International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, Delaware, USA
Volume 13– No. 2, February 2026 – www.ijais.org

# An Optimized Feature-based Lightweight Intrusion Detection System with Dual Classifiers

Omolola M. Bobade
Department of Computer Science, Federal University of Technology, Akure, Nigeria

Chukwuemeka C. Ugwu
Department of Data Science, Federal University of Technology, Akure, Nigeria

Adebayo O. Adetunmbi
Department of Data Science, Federal University of Technology, Akure, Nigeria

Olumide O. Obe
Department of Computer Science, Federal University of Technology, Akure, Nigeria

## ABSTRACT
As network environments grow increasingly complex, so too do the patterns and sophistication of cyberattacks, necessitating more effective and intelligent security mechanisms. Machine learning-based intrusion detection systems (ML-IDS) have emerged as promising tools; however, they frequently encounter challenges such as low detection accuracy and high false positive rates. This study presents an enhanced ML-IDS that integrates Support Vector Machine (SVM) and Decision Tree (DT) classifiers, evaluated on the benchmark UNSW-NB15 dataset. The preprocessing pipeline involved converting categorical attributes into numerical representations, followed by Min-Max normalization to ensure uniform feature scaling. Feature selection and extraction were performed using Information Gain and Principal Component Analysis to reduce dimensionality and retain the most informative features. The classifiers were trained on the first top ten features for both to distinguish between normal and malicious traffics. Experimental results show that C4.5 DT outperforms SVM on both reduced feature reduction methods as it returns accuracy, precision, recall, and F1-score of 99.99%, 99.99%, 100% and 100% respectively on PCA, and 99.26%, 98.83%, 99.83% and 99.33% on IG feature selection. SVM returns the highest accuracy, recall, and F1-Score of 79.39%, 90.08% and 83.46% on PCA against 72.56%, 71.01%, and 82.00% on IG.

## Keywords
Intrusion Detection System, Machine Learning, Support vector Machine, Network Security, Decision Tree, Feature Selection.

## 1. INTRODUCTION
In the context of cybersecurity, intrusion refers to unauthorized access to a computer system or network with the intent to compromise the integrity, confidentiality, or availability of data [1]. Intrusions may be categorized as either passive, where an attacker covertly monitors or intercepts communication, or active, involving direct disruption or manipulation of system operations. These attacks can originate from external sources, such as hackers, and crackers or from internal actor (masqueraders), including employees, customers, or partners.

The consequences of successful intrusions are severe and far-reaching. They can result in privacy violations, unauthorized access to confidential resources, and critical disruptions to system functionality [2]. A comprehensive understanding of attack mechanisms is therefore crucial for developing robust intrusion detection systems (IDS). According to [3], network intrusions can be grouped into four categories, namely Denial-of-Service (DoS), Remote to Local (R2L), User to Root (U2R), and Probe attacks. DoS attacks aim to make network services unavailable by overwhelming system resources with excessive requests; notable examples include Smurf and Neptune attacks. R2L involves gaining unauthorized access to a local machine from a remote location, often through methods such as password guessing or exploiting unsecured services like file transfer protocols. U2R attack is a type of intrusion where an attacker gets limited user privileges and then exploits system vulnerabilities to gain more privileges up to the root or administrative levels. Probing attacks are typically preliminary efforts in which an attacker scans a network to identify exploitable vulnerabilities.

IDSs are designed to identify and respond to these malicious activities before they cause significant damage to the system [4]. IDSs can be classified according to their source of data, mode of response, and detection. Based on the source of data, IDSs are grouped into three categories; Host-Based IDS (HIDS), which monitor activities occurring within individual computing devices; Network-Based IDS (NIDS), which analyze traffic flow across the network infrastructure [5]-[8]; and Hybrid IDS, which combines host-level and network-level data to provide more holistic security framework [9].

Based on their response mode. IDS can either be a passive or active system. The passive IDSs operate by logging threats and notifying system administrators, while active IDSs take automated, real-time actions, such as blocking suspicious IP addresses, to mitigate the detected threat. Based on detection methodology, IDSs are typically categorized into signature-based and anomaly-based systems. These are referred to as the traditional detection approaches. Signature-based IDSs operate by comparing current network behavior against a database of known attack signatures, offering high accuracy for known threats, while the anomaly-based IDSs model normal system behavior and flag deviations from this norm as potential intrusions. Figure 1 shows a graphical representation of IDSs categories.

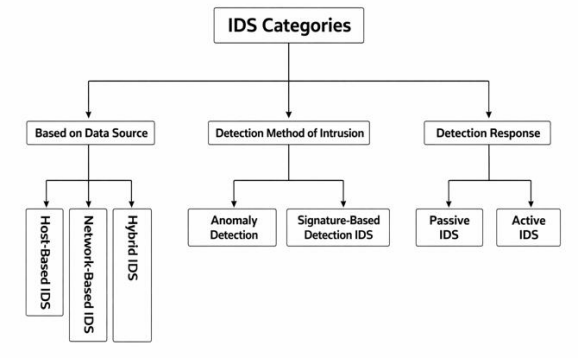


**Fig 1: Categories of intrusion detection system**

However, with the increasing sophistication of cyberattacks, traditional IDS detection approaches face significant limitations. The signature-based IDS rely heavily on predefined attack patterns and known threat signatures. As a result, they struggle to detect novel, zero-day, and polymorphic attacks that continuously evolve to evade static detection rules. While the anomaly-based IDSs suffer from high false positives [10][11]. To address these challenges, recent advances in artificial intelligence (AI), particularly machine learning (ML), have been increasingly incorporated into IDS design. ML-based IDSs are capable of automatically learning patterns from network traffic data and distinguish between normal and malicious traffics.

Despite the effectiveness of ML-based IDSs, they often face challenges related to computational complexity, especially when processing high-dimensional network traffic data. The performance and efficiency of these systems are greatly influenced by the quality and dimensionality of the input features [12]. High-dimensional data can lead to increased training time, resource consumption, and overfitting. To address this, feature reduction techniques are employed. These techniques fall into two main categories: feature selection and feature extraction. Feature selection involves identifying the most relevant subset of features from the original dataset, while feature extraction transforms the original features into a new, lower-dimensional space, retaining the most significant information. Both methods aim to reduce computational complexity while enhancing detection accuracy [13].

This study proposes the development of a lightweight, network-based intrusion detection system that leverages both SVM and Decision Tree (DT) algorithms. To optimize detection performance, the system incorporates feature extraction via Principal Component Analysis (PCA) and feature selection using Information Gain (IG). By combining these reduction techniques with supervised ML classifiers, the system aims to improve classification accuracy and reduce false-positive rates, thereby enhancing the overall efficiency and robustness of network intrusion detection.

