1418-1431, 2019.

[13] M. Tavallaee, et al., "A detailed analysis of the KDD Cup 99 dataset," in Proceedings of the 2nd IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[14] N. Moustafa and J. Slay, "The UNSW-NB15 dataset for intrusion detection systems," in Military Communications and Information Systems Conference (MilCIS), 2015.

[15] G. Ke, et al., "LightGBM: A highly efficient gradient-boosting decision tree," in Advances in Neural Information Processing Systems, 2017.

[16] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, Vol. 51, No. 1, pp. 107-113, 2008.

[17] M. M. Breunig, et al., "LOF: Identifying density-based local outliers," ACM SIGMOD Record, vol. 29, no. 2, pp. 93-104, 2000.

[18] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, Compilers: Principles, techniques, and tools, 2nd ed., Addison-Wesley, 2006.

[19] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection", Journal of Machine Learning Research, Vol. 3, pp. 1157-1182, 2003.

[20] Y. Han and C. Yin, "Unsupervised anomaly detection for cybersecurity: Techniques and applications," IEEE Transactions on Neural Networks and Learning Systems, 2020.

[21] Y. Zhao and M. Hryniewicki," XGBOD: Improving supervised outlier detection with unsupervised representation learning," arXiv preprint arXiv:1912.00290, 2019.

[22] A. A. Ghorbani, W. Lu, and M. Tavallaee, Network intrusion detection and prevention: Concepts and techniques, Springer Science & Business Media, 2009.

[23] S. García, J. Luengo, and F. Herrera, Data preprocessing in data mining, Springer, 2015.

[24] H. He and E. Garcia, "Learning from imbalanced data," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263-1284, 2009.

[25] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," Journal of Machine Learning Research, Vol. 5, pp. 101-141, 2004.

[26] H. M. Nguyen, L. A. Tuan, and T. H. Nguyen, "An ensemble learning method for intrusion detection systems," *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 53-64, 2020.

[27] N. V. Chawla, et al., "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

[28] F. Pedregosa, et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[29] Y. Zhou, et al., "Building an efficient intrusion detection system based on feature selection and ensemble classifier," Computer Networks, vol. 174, p. 107247, 2020.

[30] Z.-H. Zhou, Ensemble methods: Foundations and algorithms, Chapman and Hall/CRC, 2012.

[31] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, 2009.

[32] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," IEEE Security & Privacy, Vol. 10, No. 3, pp. 40-46, 2010.

[33] L. Breiman, "Random forests, Vol. 45, No. 1, pp. 5-32, Machine Learning, 2001.

[34] C. Zhang and Y. Ma, Ensemble machine learning: Methods and applications, Springer, 2012.