The remaining part of the paper is sectioned as follows: a review of related work in Section 2. materials and methodology are described in Section 3. Section 4 is the Validation of the classification models. Section 5 presents discussion of results and comparison with other works, while Section 6 presents the conclusion of the study.

## 2. RELATED WORKS

Extensive research has focused on improving network intrusion detection systems (IDS) through various approaches and technologies.

[14] introduced a hybrid IDS combining Honeypots and IDS technologies to create an autonomous, self-learning system. This approach used a Markov model for probabilistic attack prediction but struggled with detecting unknown attacks. Similarly, [15] proposed a two-level IDS with fine-grained and coarse-grained detection mechanisms, utilizing a decision tree algorithm to enhance performance and reduce false alarms, achieving a 93% accuracy rate on the DARPA KDD CUP 99 dataset. Several studies have explored machine learning techniques and feature-based approaches. [16] used Self-Organized Feature Maps (SOFMs) for anomaly detection, creating a vote-based ranking system that improved detection rates for specific attack types but faced high false positive rates for DoS and R2L attacks. [17] applied k-means clustering to the NSL-KDD dataset, noting that efficiency depended on the number of clusters used and observed improved performance when the number of clusters matched the data types. However, their model still struggled with detecting malicious data. [18] utilized Hypothesis Testing with a decision tree algorithm to classify intrusion attacks on the KDD'99 dataset. Their system, focused on detecting DoS attacks, showed limitations in scope. [19] proposed a decision tree-based IDS with a 79.52% accuracy and a high positive rate of 60%, highlighting the effectiveness and limitations of their approach.

[20] compared association rule-based IDS with SVM-based IDS on UNSW-NB15 and NSL-KDD datasets, finding that SVM-based IDS outperformed the association rule method with an accuracy of 90.41%. However, both methods were computationally intensive. [21] developed a lightweight IDS using information gain and multilayer perceptron (MLP), achieving 76.96% accuracy which is relatively low. [22] used singular value decomposition and long short-term memory networks to detect DDoS attacks with over 90% accuracy, though their system was limited to DDoS detection.

[23] introduced a stacked IDS combining k-nearest neighbor, Naïve Bayes, and decision tree algorithms with meta-algorithms. This ensemble improved classification accuracy by 3% over base models but involved trade-offs in computation speed and memory usage.

[24] developed an improved intrusion detection system (IDS) by combining genetic algorithm-based feature selection with machine learning models Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP). Using the UNSW-NB15 dataset, they applied preprocessing techniques and evaluated model performance through confusion matrices and ROC curves. Results showed that MLP outperformed SVM, achieving a classification accuracy of 89.97% and an overall optimization capability of 92.39%, highlighting its effectiveness for IDS tasks.

**Table 1. Summary of related works**

| Author | Dataset | Method/Approach | Limitations |
|---|---|---|---|
| Obimbo *et al.,* [16] | KDD'99 | Self-Organizing Feature Maps (SOFMs) with vote-based ranking | High false positive rates for DoS and R2L attacks |
| Boulaiche *et al.,* [14] | Not Specified | Hybrid IDS using Honeypots + IDS; Markov model | Poor at detecting unknown attacks. |

| | | | |
|---|---|---|---|
| Aladesote et al., [18] | KDD'99 | Hypothesis Testing + Decision Tree for DoS detection | Limited to DoS attacks. |
| Al-mamory & Jassim [15] | DARPA KDD CUP99 | Two-level IDS with fine/coarse detection; Decision Tree classifier | Dataset lacks modern attack types |
| Duque & Omar [17] | NSL-KDD | K-means clustering-based anomaly detection | Limited ability to detect malicious data. |
| Rai & Guleria [19] | NSL-KDD | Decision Tree-based IDS | Moderate accuracy (79.52%). |
| Sarumi et al., [20] | UNSW-NB15, NSL-KDD | Comparison of Association Rule vs. SVM-based IDS | Computationally intensive models |
| Mebawondu et al. [21] | UNSW-NB15 | Lightweight IDS using Information Gain + MLP | Low accuracy (76.96%). |
| Olasehinde [23] | UNSW-NB15 | Stacked ensemble of KNN, Naïve Bayes, Decision Tree with meta-algorithms | Involved trade-offs in computation speed and memory usage. |
| Ugwu et al. [22] | UNSW-NB15, NSL-KDD | Singular Value Decomposition + LSTM for DDoS detection | Only detects DDoS attacks |
| Popoola et al., [6] | UNSW-NB15 | SVM and MLP | High misclassification rate of 20.35% |

# 3. METHODOLOGY

## 3.1 Proposed System Architecture

The figure 2 presents the architecture for the proposed NIDS. The process is organized into sequential stages that comprised data source and description, data preprocessing, feature reduction, model training, and final evaluation, ensuring an efficient NIDS model.

The system begins with the acquisition of the already partitioned UNSW-NB15 datasets: a training set and a test set. Thereafter, the two partitioned datasets undergo some preprocessing steps. This includes numericalization, where categorical attributes are converted into numerical values. Following this, normalization is applied to scale the features into a consistent range. Next, the preprocessed dataset is subjected to feature reduction, which is implemented using both feature extraction and feature selection strategies. Principal Component Analysis (PCA) is employed as a feature extraction technique to reduce dimensionality by transforming the feature space into a set of orthogonal components that preserve most of the variance. In parallel, Information Gain is used for feature selection to identify the most informative attributes based on their contribution to reducing entropy. The reduced feature set from the training set is then used to train two the distinct classifiers chosen for this study: Support Vector Machines (SVM) and Decision Trees (DT) simultaneously. The classifiers are trained differently and validated using the test dataset. Finally, the system proceeds to evaluation and classification reporting. The classification performance is assessed using appropriate metrics such as accuracy, precision, recall, and F1-score.

## 3.2 Data Source and Description

The UNSW-NB15 data set was sourced from https://research.unsw.edu.au/projects/unsw-nb15-dataset. The dataset was created by making use of IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) to extract a hybrid of modern normal activities and synthesized contemporary attack patterns of network traffic. A tcpdump tool was used to capture 100 GB of raw network traffic stored in pcap files. Each pcap file contains 1000 MB of captured network traffic. The 42 features of the dataset (Table 1 presents the features, positions, and datatypes) were generated by combining Argus and Bro-IDS techniques with twelve other algorithms. The generated features are partitioned into six groups, namely: flow features, basic features, content features, time features, and additional generated features. The UNSW-NB15 dataset contains 2, 540,044 records, which are stored in four CSV files. In one of the four files, the dataset is split into a training set and a testing set. The training set involved 175,341 records, while the testing set contained 82,332 records (Moustafa and Slay, 2015). The dataset reflects modern traffic scenarios and contains modern attack types. The attack types are categorized into nine groups, namely Reconnaissance, Shellcode, Exploit, Fuzzers, Worm, DoS, Backdoor, Analysis, and Generic. The binary class distribution of the dataset is presented in Table 2.

**Table 2. UNSW-NB15 Features, Position, and Datatype**

| F/No | Feature | Type | F/No | Feature | Type |
|---|---|---|---|---|---|
| F1 | dur | float | F22 | dtcpb | integer |
| F2 | proto | categorical | F23 | dwin | integer |
| F3 | service | categorical | F24 | tcprtt | float |
| F4 | state | categorical | F25 | synack | float |
| F5 | spkts | integer | F26 | ackdat | float |
| F6 | dpkts | integer | F27 | smean | integer |
| F7 | sbytes | integer | F28 | dmean | integer |
| F8 | dbytes | integer | F29 | trans_depth | integer |
| F9 | rate | float | F30 | response_body_len | integer |
| F10 | sttl | integer | F31 | ct_srv_src | integer |
| F11 | dttl | integer | F32 | ct_state_ttl | integer |
| F12 | sload | float | F33 | ct_dst_ltm | integer |
| F13 | dload | float | F34 | ct_src_dport_ltm | integer |
| F14 | sloss | integer | F35 | ct_dst_sport_ltm | integer |
| F15 | dloss | integer | F36 | ct_dst_src_ltm | integer |
| F16 | sinpkt | float | F37 | is_ftp_login | binary |
| F17 | dinpkt | float | F38 | ct_ftp_cmd | integer |
| F18 | sjit | float | F39 | ct_flw_http_mthd | integer |
| F19 | djit | float | F40 | ct_src_ltm | integer |
| F20 | swin | integer | F41 | ct_srv_dst | integer |
| F21 | stcpb | integer | F42 | is_sm_ips_ports | binary |

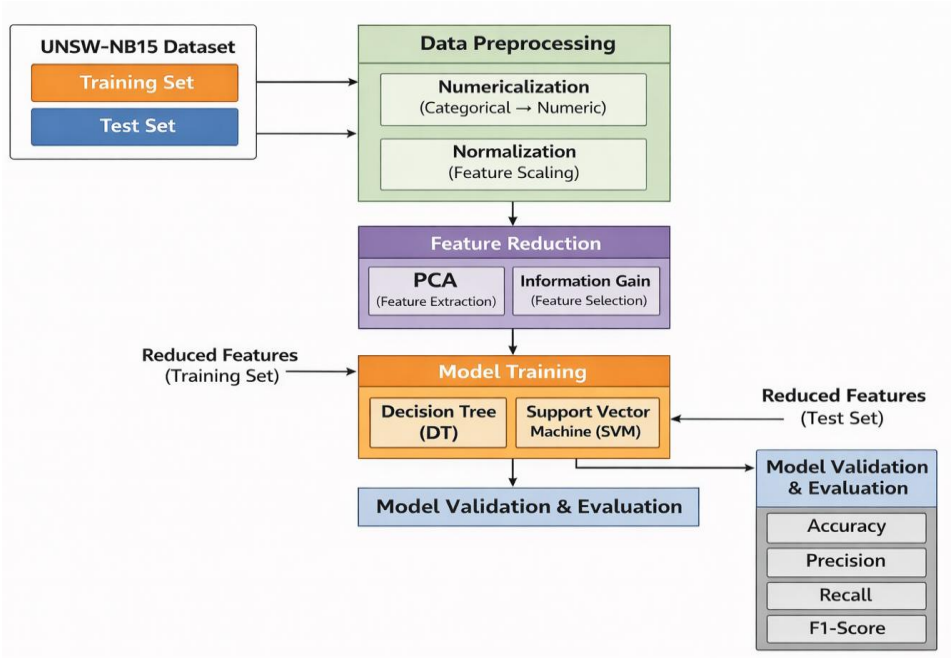


**Fig 2: Proposed machine learning based NIDS architecture**

**Table 3. Distribution of UNSW-NB15 Data**

| Class | UNSW-NB15 | | | |
|---|---|---|---|---|
| | Test | Test % | Training | Training % |
| Normal | 37,000 | 44.9 | 56,000 | 31.9 |
| Attack | 45,332 | 55.1 | 119,341 | 68.1 |
| Total | 82,332 | 100 | 175,341 | 100 |

## 3.3 Data Preprocessing

In this study, UNSW-NB15 was pre-processed by means of feature imputation, feature numericalization, normalization, and feature reduction. The pre-processing steps are presented as follows

### *3.3.1 Feature Imputation*

The heading for subsubsections should be in Times New Roman 11-point italic with initial letters capitalized and 6-points of white space above the subsubsection head.

### *3.3.2 Feature Numericalization*

This is the process of converting non-numeric data into a numerical format that machine learning algorithms can understand and process. Most machine learning models, especially traditional algorithms employed in this study input features to be numeric vectors. Here, integer values were assigned to unique values of Proto, Service, and State.

### *3.3.3 Feature Normalization*

In the proposed system, a feature normalization module is incorporated to address potential bias that may result from significant disparities in the magnitude of feature values. Such normalization enhances both the effectiveness of feature selection and the accuracy of classification models. To achieve this, the Min-Max normalization technique was applied to the UNSW-NB15 dataset. This method rescales feature values to fall within a defined range of 0 to 1, using the formula shown in Equation (1):

$$x'_{val} = \frac{x_{val} - min_{val}}{max_{val} - min_{val}} \quad (1)$$

where $x'_{val}$ represents the normalized value, $x_{val}$ represents the value to be normalized, $min_{val}$ represents minimum value of the dataset, and $max_{val}$ represents maximum value of the dataset. For illustration, where "Ct_Src_ltm" has a maximum value of 10 and minimum value of 1. Thus, normalization enables uniform scaling of all feature values, thereby mitigating the influence of dominant features and ensuring a balanced contribution to model training and evaluation.

### *3.3.4 Feature Reduction*

This subsection explains the activities of reducing network features to the most relevant ones for intrusion predictions which takes place at the feature reduction module. We adopted PCA for feature extraction and Information Gain (IG) for feature selection. The reason for choosing these techniques is to ensure optimal classification performance at reduced time. Reduced feature from each technique will be to train and validate the classifiers. Detailed analysis of how these techniques is presented as follows:

#### *3.3.4.1 Network Feature Extraction using PCA*

PCA is feature extraction technique widely used to reduce the number of dimensions (features) in data to set of most informative or relevant features that can better contribute to solving a problem in machine learning. PCA does this by creating a new set of features that is linear combination of the input features. These new set of features are called Principal Components (PC). However not all PC are used but a subset is chosen based on the percentage of the variance of the data to be maintained. In this study, network data were represented in matrix $X_{n*m}$, where n represents the number of traffic instances in rows and m represents the number features $f_k$ in columns. Thereafter, Matrix $X$ is centered to form a new matrix D using (2) and (3).

$$\bar{f_k} = \frac{1}{n-1}\sum_{i=1}^{n} v_i |\, f_k, k = 1,2,3,\dots,m \quad (2)$$

$$D = \sum_{i=1}^{n} (v_i - \bar{f_k}) |\, f_k, k = 1,2,3,\dots,m. \quad (3)$$

where each value of feature $f_k$in the data set is denoted by $v$ with a subscript $i$ that goes from 1 to $n$, $\bar{f_k}$ is denoted as the mean of a particular feature, and $D$ represents the centered data matrix. Furtherance to that, the covariance matrix of the whole dataset is computed by multiplying (transpose) with $D$ as expressed in equation (4):

$$C = \frac{1}{n-1} D^T D \quad (4)$$

where $C$ represents the covariance matrix. PCA involves obtaining set of eigenvalues $\lambda$ and eigenvectors $\mu$from the covariance matrix. The eigenvalues is computed using the formular expressed in equation (5)

$$(C - I\lambda_i)\mu_i = 0 \mid i = 1,2,3,\dots,R \quad (5)$$

where $\lambda_i$represents an instance of eigenvalue with its corresponding eigenvector $\mu_i$, and $R$ is the total number of eigenvalues. However, the number of eigenvectors (principal components) to be selected is determined using equation (6) as $j \in R$ numbers of eigenvectors $\mu$ were selected based on the magnitude of their corresponding eigenvalues $\lambda$.

$$\frac{\sum_{i=1}^{j} \lambda_i}{\sum_{i=1}^{R} \lambda_i} > threshold \quad (6)$$

where threshold value specifies the percentage of information to be retained.

### *3.3.4.2 Network Feature Selection using IG*

In this research, Information Gain (IG) is adopted as the second feature reduction technique to identify the most informative network attributes that effectively characterize traffic patterns. IG operates based on the concept of entropy, selecting features that provide the most reduction in uncertainty. Features with high information gain are considered to have lower entropy, making them more relevant and discriminative for the classification task, whereas features with low information gain contribute less to the decision-making process.

Each network feature is evaluated and ranked according to its IG score, with the feature possessing the highest gain ranked first. This ranking reflects the extent to which each attribute contributes to distinguishing between different traffic classes. To calculate the information gain for individual features, the process begins by determining the entropy of the entire dataset, which is based on the probability distribution of class labels. The entropy of the network data, denoted as S, is computed using the formulations provided in Equation (7).

$$E(S) = -\sum_{i=1}^{k} p_i log_2 p_i \quad (7)$$

where $E(S)$ represents the entropy of $S$, and $p_i$ represents the probability that a network instance in $S$ belong to a distinct class $c_i$ which ranges from $i$ to $k$.

Hence, the sample data is partitioned by the number of unique feature values under a particular feature category. Then the entropy of group of samples is computed using equation (8)

$$info_A(S) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v) \quad (8)$$

where $info_A$ represents the entropy of a particular feature $A$, $S_v$ represents the subsets of $S$ for feature $A$, $Values(A)$ represents all possible values contained in A which value $v$ is a member, and $E(S_v)$ denotes the entropy of $S_v$. Thereafter, the entropy of the entire dataset is subtracted from the entropy of the feature as expressed in equation (9)

$$I(S,A) = E(S) - info_A(S) \quad (9)$$

where $I(S,A)$ represents the information gain of the feature.

## 3.4 Machine learning model training

This section focuses on training the Decision Tree (DT) and Support Vector Machine (SVM) models using the UNSW-NB15 data that includes both normal and attack traffics. The objective is to enable the classification algorithms to learn the underlying patterns within the dataset, allowing them to distinguish between intrusive and normal network activities.

### *3.4.1.1 Support vector machine (SVM)*

This study selected SVM due to its high classification accuracy on binary datasets and robustness against overfitting. The objective of using a linear SVM is to effectively distinguish between normal and attack network traffic by finding an optimal separating hyperplane, as expressed in Equation (10):

$$W \,.\, x_i + b = 0 \quad (10)$$

where $W$ represents the weight vector, $x_i$ represents an instance from the network training set, and $b$ represents the bias factor. In this context, the normal traffic points lie above the hyperplane while the attack points are below the hyperplane, defined respectively by:

$$W \,.\, x_i + b \geq 1 \qquad if\ y_i = +1 \quad (11)$$

$$W \,.\, x_i + b \leq -1 \qquad if y_i = -1 \quad (12)$$

where $y_i \in \{+1, -1\}$ denotes the class labels as $+1$ for normal traffic and -1 for attack traffic. The SVM algorithm aims to find the hyperplane that maximizes the margin $M$ between the two classes, where

$$M = \frac{2}{||w||} \quad (13)$$

To achieve this, the goal is to minimize $\frac{1}{2}||w||^2$, subject to the constraint that all training samples are correctly classified. This leads to a constrained optimization problem, presented in equation (14)

$$w^*, b^* = \underset{w,b}{\operatorname{argmin}} \frac{1}{2}||w||^2 \quad \text{subject to } y_i(x_i w + b) - 1 \geq 0 \forall_i \quad (14)$$

To resolve this, the method of Lagrange multiplier is applied, introducing a multiplier $\alpha_i$for each training constraint. The resulting Lagrange function is:

$$\mathcal{L} = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i (x_i{}^T . w + b) - 1) \quad (15)$$

Expanding and rearranging give:

$$\mathcal{L} = \frac{1}{2} w^T w - w.\left(\sum_{i=1}^{n} \alpha_i y_i x_i^T\right) + b\left(\sum_{i=1}^{n} \alpha_i y_i\right) + \sum_{i=1}^{n} \alpha_i \quad (16)$$

where $\mathcal{L}$ represents the Lagrange function.

To find the optimal $\boldsymbol{w}$, $b, and\ \alpha_i$, we compute the partial derivative of $\mathcal{L}$ with respect to $w$ and $b$ as follows:

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w^* = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (17)$$

$$\boldsymbol{\frac{\partial \mathcal{L}}{\partial b}} = 0 \rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \quad (18)$$

By substituting these expressions back into the Lagrangian, we eliminate $w$ and $b$, resulting in the dual optimization problem:

$$\mathcal{L}_D = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i . x_j + \sum_{i=1}^{n} \alpha_i \quad (19)$$

where $x_i . x_j$ represents the product of two training samples

The dual optimization is maximized for $\alpha_i$, under the following constraints:

$$\alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i\, y_i = 0 \quad (20)$$

The dual formulation highlights a key aspect of SVMs: finding the maximum-margin hyperplane is equivalent to identifying the support vectors, i.e., training points with non-zero $\alpha_i$. These support vectors directly define the decision boundary through:

$$w^* = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (21)$$

The dual form of the optimization problem illustrates one important point. That is, it shows that searching for the maximum-margin decision boundary is equivalent to searching for the support vectors (these are the training samples with non-zero Lagrange multipliers), and the decision boundary is obtained.

#### *3.4.1.2 Decision tree (DT)*

The DT learning algorithm is employed to classify network traffic by recursively partitioning the dataset until each leaf node represents a distinct class either normal or a specific category of attack. This recursive partitioning process aims to learn hierarchical rules that can accurately distinguish various forms of network behavior based on observable features.

For this study, the C4.5 DT algorithm was adopted due to its simplicity, intuitive structure, and its capability to handle both discrete and continuous network features, a common characteristic of real-world intrusion detection datasets like UNSW-NB15. The C4.5 algorithm selects the most relevant features for splitting by computing the Gain Ratio (GR), which ensures a balanced decision tree by penalizing attributes with many distinct values.

The construction of the decision tree in this research involved four main steps:

1. Start with the entire network data $S$ with $c_1, c_2, \ldots, c_k$ as the root
2. Select the best attribute to split the network data based on the highest Gain Ratio. The gain ratio is computed by:
   a. Determining the entropy of $S$

$$H(S) = -\sum_{i=1}^{k} p(c_i) \log_2 p(c_i)$$

where $p(c_i)$ is the proportion of network traffic belonging to $c_i$ in set $S$

   b. Compute the information gain of each attribute to quantify the reduction in entropy in the attributes:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

where $S_v$ is the subset of $S$ where attribute $A$ has value $v$.

   c. Compute the Gain Ratio (GR) to address the bias of Information Gain towards attributes with many values.

$$GR(S, A) = \frac{IG(S, A)}{IV(A)}$$

where the $IV(A)$ is the intrinsic value is:

$$IV(A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|}\right)$$

   d. Select attribute with highest GR:

$$A^* = argmax_A\, GR$$

where $A^*$ is the attribute with highest gain ratio

3. Create a branch for each possible value of chosen attribute.
4. Assigning the appropriate subset of the data to each branch.
5. Recursively apply the same process on each branch until all instances belong to a single class, or a maximum depth is reached.

## 3.5 Performance Metrics

After classification, the quality of the model built on test data needs to be estimated. That is, the percentage with which the proposed model correctly distinguishes intrusion attacks from normal network traffic.

A confusion matrix is used in this research to evaluate the performance of the classification. The test dataset (without class labels) was fed to the trained classifier for prediction. Thereafter, a comparison between the actual class and the predicted class was performed by counting the number of correct and incorrect predictions for each class. The obtained values (correct and incorrect values) are presented in the confusion matrix table, where the actual classes are presented in rows and the predicted classes in columns. The arrangement of all predicted classes and actual class in the confusion matrix table for binary classification, in which all correct classifications lie along the diagonal (TP and TN) and the incorrect classifications correspond to numbers off the diagonal (FP and FN).$TN, FP, FN$, and $TP$ are defined as follows:

True Positive (TP): This is the number of intrusion attack instances that were predicted correctly as an attack

False Positive (FP): This is number of normal network traffic that are incorrectly classified as attack.

True Negative (TN): This is the number of normal traffic that were correctly predicted as normal traffics

False Negative (FN): This is the number of attack instances that were wrongly predicted to be normal traffics.

Based on the application of these parameters, performance

metrics such as accuracy, precision, false alarm rate, recall, and F1-measure were computed as follows:

**i.** Accuracy: offers a general view of the system's performance. It measures the proportion of the total number of correct classifications.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (22)$$

**ii.** Precision: In a classification task, the precision for a class is the number of TP (i.e. the number of items correctly labelled as belonging to the positive class) divided by the total number of elements labelled as belonging to the positive class (i.e. the sum of TP and FP, which are items incorrectly labelled as belonging to the class).

$$Precision = \frac{TP}{TP+FP} \quad (23)$$

**iii.** Recall: is the ratio of correctly predicted attacks in the actual class to all the number of actual attacks.

$$Recall = \frac{TP}{TP+FN} \quad (24)$$

**iv.** False Alarm Rate: is a ratio of misclassified instances.

$$FAR = \frac{FP}{TN+FP} \quad (25)$$

**v.** F-Score considers both the Precision and the Recall of the test to compute the score. It can be interpreted as a weighted average of the Precision and the Recall, where 1 is its best value and 0 its worst.

$$F1-Score = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (26)$$

# 4. RESULT AND DISCUSSION

## 4.1 Feature Selection Results

This section presents the outcome of two feature selection techniques, IG and PCA used to identify the most relevant network attributes for intrusion detection. In IG, features were ranked based on their gain values, while PCA ranked transformed features (principal components) based on their explained variance.

### *4.1.1 IG Results*

For the IG-based approach, features from the UNSW-NB15 dataset were ranked in descending order according to their respective information gain values. A grid-based selection strategy was employed, where subsets of features were incrementally selected in groups of five and evaluated using SVM and DT classifiers. The subset of features yielding the highest classification accuracy was retained as the optimal set.

Based on this evaluation, the top 10 features were selected as the most informative for detecting network intrusions. Figure 3 presents the ranking of the top ten (10) selected features.

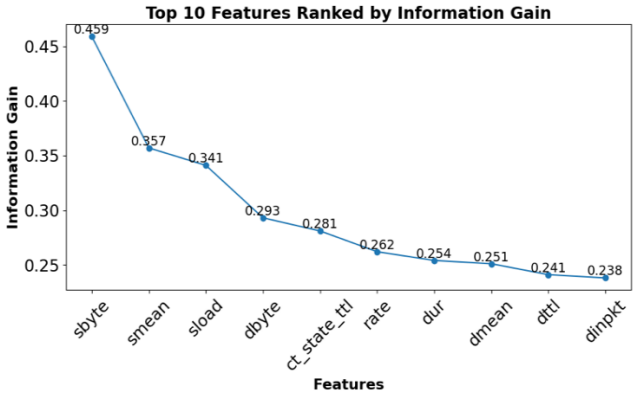


**Fig 3: Top 10 Feature Ranked by IG**

Based on Fig 3, the line plot shows a clear descending trend, with *sbyte* emerging as the most informative feature, followed by *smean* and *sload*. The sharp drop between the first and second features indicates that *sbyte* contributes substantially more discriminatory power than the remaining attributes. Beyond the top three features, the IG values decline more gradually, suggesting that while features such as *dbyte*, *ct_state_ttl*, and *rate* remain relevant, their individual contributions are comparatively moderate. The lower-ranked features (*dmean*, *dttl*, and *dinpkt*) still provide useful information but contribute less to the overall decision-making capability of the model.

### *4.1.2 PCA Extraction Results*

In contrast to IG, PCA is a feature extraction method that transforms the original feature space into a new set of orthogonal components, known as principal components (PCs). Each principal component is a linear combination of the original features and captures a portion of the dataset's variance.

The components are ordered by their explained variance, which is derived from their corresponding eigenvalues. For this study, the number of PCs was selected to preserve 90% of the total variance in the data. This threshold resulted in the selection of 10 principal components. Fig 4 provides a summary of these selected components along with their individual explained variance contributions

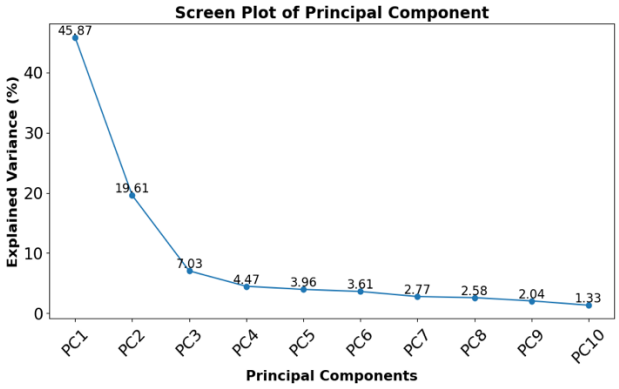


**Fig 4: Top 10 Feature Ranked by PCA**

The scree plot (Fig 4) shows that PC1 explains the largest proportion of variance (≈45.9%), followed by PC2 (≈19.6%), after which the explained variance drops sharply. Together, PC1 and PC2 account for approximately 65.5% of the total

variance, indicating that most of the information in the dataset is captured by the first two principal components. Beyond PC3, each additional component contributes less than 8% individually, with a gradual flattening of the curve from PC4 onward.

## 4.2 Model Results on Selected Features

Table 3 summarises the confusion matrix and performance evaluation results obtained on the UNSW-NB15 training and test datasets for the SVM and DT classifiers, using feature subsets derived from both PCA and IG.

### *4.2.1 Confusion Matrix Result*

For the C4.5 DT trained with PCA features, the confusion matrix showed a high number of TP (101,933) and TN (47,507), indicating that the classifier is effective at correctly identifying both positive and negative instances during training. However, the presence of 17,408 FP suggested that the model tends to incorrectly label a substantial number of negative instances as positive. While this behaviour contributes to the high recall, it also explains the elevated false alarm rate. The 8,493 FN, although relatively smaller compared to the TP, indicated that some positive cases were missed during training.

In comparison, the SVM trained with PCA features exhibited a different error profile. Although it recorded a high number of TP (91,838) and a very low number of FN (3,730), it also produced a large number of FP (27,503). This imbalance indicated that the SVM classifier under PCA is highly aggressive in predicting the positive class, which boosts recall but significantly increases false alarms. The confusion matrix therefore revealed that PCA-based features led SVM to sacrifice specificity for sensitivity during training.

When IG selected features were applied, the confusion matrix values improved noticeably for both classifiers. For the C4.5 DT, TP increased to 104,962, while FN drop sharply to 2,772, indicating a much stronger ability to correctly capture positive instances. At the same time, FP reduced to 14,379, and TP increased to 53,228, showing improved discrimination between classes. This balanced reduction in both FP and FN directly explains the observed increase in accuracy, precision, and recall.

The SVM trained with IG features shows a particularly striking change in its confusion matrix. The number of FP dropped dramatically to 4,445, while TP rose to 114,896, indicating that the classifier becomes far more selective and confident in its predictions. However, the increase in FN to 12,368 suggested that, although the model is now more conservative, it misses a larger number of positive instances. This trade-off is clearly reflected in the confusion matrix and explains why SVM achieves very high precision but comparatively lower recall under this configuration.

On the test set, the C4.5 Decision Tree with PCA features demonstrated near-ideal confusion matrix behaviour. The classifier records 45,326 TP, 37,000 TN, and almost no misclassifications, with only 6 FP and 0 false negatives. This indicated that the learned decision boundaries generalise extremely well to unseen data, achieving both perfect sensitivity and near-perfect specificity.

In contrast, the SVM under PCA on the test set exhibited a more error-prone pattern. While it correctly identified a large number of TP (35,245), it also produces a substantial number of FP (10,087) and FN (3,880). This indicates instability in generalisation, where the classifier struggles to consistently distinguish between classes when exposed to unseen data. The confusion matrix thus highlights that PCA-based dimensionality reduction does not preserve sufficient discriminative structure for SVM during testing.

With IG features, the C4.5 DT shows strong and consistent confusion matrix outcomes. It achieves 44,802 TP and 36,924 TN, with only 530 FP and 76 FN. This low error count demonstrated that the model maintains high sensitivity while keeping false alarms minimal, reinforcing its suitability for operational use.

The SVM classifier under IG on the test set presents a contrasting pattern. Although it achieves a high number of TP (43,980) and a relatively low number of FP (1,352), it suffers from a very large number of FN (17,951). This indicates that the model fails to detect a significant portion of positive instances, despite being highly precise when it does make positive predictions. The confusion matrix therefore reveals a critical limitation: SVM becomes overly conservative under IG during testing, leading to missed detections.

Across all experiments, the confusion matrix analysis reveals that C4.5 DT consistently maintains a better balance between FP and FN, especially on the test set. This balance is crucial in applications where both missed detections and false alarms carry significant consequences. In contrast, SVM exhibits strong sensitivity to feature selection, oscillating between high false alarm rates (under PCA) and high missed detection rates (under Information Gain).

### *4.2.2 Performance Result*

#### *4.2.2.1 Performance on the Training Set*

Under PCA-based feature reduction, the C4.5 Decision Tree achieves an accuracy of 85.23%, with a relatively high recall of 92.31%, indicating strong capability in correctly identifying positive instances. However, the false alarm rate of 26.81% suggests a notable tendency to misclassify negative instances as positives. The precision of 85.41% and F1-score of 88.73% reflect a reasonably balanced performance but also highlight room for improvement in reducing false positives. In contrast, the SVM classifier under PCA exhibited a lower accuracy of 82.19% and a reduced precision of 76.95%, despite achieving a very high recall of 96.10%. This indicates that while SVM is highly effective at capturing positive cases, it does so at the cost of a higher false positive rate (34.48% FAR). Consequently, the F1-score drops to 79.48%, suggesting that PCA-based features may not optimally separate classes for the SVM model during training.

When Information Gain is used for feature selection, both classifiers demonstrate substantial performance improvements. The C4.5 Decision Tree records an accuracy of 90.22%, with recall rising to 97.43% and FAR decreasing significantly to 21.27%. The corresponding F1-score of 92.45% indicates a strong balance between precision and recall. This improvement suggests that IG-selected features better capture class-discriminative information relevant to decision tree splits. The SVM classifier also benefited markedly from IG, achieving an accuracy of 90.41%, precision of 96.28%, and an F1-score of 93.17%, the highest among all training configurations. Notably, the FAR drops to 9.25%, indicating a much lower false positive tendency compared to PCA-based training. These results demonstrate that SVM is particularly sensitive to the quality of feature selection and performs optimally when

trained on highly informative features identified through Information Gain.

### *4.2.2.2 Performance on the Test Set*

On the test dataset, the C4.5 Decision Tree under PCA exhibits near-perfect performance, with an accuracy of 99.99%, precision of 99.99%, recall of 100%, and a negligible FAR of 0.02%. This exceptional performance suggests that the PCA-transformed features align very well with the learned decision boundaries of the tree model, enabling strong generalisation to unseen data. In contrast, the SVM classifier performs significantly worse on the PCA-based test set. Although recall remains high at 90.08%, the accuracy drops to 79.39%, and the FAR increases. The precision of 77.75% and F1-score of 83.46% indicate that SVM struggles to generalise effectively under PCA, potentially due to the loss of discriminative class-specific information during dimensionality reduction.

Under IG–based feature selection, the C4.5 DT demonstrated excellent generalisation performance, achieving an accuracy of 99.26%, precision of 98.83%, recall of 99.83%, and an F1-score of 99.33%. The low FAR of 1.42% confirms that the model maintains strong discrimination between positive and negative classes even on unseen data. The SVM classifier under IG showed mixed performance. While precision remains very high at 97.02%, recall drops substantially to 71.01%, indicating that a significant proportion of positive instances are misclassified. This imbalance leads to a reduced accuracy of 76.56% and an F1-score of 82.00%. The results suggest that although IG-selected features help SVM reduce false positives, they also limit its ability to capture all relevant positive patterns during testing.

### *4.2.3 ROC Curve on the Test Set*

The ROC analysis presented in Figure 6 clearly demonstrated that DT–based models, particularly when combined with PCA-based feature reduction, offer superior intrusion detection capabilities. They achieve a favourable balance between high detection rates and low false alarm rates, which is essential for effective and practical IDS deployment. Conversely, SVM models show greater sensitivity to feature-reduction techniques, but failed to consistently maintain strong detection performance under strict false alarm constraints. These findings reinforce the suitability of C4.5 DT with PCA as the most reliable and operationally effective choice for intrusion detection in the UNSW-NB15 dataset.

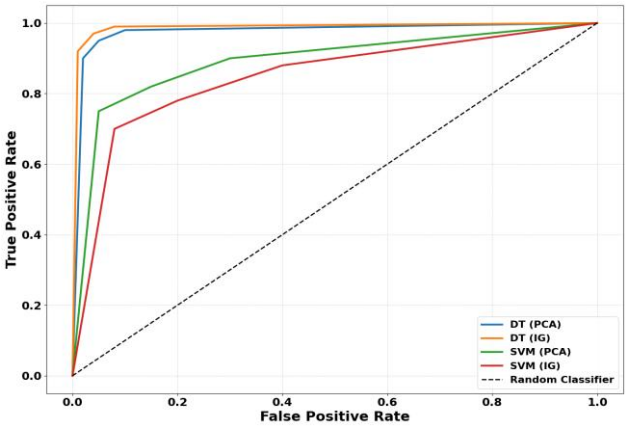


**Fig 6: ROC Curve of Studied Models**

### *4.2.4 Performance Comparison of Existing Studies*

Table 4 compared the proposed approach with existing intrusion detection systems evaluated on the UNSW-NB15 dataset. Most prior studies report classification accuracies ranging between 75% and 90%, particularly when using all features or shallow learning models. In contrast, the proposed C4.5 Decision Tree combined with PCA achieves near-perfect detection accuracy (99.99%) on the test set. This improvement can be attributed to effective removal of noisy and redundant features through PCA and Information Gain, which simplifies the decision space and enhances class separability. The proposed lightweight decision tree model therefore offers a favorable trade-off between accuracy, efficiency, and explainability, making it well suited for real-time intrusion detection environments.

**Table 3. Confusion Matrix and Performance Evaluation Results**

| | Confusion Matrix | | | | Performance Measures | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Performance on Training Set (PCA) | | | | | | | | |
| Classifier | TP | FP | TN | FN | Accuracy | Precision | Recall | FAR | F1-Score |
| C4.5 DT | 101,933 | 17,408 | 47,507 | 8,493 | 85.23% | 85.41% | 92.31 | 26.81 | 88.73 |
| SVM | 91,838 | 27,503 | 52,270 | 3,730 | 82.19% | 76.95% | 96.10 | 34.48 | 79.48 |
| | Performance on Training set (Information Gain) | | | | | | | | |
| C4.5 DT | 104,962 | 14,379 | 53,228 | 2,772 | 90.22% | 87.95% | 97.43 | 21.27 | 92.45 |
| SVM | 114,896 | 4,445 | 43,632 | 12,368 | 90.41% | 96.28% | 90.28 | 9.25 | 93.17 |
| | Performance on Test Set (PCA) | | | | | | | | |
| C4.5 DT | 45,326 | 6 | 37, 000 | 0 | 99.99% | 99.99% | 100.00% | 0.02 | 100.00 |
| SVM | 35,245 | 10,087 | 33,120 | 3880 | 79.39% | 77.75% | 90.08 | 0.23 | 83.46 |
| | Performance on Test Set (Information Gain) | | | | | | | | |
| C4.5 DT | 44,802 | 530 | 36,924 | 76 | 99.26% | 98.83% | 99.83 | 1.42 | 99.33 |
| SVM | 43,980 | 1,352 | 19,049 | 17,951 | 76.56% | 97.02% | 71.01 | 6.63 | 82.00 |

\


**Fig 5: Performance comparison across studied models**

**Table 4. Performance Comparison with Existing Studies**

| Study | Dataset | Method/ Model | Feature Handling | Classification Type | Reported Accuracy (%) |
|---|---|---|---|---|---|
| [21] | UNSW-NB15 | Information Gain + MLP | Filter-based IG | Binary | 76.96 |
| [6] | UNSW-NB15 | SVM, MLP | All features | Binary | ~79.65 |
| [20] | UNSW-NB15 | Association Rules vs SVM | Full feature set | Binary | 90.41 (SVM) |
| [23] | UNSW-NB15 | Stacked Ensemble (KNN + NB + DT) | Full features | Binary | ~93.00 |
| [22] | UNSW-NB15 | SVD + LSTM | Feature extraction | Binary (DDoS only) | >90.00 |
| This Study | UNSW-NB15 | C4.5 Decision Tree + PCA | Feature extraction (10 PCs) | Binary | 99.99 |
| This Study | UNSW-NB15 | C4.5 Decision Tree + IG | Top-10 selected features | Binary | 99.26 |

## 5. CONCLUSION

This study demonstrated that an optimized feature-based lightweight intrusion detection system can achieve high detection accuracy and low false alarm rates when appropriate feature reduction and classifier selection are employed. By integrating PCA and IG for feature optimization, the proposed system effectively reduces dimensionality while preserving attack-discriminative information. The dual-classifier evaluation reveals that the C4.5 DT, particularly when combined with PCA, provided superior and more stable intrusion detection performance compared to SVM across both training and test sets. These findings confirm that lightweight

IDS architectures can deliver robust security performance without relying on high computational complexity, making the proposed approach suitable for real-time and resource-constrained environments.

Future research will focus on extending the proposed lightweight intrusion detection framework to multi-class attack classification in order to capture finer-grained intrusion behaviours. Additional feature optimization techniques, including hybrid filter–wrapper methods and embedded learning approaches, will be explored to further enhance detection efficiency. The integration of adaptive and online learning mechanisms is also envisaged to enable real-time model updates in dynamic network environments.

## 6. REFERENCES

[1] Kaur, T., Malhotra, V., and Singh, D. 2014. Comparison of network security tools-firewall, intrusion detection system and Honeypot. *Int. J. Enhanced Res. Sci. Technol. 3(2):* 200-204.

[2] Adetunmbi, A.O., Falaki S. O., Adewale O. S., and Alese B. K. 2008. Network Intrusion Detection Based on Rough Set and K-Nearest Neighbor. International Journal of Computing and ICT Research, 2, 60-66.

[3] Kendall, K. 1999. A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Master's Thesis. *Massachusetts Institute of Technology*, Cambridge, MA

[4] Verwoerd, T., and Hunt, R. 2002. Intrusion detection techniques and approaches. *Computer communications*, *25*(15), 1356-1365.

[5] Olayinka, T. C., Ugwu, C. C., Okhuoya, O. J., Adetunmbi, A. O., and Popoola, O. S. 2022. Combating network intrusions using machine learning techniques with multilevel feature selection method. In *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)* (pp. 1-5). IEEE.

[6] Popoọla, O. S., Ayogu, I. I., Mebawondu, Ọ. J., Ugwu, C. C., and Adetunmbi, A. O. 2022. An evolutionary approach towards achieving enhanced intrusion detection system. In *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)* (pp. 1-5). IEEE.

[7] Mebawondu, O. J., Popoọla, O. S., Ayogu, I. I., Ugwu, C. C., and Adetunmbi, A. O. 2022. Network intrusion detection models based on Naives Bayes and C4. 5 algorithms. In *2022 IEEE Nigeria 4th international conference on disruptive technologies for sustainable development (NIGERCON)* (pp. 1-5). IEEE.

[8] Hozouri, A., Mirzaei, A., & Effatparvar, M. (2025). A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges. *Discover Artificial Intelligence*, *5*(1), 314.

[9] Lin, Y., Zhang, Y., and Ou, Y. 2010. The Design and Implementation of Host-Based Intrusion Detection System. Third International Symposium on Intelligent Information Technology and Security Informatics.

[10] Larriva-Novo X, Sánchez-Zas C, Villagrá VA, Marín-Lopez A, Berrocal J. (2023). Leveraging explainable artificial intelligence in real-time cyberattack identification: intrusion detection system approach. Appl Sci.13(15):8587.

[11] Arreche O, Guntur T, Abdallah M. (2024) XAI-IDS: toward proposing an explainable artificial intelligence framework for enhancing network intrusion detection systems. Appl Sci.14(10):4170.

[12] Dini, P., Elhanashi, A., Begni, A., Saponara, S., Zheng, Q., & Gasmi, K. (2023). Overview on intrusion detection systems design exploiting machine learning for networking cybersecurity. *Applied Sciences*, *13*(13), 7507.

[13] Singh, S., Silakari, S., and Patel, R., 2011. An efficient feature reduction technique for intrusion detection system. International Conference on Machine Learning and Computing Singapore, 3(8): 147-153

[14] Boulaiche, A., Bouzayani, H., and Adi, K. 2012. A Quantitative Approach for Intrusion Detection and Prevention based on Statistical N-gram Models. 3rd International Conference on Ambient Systems, Networks and Technology (ANT). Pp. 450-457

[15] Al-mamory, S. O., and Jassim, F. S. 2015. On the designing of two grains levels network intrusion detection system. *Karbala International Journal of Modern Science*, *1*(1), 15-25.

[16] Obimbo, C, Zhou, H. and Wilson, R. 2011. Multiple SOFMs Working Cooperatively in a Vote-Based Ranking System for Network Intrusion Detection. Procedia Computer Science. 6(12): 219-224

[17] Duque, S., and bin Omar, M. N. 2015. Using data mining algorithms for developing a model for intrusion detection system (IDS). *Procedia Computer Science*, *61*, 46-51.

[18] Aladesote, O. I., Alese, B. K., and Dahunsi, F. 2014. Intrusion detection technique using hypothesis testing. In *Proceedings of the World Congress on Engineering and Computer Science* (Vol. 1, No. 2, pp. 978-988).

[19] Rai, K., Devi, M. S., and Guleria, A. 2016. Decision tree-based algorithm for intrusion detection. *International Journal of Advanced Networking and Applications*, *7*(4), 2828.

[20] Sarumi, O. A., Adetunmbi, A. O., and Adetoye, F. A. 2020. Discovering computer networks intrusion using data analytics and machine intelligence. *Scientific African*, *9*, e00500.

[21] Mebawondu, J. O., Alowolodu, O. D., Mebawondu, J. O., and Adetunmbi, A. O. 2020. Network intrusion detection system using supervised learning paradigm. *Scientific African*, *9*, e00497.

[22] Ugwu, C. C., Obe, O. O., Popoọla, O. S., and Adetunmbi, A. O. 2021. A distributed denial of service attack detection system using long, short-term memory with Singular Value Decomposition. In *2020 IEEE 2nd International conference on Cyberspac (CYBER NIGERIA)* (pp. 112-118). IEEE.

[23] Olasehinde, O. O. 2020. A Stacked Ensemble Intrusion Detection Approach for the Protection of Information

System. *Int. J. Inf. Secur. Res*, *10*, 910-923.

[24] Moustafa, N., and Slay, J. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (MilCIS)* (pp. 1-6). IEEE.